Signal Processing 166 (2020) 107256

Contents lists available at ScienceDirect

Signal Processing

journal homepage: www.elsevier.com/locate/sigpro

Making linear prediction perform like maximum likelihood in Gaussian autoregressive model parameter estimation

Çağatay Candan

Department of Electrical and Electronics Engineering, Middle East Technical University (METU), Ankara 06800, Turkey

ARTICLE INFO

Article history:
Received 17 February 2019
Revised 9 June 2019
Accepted 9 August 2019
Available online 17 August 2019

Keywords:
Autoregressive process
Yule-Walker equations
Burg's method
Forward-backward prediction
Parameter estimation
Spectrum estimation

ABSTRACT

A two-stage method for the parameter estimation of Gaussian autoregressive models is proposed. The proposed first stage is an improved version of the conventional forward-backward prediction method and can be interpreted as its weighted version with the weights derived from the arithmetic mean of the log-likelihood functions for different conditioning cases. The weighted version is observed to perform better than the conventional forward-backward prediction method and other linear prediction based methods (correlation method, covariance method, Burg's method etc.) in terms of attained likelihood value. The proposed second stage uses the estimate of the first stage as the initial condition and approximates the highly non-linear log-likelihood function with a quadratic function around the initial estimate. The optimization of the quadratic cost function yields the optimal perturbation vector that locally maximizes the likelihood in the vicinity of the initial condition. The proposed method is compared with other methods and it has been observed that the likelihood value attained at the end of two-stages is almost identical to the value attained by higher complexity numerical-search based optimization tools in a wide range of experiments. The maximum likelihood-like performance at a significantly lower implementation cost makes the proposed method especially valuable for the applications with short data-records and limited computational resources.

1. Introduction

Autoregressive (AR) modeling of random processes is a powerful tool of statistical signal processing utilized in speech processing, seismic signal processing, radar signal processing and several other applications [1–4]. The preference of AR models over the moving average (MA) or autoregressive moving average (ARMA) models partially stems from the availability of computationally feasible estimation techniques involving only linear equation systems. Powerful modeling capabilities of AR systems, along with their practicality, kept the estimation of AR model parameters as a key problem of focus for many decades. In this paper, we present a low complexity parameter estimation method, requiring no more than the solution of a linear equation system, that works almost as well as the maximum likelihood (ML) method. The suggested method can be especially useful in applications with short data-records and limited computational resources.

The computational difficulties associated with the exact maximum likelihood solution led to the development of several low cost AR parameter estimation methods. Among these methods, Yule-Walker equations (also known as the autocorrelation method), Burg's method and forward-backward prediction based method can be considered as methods based on the linear prediction operation [1]. These methods are highly practical and they can be interpreted as substitutes for a much higher complexity maximum likelihood estimator. It is a generally accepted fact that the best performing method among these methods is the forward-backward prediction method, trailed by Burg's method, covariance method and Yule-Walker method, [5, Section 6.4]. The suggested method, weighted forward-backward prediction method, is also a member of the same class. Here we show that the suggested method is better than other members in terms of likelihood maximization; but, suffers from the stability problem at small sample sizes, a problem common to some members of this class.

In [6], Kay proposed an approximate ML estimator, mimicking Burg's approach, that produces exact ML estimate only for AR(1) process. Kay suggests to solve for the kth reflection coefficient by maximizing the likelihood function while keeping the earlier reflection coefficients fixed. Vis and Scharf have improved the efficiency of Kay's estimator by integrating the Levinson recursion to the estimator [7]. Tuan extended Kay's estimator, at the expense of more computation, by allowing iterative optimization of reflection coefficients [8]. Whorter and Scharf have given the exact ML solution for AR model parameters in [9]. Unfortunately, the exact

E-mail address: ccandan@metu.edu.tr

https://doi.org/10.1016/j.sigpro.2019.107256
0165-1684/© 2019 Elsevier B.V. All rights reserved.

ML solution requires root finding of very high degree polynomials. To illustrate the difficulty of the problem, the root finding operation of a 6561 degree polynomial is required for the estimation of AR(4) parameters, [9, Fig. 4].

The main goal of this paper is to present a method that is statistically close to maximum likelihood estimation and computationally faster than other existing methods. The suggested method consists of two stages. In the first stage, we generalize the conventional forward-backward prediction approach and propose its weighted version. The performance of weighted forward-backward prediction is generally much better than the conventional one and its computation requirements are the same. A second stage is proposed to further improve the likelihood value. In the second stage, the result of the first stage is taken as the initial condition and the perturbation vector that maximizes the likelihood value in the neighborhood of the initial condition is found. The numerical results indicate that the suggested method generates almost identical likelihood values to the ML solution in AR(1) and AR(2) cases, the cases for which exact ML solutions are available. For higher order processes, for which there is no exact ML solution, the suggested method outperforms other linear prediction based alternatives in the maximization of the likelihood function and works as well as numerical-search based non-linear optimization tools in spite of its lower computational requirements. We can summarize the main contributions of this study as the development of an improved version of conventional forward-backward prediction scheme (first stage method) and an efficient solver for the local maxima of the likelihood function around an initial condition (second stage method).

The main application target for the suggested method is the AR parameter estimation problems with short data records, as in radar signal processing [3,10]. In many radar signal processing applications, the computational sources are limited and do not allow a real-time implementation of a general purpose optimization tool. More specifically, a pulse-Doppler radar system with 10 MHz pulse bandwidth utilizing 100 pulses in a coherent processing interval generates 100,000 vectors per second where each vector contains 100 samples (slow/fast time decomposition). AR modeling can be used to model the clutter process in each observation vector. Unfortunately, it is not feasible to apply general purpose non-linear optimizers on each vector due to the inflow rate of vectors. Typically, low complexity methods, such as Burg's method, is utilized to model the clutter power spectral density. The suggested fixed latency, reduced complexity, ML-like performing method can be useful in such applications. In the other extreme of long data records, as in speech processing, the frequency domain methods can be preferred [11,12]. As the record size increases, the performance difference between alternative methods vanishes [5] and the method with the lowest computational load is typically preferred.

There are similar works in the literature that aim to generate good approximations to the ML solution such as Tufts and Kumaresan, [13], where the conventional linear prediction is "transformed" to attain a ML like performance for frequency estimation problem with multiple sinusoids. Present work can be interpreted as a similar effort on a different problem, AR model parameter estimation problem.

2. Preliminaries

The elements of the $N \times 1$ random vector $\mathbf{x} = [x_1 \quad x_2 \quad \dots \quad x_N]^T$ are the samples of the Gaussian AR(P) process, that is assumed to be synthesized with the application of zero-mean Gaussian distributed white noise with variance σ_ϵ^2 to the filter with the transfer function

$$H(z) = \frac{1}{1 + a_1 z^{-1} + a_2 z^{-2} + \dots + a_P z^{-P}}.$$

The autocorrelation matrix of the vector $\mathbf{x}$ can be written as $\mathbf{R} = \mathbf{R}_{f,N}\sigma_\epsilon^2$. Here, $\mathbf{R}_{f,N}$ is a $N \times N$ Hermitian Toeplitz matrix whose first column entries are $r_f[k]$ for $k = \{0, 1, \dots, N-1\}$, where $r_f[k] = E\{x[n]x^*[n-k]\} = h[n] * h^*[-n] = \mathcal{Z}^{-1}\{H(z)H^*(1/z^*)\}$,

$$\mathbf{R}_f = \begin{bmatrix} r_f[0] & r_f[-1] & \dots & r_f[-N+1] \\ r_f[1] & r_f[0] & \dots & r_f[-N+2] \\ \vdots & \vdots & \ddots & \vdots \\ r_f[N-1] & r_f[N-2] & \dots & r_f[0] \end{bmatrix}. \quad (1)$$

The parameter σ_ϵ^2 denotes the variance of process noise at the input of the synthesis filter generating AR(P) process or equivalently, the mean square error (MSE) value of the Pth or higher order linear prediction filter for the same process.

The density of $N \times 1$ circularly symmetric complex Gaussian vector $\mathbf{x}$ is denoted by $\mathrm{CN}(\mathbf{x}; \mathbf{0}, \sigma_\epsilon^2 \mathbf{R}_f)$ and expressed as

$$f_{\mathbf{x}}(\mathbf{x}; \sigma_\epsilon^2, \mathbf{a}) = \frac{1}{\pi^N \sigma_\epsilon^{2N} |\mathbf{R}_{f,N}|} \exp\left(-\frac{1}{\sigma_\epsilon^2} \mathbf{x}^H \mathbf{R}_{f,N}^{-1} \mathbf{x} \right)$$

where $\mathbf{a} = \begin{bmatrix} a_1 & a_2 & \dots & a_P \end{bmatrix}^T$. The problem is to find $\mathbf{a}$ and σ_ϵ^2 such that the likelihood function $f_{\mathbf{x}}(\mathbf{x}; \sigma_\epsilon^2, \mathbf{a})$ is maximized. Taking the logarithm of likelihood function, we get

$$\Lambda(\sigma_\epsilon^2, \mathbf{a}) \overset{c}{=} -\left(N \log \sigma_\epsilon^2 + \log |\mathbf{R}_{f,N}| + \frac{1}{\sigma_\epsilon^2} \mathbf{x}^H \mathbf{R}_{f,N}^{-1} \mathbf{x} \right). \quad (2)$$

Here $\overset{c}{=}$ indicates equality of both sides up to an additive constant, not affecting the subsequent optimization. Optimizing (2) for σ_ϵ^2, by differentiation, yields the estimate of $\widehat{\sigma}_\epsilon^2 = \frac{1}{N} \mathbf{x}^H \mathbf{R}_{f,N}^{-1} \mathbf{x}$. Inserting the optimized $\widehat{\sigma}_\epsilon^2$ estimate into the log-likelihood relation, we get (also see [6])

$$\Lambda(\widehat{\sigma}_\epsilon^2, \mathbf{a}) \overset{c}{=} -\left\{ N \log(\mathbf{x}^H \mathbf{R}_{f,N}^{-1} \mathbf{x}) + \log |\mathbf{R}_{f,N}| \right\}. \quad (3)$$

The maximization of $\Lambda(\widehat{\sigma}_\epsilon^2, \mathbf{a})$ given in (3), a quantity also called the compressed likelihood by Scharf et al. in [7], is trivially equivalent to the minimization of $N \log(\mathbf{x}^H \mathbf{R}_{f,N}^{-1} \mathbf{x}) + \log(|\mathbf{R}_{f,N}|)$. Note that the first term of the compressed likelihood function, i.e. $N \log(\mathbf{x}^H \mathbf{R}_{f,N}^{-1} \mathbf{x})$, depends on the observation vector $\mathbf{x}$. The second term, i.e. $\log(|\mathbf{R}_{f,N}|)$, is data-independent and related to the entropy of the Gaussian vector whose parameters are to be estimated. It can be said that the second term penalizes the estimates according to the entropy of the resulting process.

Forward prediction: The joint density for the entries of vector $\mathbf{x}$ can also be expressed as follows:

$$\begin{aligned} f_{\mathbf{x}}(\mathbf{x}_{1:N}) &= f(\mathbf{x}_{1:P}) f(\mathbf{x}_{P+1:N} | \mathbf{x}_{1:P}) \\ &= f(\mathbf{x}_{1:P}) \prod_{n=P+1}^{N} \mathrm{CN}(x_n; -\mathbf{a}^T \mathbf{x}_{n-1:-1:n-P}, \sigma_\epsilon^2) \end{aligned} \quad (4)$$

The vectors with the subscript, such as $\mathbf{x}_{1:P}$, indicate column vectors whose entries are restricted to the entries of the original vector with indices denoted in the subscript, i.e. $\mathbf{x}_{1:P} = [x_1 \quad x_2 \quad \dots \quad x_P]^T$. Similarly, the subscript as in $\mathbf{x}_{n-1:-1:n-P}$ denotes entries in the reversed order, i.e. $\mathbf{x}_{n-1:-1:n-P} = [x_{n-1} \quad x_{n-2} \quad \dots \quad x_{n-P}]^T$. With this notation, the log-likelihood function for the factorization given in (4) can be expressed as

$$\begin{aligned} \Lambda(\sigma_\epsilon^2, \mathbf{a}) \overset{c}{=} -\Bigg(& N \log \sigma_\epsilon^2 + \log |\mathbf{R}_{f,P}| \\ & + \frac{1}{\sigma_\epsilon^2} \underbrace{\left[\mathbf{x}_{1:P}^H \mathbf{R}_{f,P}^{-1} \mathbf{x}_{1:P} + \sum_{n=P+1}^{N} |e_f[n]|^2 \right]}_{(5)} \Bigg) \end{aligned} \quad (5)$$

where $e_f[n] = x_n + \mathbf{a}^T \mathbf{x}_{n-1:-1:n-P}$ is the forward prediction error and $\widehat{x}_n = -\mathbf{a}^T \mathbf{x}_{n-1:-1:n-P}$ is the forward prediction result of the sample x_n.

We note that for all $M \geq P$ dimensional $\mathbf{R}_f$ matrices of AR(P) processes, the determinant of $\mathbf{R}_f$ is identically the same, that is $|\mathbf{R}_{f,M}| = |\mathbf{R}_{f,P}| = \prod_{i=1}^{P} (1 - |k_i|^2)^{-i}$ for $M \geq P$, [6, Eq. 14]. Here k_i is the ith reflection coefficient. Given this, when (2) and (5) are compared, we note that the first two terms of both equations are identical. Leading to the fact that the data dependent term in (2) ($\mathbf{x}^H \mathbf{R}_{f,N}^{-1} \mathbf{x}$) is identical to the term in the square brackets of (5).

Optimizing over σ_ϵ^2, we get

$$\Lambda(\widehat{\sigma}_\epsilon^2, \mathbf{a}) \overset{c}{=} -\left\{ N \log [\cdot]_{(5)} + \log |\mathbf{R}_f| \right\}$$

where $[\cdot]_{(5)} = [\mathbf{x}_{1:P}^H \mathbf{R}_{f,P}^{-1} \mathbf{x}_{1:P} + \sum_{n=P+1}^{N} |e_f[n]|^2]$ is the contents of the square bracket, the data dependent term in (5).

The conventional forward prediction method neglects all the terms in the compressed likelihood of $\Lambda(\widehat{\sigma}_\epsilon^2, \mathbf{a})$ except the prediction error term, $\sum_{n=P+1}^{N} |e_f[n]|^2$. This method is also known as the covariance method, [1].

Backward Prediction: Backward prediction is initiated with the following factorization of the joint density,

$$\begin{aligned} f_{\mathbf{X}}(\mathbf{x}_{1:N}) &= f(\mathbf{x}_{N:-1:N-P+1}) f(\mathbf{x}_{1:N-P} | \mathbf{x}_{N:-1:N-P+1}) \\ &= f(\mathbf{x}_{N:-1:N-P+1}) \prod_{n=1}^{N-P} \mathrm{CN}(x_n; -\mathbf{a}^H \mathbf{x}_{n+1:n+P}, \sigma_\epsilon^2). \end{aligned} \tag{6}$$

Following the same route, the log-likelihood function can be expressed as

$$\Lambda(\sigma_\epsilon^2, \mathbf{a}) \overset{c}{=} -\Bigg(N \log \sigma_\epsilon^2 + \log |\mathbf{R}_{f,P}| + \frac{1}{\sigma_\epsilon^2} \left[\mathbf{x}_{N:-1:N-P+1}^H \mathbf{R}_{f,P}^{-1} \mathbf{x}_{N:-1:N-P+1} + \sum_{n=1}^{N-P} |e_b[n]|^2 \right]_{(7)} \Bigg)$$

where $e_b[n] = x_n + \mathbf{a}^H \mathbf{x}_{n+1:n+P}$ is the backward prediction error and $\widehat{x}_n = -\mathbf{a}^H \mathbf{x}_{n+1:n+P}$ is the backward predictor.

Optimizing over σ_ϵ^2, we get

$$\Lambda(\widehat{\sigma}_\epsilon^2, \mathbf{a}) \overset{c}{=} -\left\{ N \log [\cdot]_{(7)} + \log |\mathbf{R}_f| \right\}.$$

As in forward prediction, the backward prediction method ignores all the terms of $[\cdot]_{(7)}$ except the quadratic term representing the total squared backward prediction error, $\sum_{n=1}^{N-P} |e_b[n]|^2$. Here $[\cdot]_{(7)}$ denotes the contents of the square brackets in (7). We prefer to use $[\cdot]_{(5)}$ and $[\cdot]_{(7)}$ notation to illustrate the similarities between different prediction methods.

Forward-Backward Prediction: The log-likelihood functions given by (2), (5) and (7) are different factorizations of the same likelihood expression. The difference arises from the conditioning of variables that the density is written.

The forward-backward method can be motivated by considering the arithmetic average of the log-likehoods given by (5) and (7). Taking the arithmetic average of Eqs. (5) and (7) as the objective function and optimizing over σ_ϵ^2, we get

$$\Lambda(\widehat{\sigma}_\epsilon^2, \mathbf{a}) \overset{c}{=} -\frac{1}{2} \left\{ N \log \left\{ [\cdot]_{(5)} + [\cdot]_{(7)} \right\} + \log |\mathbf{R}_f| \right\}.$$

The forward-backward prediction method ignores all the terms except the quadratic terms in $[\cdot]_{(5)} + [\cdot]_{(7)}$. Stated differently, the forward-backward prediction method optimizes $\mathbf{a}$ such that $\sum_{n=P+1}^{N} |e_f[n]|^2 + \sum_{n=1}^{N-P} |e_b[n]|^2$ is minimized.

Once the filter coefficients are estimated via some method, the maximum likelihood estimate of the remaining parameter (σ_ϵ^2) is given by $\widehat{\sigma}_\epsilon^2 = \frac{1}{N} \mathbf{x}^H \mathbf{R}_{f,N}^{-1} \mathbf{x}$. The straightforward application of this step requires on the order of N^3 operations due to inversion of $\mathbf{R}_{f,N}$ matrix. By carefully incorporating the Levinson recursion, the computational load can be reduced to order of $N \times P$ multiplications with the method given in Algorithm 1. This algorithm is a corrected and extended version of a related algorithm in [7].

Algorithm 1: Efficient calculation of the quadratic form with the inverse AR(P) covariance matrix. (Requires on the order of N(3P+2) multiplications per execution).

```
output = function xHinvRx ($a, x$);
    Input : $a = [1 \quad a_1 \quad a_2 \quad \ldots \quad a_P]$.
            $x$ : $N \times 1$ vector
    Output: $\mathbf{x}^H \mathbf{R}_f^{-1} \mathbf{x}$
gamma = atog(a); %Step-down recursion, [1, p.236]
N = length(x); sigmasq = 1/N*real(x'*x);
e = x; f = x;
alpha = abs(e(1))^2; beta = abs(f(end))^2;
etilde = e(2:end); ftilde = f(1:end-1);
c = etilde'*ftilde; d = 2*N*sigmasq - alpha - beta;
hsq = 0; ARorder = length(a) - 1;
for order = 1:ARorder do
    k = gamma(order);
    V = (1 + abs(k)^2)*d + 4*real(k*c);
    hsq = (1 - abs(k)^2)*(hsq + alpha + beta);
    e = etilde + k*ftilde;
    f = conj(k)*etilde + ftilde;
    alpha = abs(e(1))^2; beta = abs(f(end))^2;
    etilde = e(2:end); ftilde = f(1:end-1);
    c = etilde'*ftilde;
    d = V - alpha - beta;
end
output = (hsq + V)/2;
```

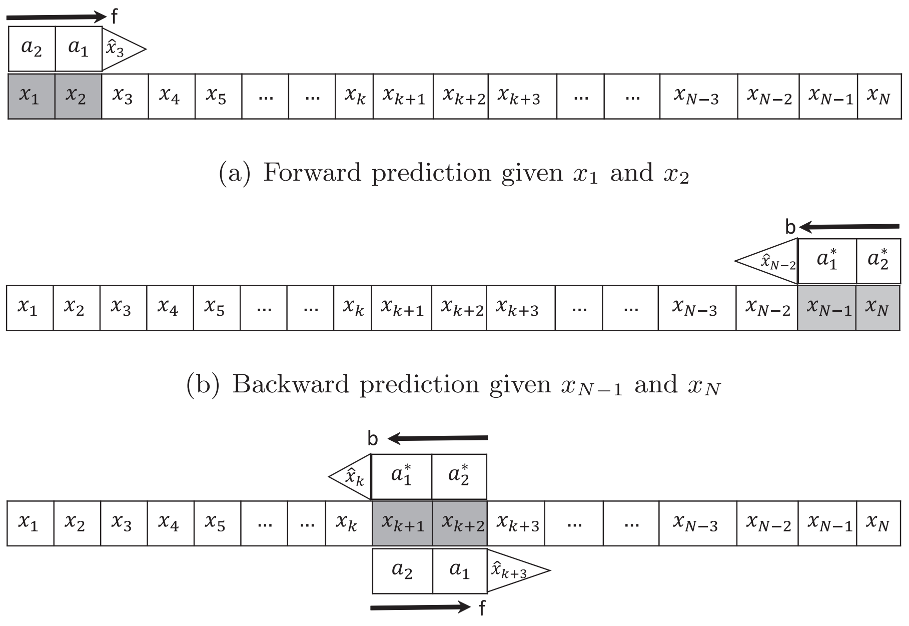
3. Proposed method

The proposed method consists of two stages. The first stage generates a good initial estimate around which the likelihood function is locally maximized. There are several methods in the literature that can be used to generate a suitable initial condition for the second stage. Yet, we propose a novel linear prediction based method which can also be utilized on its own in many applications.

3.1. First stage: Weighted forward-Backward prediction

Fig. 1 illustrates different forward/backward prediction schemes for second order prediction. In the top panel of this figure, the forward predictor is shown where observations $\{x_1, x_2\}$ is assumed to be available for the prediction of x_3. The observations are indicated by dark colored cells. By weighting the observation cells with coefficients a_1 and a_2, as described in the preliminaries, the prediction for the next cell is generated, $\widehat{x}_3 = -(a_1 x_2 + a_2 x_1)$. The forward prediction can be repeated for all possible pairs of x_{k-1} and x_{k-2}, $k = \{3, \ldots, N\}$. That is, the predictor can be "translated" step-by-step in the direction shown with $\mathbf{f}$ and at each step, a prediction for the next cell is generated by weighting and summing the contents of dark colored cells.

Fig. 1b illustrates the backward prediction resulting from the conditioning on $\{x_{N-1}, x_N\}$. The interpretation is similar to the forward prediction. The only difference is the direction of predictor motion. Hence, the forward and backward prediction methods results from the conditioning of the joint distribution with the samples at the beginning and end of the observation vector $\mathbf{x}$.


f
a_2 a_1 $\hat{x}_3$
x_1 x_2 x_3 x_4 x_5 … … x_k x_{k+1} x_{k+2} x_{k+3} … … x_{N-3} x_{N-2} x_{N-1} x_N
b
$\hat{x}_{N-2}$ a_1^* a_2^*
b
$\hat{x}_k$ a_1^* a_2^*
a_2 a_1 $\hat{x}_{k+3}$
f

(a) Forward prediction given x_1 and x_2

(b) Backward prediction given x_{N-1} and x_N

(c) Forward and backward prediction given x_{k+1} and x_{k+2}

Fig. 1. Forward and backward prediction for a second order predictor.

Fig. 1c illustrates the forward and backward prediction when conditioning is generalized to arbitrary consecutive samples, say $\{x_{k+1}, x_{k+2}\}$. For an arbitrary conditioning of $\mathbf{x}_{k+1:k+P}$, the joint density for the observations can be factorized as

$$\begin{aligned} f_{\mathbf{X}}(\mathbf{x}_{1:N}) &= f(\mathbf{x}_{k+1:k+P}) f(\mathbf{x}_{1:k}|\mathbf{x}_{k+1:k+P}) f(\mathbf{x}_{k+P+1:N}|\mathbf{x}_{k+1:k+P}) \\ &= f(\mathbf{x}_{k+1:k+P}) \prod_{n=1}^{k} \mathrm{CN}(x_n; -\mathbf{a}^H \mathbf{x}_{n+1:n+P}, \sigma_\epsilon^2) \prod_{n=k+P+1}^{N} \mathrm{CN}(x_n; \\ &\quad -\mathbf{a}^T \mathbf{x}_{n-1:-1:n-P}, \sigma_\epsilon^2). \end{aligned} \quad (7)$$

From (7), the log-likelihood function can be written as

$$\Lambda(\sigma_\epsilon^2, \mathbf{a}) \overset{c}{=} -\Bigg(N \log \sigma_\epsilon^2 + \log |\mathbf{R}_{f,P}| + \frac{1}{\sigma_\epsilon^2} \left[\mathbf{z}^H \mathbf{R}_{f,P}^{-1} \mathbf{z} + \sum_{n=1}^{k} e_b^2[n] + \sum_{n=k+P+1}^{N} e_f^2[n] \right] \Bigg)$$

where $\mathbf{z} = \mathbf{x}_{k+1:k+P}$ represents the initial conditioning variables.

Previously, we have made the observation that the conventional forward-backward prediction method can be considered as the averaging of the log-likelihood functions arising from *two* different factorizations of the joint density. As shown in (7), for an AR(P) model, arbitrary P consecutive variables $\mathbf{x}_{k+1:k+P}$ for $k = \{0, \ldots, N-P\}$ can be selected for initial conditioning variables. This leads to a different factorization for each value of $k \in \{0, \ldots, N-P\}$, a grand total ot $N-P+1$ factorizations.

The arithmetic average of log-likelihood functions for all possible conditioning cases can be expressed as

$$\Lambda(\sigma_\epsilon^2, \mathbf{a}) \overset{c}{=} -\Bigg(N \log \sigma_\epsilon^2 + \log |\mathbf{R}_{f,P}| + \sum_{k=0}^{N-P} \frac{\mathbf{z}_k^H \mathbf{R}_{f,P}^{-1} \mathbf{z}_k}{(N-P+1)\sigma_\epsilon^2} + \frac{1}{\sigma_\epsilon^2} \left[\sum_{n=1}^{N-P} w_b[n] |e_b[n]|^2 + \sum_{n=P+1}^{N} w_f[n] |e_f[n]|^2 \right]_{(8)} \Bigg) \quad (8)$$

where $\mathbf{z}_k$ is the vector of variables for the kth conditioning set, $\mathbf{z}_k = \mathbf{x}_{k+1:k+P}$ and $w_b[n] = N-P+1-n$, $w_f[n] = n-P$ are the weights for the forward and backward prediction errors, respectively. To illustrate the weight calculation, x_5 can be forward predicted with a 2nd order predictor, as in Fig. 1, a total of 3 times by conditioning the joint distribution with (x_1, x_2), (x_2, x_3) or (x_3, x_4). This leads to $w_f[5] = 3$. Similarly, x_5 is backward predicted $N-P+1-n|_{\{n=5,P=2\}} = N-6$ times with the conditioning pairs of (x_{N-1}, x_N), (x_{N-2}, x_{N-1}), $\ldots$, (x_6, x_7). This leads to $w_b[5] = N-6$.

It is worth emphasizing that the log-likelihood function given in (8) is identical to the log-likelihood functions given earlier, (5) and (7). Once we optimize over σ_ϵ^2 and insert the estimate for σ_ϵ^2 into (8), we get

$$\Lambda(\widehat{\sigma}_\epsilon^2, \mathbf{a}) \overset{c}{=} -\left\{ N \log \left([\cdot]_{(8)} + \sum_{k=0}^{N-P} \frac{\mathbf{z}_k^H \mathbf{R}_{f,P}^{-1} \mathbf{z}_k}{N-P+1} \right) + \log |\mathbf{R}_f| \right\}.$$

As in conventional forward-backward prediction method, we ignore all the terms except the quadratic term in $\Lambda(\widehat{\sigma}_\epsilon^2, \mathbf{a})$ and minimize $[\cdot]_{(8)}$. The minimization of $[\cdot]_{(8)}$ is simply the weighted average forward-backward prediction errors,

$$\widehat{\mathbf{a}} = \underset{\mathbf{a}}{\operatorname{argmin}} \sum_{n=1}^{N-P} w_b[n] |e_b[n]|^2 + \sum_{n=P+1}^{N} w_f[n] |e_f[n]|^2 \quad (9)$$

where $e_f[n] = x_n + \mathbf{a}^T \mathbf{x}_{n-1:-1:n-P}$, $e_b[n] = x_n + \mathbf{a}^H \mathbf{x}_{n+1:n+P}$ are the forward and backward prediction errors with weights $w_f[n] = n-P$ and $w_b[n] = N-P+1-n$, respectively.

The optimization problem in (9) can be solved simply by introducing the linear equation systems, $\mathbf{A}_f \mathbf{a} = -\mathbf{b}_f$ and $\mathbf{A}_b \mathbf{a} = -\mathbf{b}_b$, given below, generating the forward and backward prediction errors:

$$\underbrace{\begin{bmatrix} x_P & x_{P-1} & \ldots & x_1 \\ x_{P+1} & x_P & \ldots & x_2 \\ \vdots & \vdots & \ldots & \vdots \\ x_{N-1} & x_{N-2} & \ldots & x_{N-P} \end{bmatrix}}_{\mathbf{A}_f} \underbrace{\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_P \end{bmatrix}}_{\mathbf{a}} = -\underbrace{\begin{bmatrix} x_{P+1} \\ x_{P+2} \\ \vdots \\ x_N \end{bmatrix}}_{\mathbf{b}_f},$$

$$\underbrace{\begin{bmatrix} x^*_{N-P+1} & x^*_{N-P+2} & \dots & x^*_N \\ x^*_{N-P} & x^*_{N-P+1} & \dots & x^*_{N-1} \\ \vdots & \vdots & \dots & \vdots \\ x^*_2 & x^*_3 & \dots & x^*_{P+1} \end{bmatrix}}_{\mathbf{A}_b} \underbrace{\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_P \end{bmatrix}}_{\mathbf{a}} = -\underbrace{\begin{bmatrix} x^*_{N-P} \\ x^*_{N-P-1} \\ \vdots \\ x^*_1 \end{bmatrix}}_{\mathbf{b}_b}.$$

Using the introduced matrices, the final result of the first stage becomes

$$\widehat{\mathbf{a}}_{FS} = -(\mathbf{A}_f^H\mathbf{W}\mathbf{A}_f + \mathbf{A}_b^H\mathbf{W}\mathbf{A}_b)^{-1}(\mathbf{A}_f^H\mathbf{W}\mathbf{b}_f + \mathbf{A}_b^H\mathbf{W}\mathbf{b}_b) \tag{10}$$

where $\widehat{\mathbf{a}}_{FS}$ denotes the first stage estimate for $\boldsymbol{a}$, $\mathbf{W}$ is the diagonal matrix with the diagonal entries of $w_f[n]$ for $n = \{P+1, P+2, \dots, N\}$, i.e. $\mathbf{W} = \mathrm{diag}(1, 2, \dots, N-P)$. Notice that, when the weighting matrix $\mathbf{W}$ is replaced with the identity matrix, the method reduces to conventional forward-backward prediction method, [1].

We would like to note that the computational complexity of the first stage is almost identical to the conventional forward-backward prediction scheme. Hence, the performance improvement in using the weighted version comes at no additional implementation cost. A ready-to-use MATLAB code is available in [14].

3.2. Second stage: Maximizing likelihood around a point

The second stage of the proposed method aims to further improve the likelihood value of the first stage estimate. The likelihood function is a highly non-linear function of unknown parameters for third and higher order AR processes. The exact maximum likelihood solution is very difficult to obtain, in general [9,15]. A common approach is to use numerical optimization tools whose computational requirements increase with the desired statistical efficiency. To maximize the likelihood expression locally, the likelihood function is approximated with a quadratic function by the Taylor series expansion of the likelihood function at the point $\mathbf{a} = \widehat{\mathbf{a}}_{FS}$ where $\widehat{\mathbf{a}}_{FS}$ is the estimate of the first stage.

The maximization of the likelihood function given by (3) is equivalent to the minimization of

$$J(\mathbf{a}) = \frac{1}{N}\log|\mathbf{R}_{f,N}| + \log(\mathbf{x}^H\mathbf{R}_{f,N}^{-1}\mathbf{x}). \tag{11}$$

Note that, both terms of the cost function $J(\mathbf{a})$, that is $\log|\mathbf{R}_{f,N}|$ and $\log(\mathbf{x}^H\mathbf{R}_{f,N}^{-1}\mathbf{x})$, are non-linear functions of the unknown vector $\mathbf{a}$. Below, we study the Taylor series of these two functions at $\mathbf{a} = \widehat{\mathbf{a}}_{FS}$, individually. We start with the determinant term.

Quadratic Approximation for $\log|\mathbf{R}_{f,N}|$*:* The log-determinant term, $\log|\mathbf{R}_{f,N}|$, is equal to $\log|\mathbf{R}_{f,P}| = -\sum_{i=1}^P i\log(1-|k_i|^2)$ where k_i is the i'th reflection coefficient corresponding to the filter generating the AR(P) process, [1,6]. Due to the analytical simplicity of the determinant expression in terms of reflection coefficients, we switch the optimization domain to the domain of reflection coefficients.

We can express the first three terms of the Taylor series of the function

$$f(z, z^*) = \log(1-|z|^2) \tag{12}$$

at the expansion point of (z_0, z_0^*) as follows:

$$\begin{aligned} f^a(z_0+\delta_z, z_0^*+\delta_z^*) &= f(z_0, z_0^*) + \begin{bmatrix}\delta_z & \delta_z^*\end{bmatrix}\begin{bmatrix} f_z(z_0,z_0^*) \\ f_{z^*}(z_0,z_0^*)\end{bmatrix} \\ &\quad + \frac{1}{2}\begin{bmatrix}\delta_z & \delta_z^*\end{bmatrix}\begin{bmatrix} f_{zz}(z_0,z_0^*) & f_{zz^*}(z_0,z_0^*) \\ f_{z^*z}(z_0,z_0^*) & f_{z^*z^*}(z_0,z_0^*)\end{bmatrix}\begin{bmatrix}\delta_z \\ \delta_z^*\end{bmatrix} \end{aligned} \tag{13}$$

where

$$f_z(z, z^*) = \frac{\partial}{\partial z}f(z,z^*) = -\frac{z^*}{1-zz^*} \tag{14}$$

and $f_{z^*}(z,z^*) = f_z^*(z,z^*)$. Also,

$$\begin{aligned} f_{zz}(z,z^*) &= \frac{\partial^2}{\partial z^2}f(z,z^*) = -\frac{(z^*)^2}{(1-zz^*)^2}, \\ f_{zz^*}(z,z^*) &= \frac{\partial^2}{\partial z\partial z^*}f(z,z^*) = -\frac{1}{(1-zz^*)^2} \end{aligned} \tag{15}$$

and $f_{z^*z^*}(z,z^*) = f_{zz}^*(z,z^*)$, $f_{z^*z}(z,z^*) = f_{zz^*}^*(z,z^*)$.

Hence, $\frac{1}{N}\log|\mathbf{R}_{f,N}| = \sum_{i=1}^P -\frac{i}{N}\log(1-|k_i|^2)$ can be approximated via Taylor series expansion at the reflection coefficients of k_i $i = \{1, \dots, P\}$ as

$$\frac{1}{N}\log|\mathbf{R}_{f,N}| \approx -\frac{1}{N}\sum_{i=1}^P i f^a(k_i+\delta_{k_i}, k_i^*+\delta_{k_i}^*) \tag{16}$$

where the definition of $f^a(k_i+\delta_{k_i}, k_i^*+\delta_{k_i}^*)$ is given in (13).

For the latter use, we also present the partial derivative of $f^a(z_0+\delta_z, z_0^*+\delta_z^*)$ with respect to δ_z^* as

$$\begin{aligned} f^a_{\delta_z^*}(z_0+\delta_z, z_0^*+\delta_z^*) &= f_{z^*}(z_0,z_0^*) \\ &\quad + \begin{bmatrix} f_{zz^*}(z_0,z_0^*) & f_{z^*z^*}(z_0,z_0^*)\end{bmatrix}\begin{bmatrix}\delta_z \\ \delta_z^*\end{bmatrix} \end{aligned} \tag{17}$$

Hence, the partial derivative of $\frac{1}{N}\log|\mathbf{R}_{f,N}| = \sum_{i=1}^P -\frac{i}{N}\log(1-|k_i|^2)$ with respect to k_i^* can be approximated with $-\frac{i}{N}f^a_{\delta_{k_i^*}}(k_i+\delta_k, k_i^*+\delta_k^*)$, given in (17).

Quadratic Approximation for $\log(\mathbf{x}^H\mathbf{R}_{f,N}^{-1}\mathbf{x})$*:* First, we expand $\mathbf{x}^H\mathbf{R}_{f,N}^{-1}\mathbf{x}$ into Taylor series at the point $\widehat{\mathbf{a}}_{FS}$. From the Gohberg-Semencul formula, $\mathbf{x}^H\mathbf{R}_{f,N}^{-1}\mathbf{x}$ can be expressed as

$$\begin{aligned} \mathbf{x}^H\mathbf{R}_{f,N}^{-1}\mathbf{x} &= \mathbf{x}^H(\mathbf{A}_1\mathbf{A}_1^H - \mathbf{A}_2\mathbf{A}_2^H)\mathbf{x} = \|\mathbf{A}_1^H\mathbf{x}\|^2 - \|\mathbf{A}_2^H\mathbf{x}\|^2 \\ &= \|\mathbf{x}^*_{1:N} + \mathbf{M}_1\mathbf{a}\|^2 - \|\mathbf{M}_2\mathbf{a}\|^2 \end{aligned} \tag{18}$$

where explicit expressions for $\mathbf{A}_1$ and $\mathbf{A}_2$ are given in [16, Eq. (3.9.22), p. 130]. The matrices $\mathbf{M}_1$ and $\mathbf{M}_2$ are Hankel and Toeplitz matrices, respectively, with the definitions of

$$\mathbf{M}_1 = \begin{bmatrix} x_2^* & x_3^* & x_4^* & \dots & x_{P+1}^* \\ x_3^* & x_4^* & x_5^* & \dots & x_{P+2}^* \\ \vdots & \vdots & \vdots & \dots & \vdots \\ x_{N-1}^* & x_N^* & 0 & \dots & 0 \\ x_N^* & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix}_{N\times P},$$

$$\mathbf{M}_2 = \begin{bmatrix} x_N & x_{N-1} & x_{N-2} & \dots & x_{N-P+1} \\ 0 & x_N & x_{N-1} & \dots & x_{N-P+2} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & 0 & \dots & x_{N-1} \\ 0 & 0 & 0 & \dots & x_N \end{bmatrix}_{P\times P}. \tag{19}$$

To facilitate the Taylor series expansion, we substitute $\mathbf{a} = \widehat{\mathbf{a}}_{FS} + \boldsymbol{\delta}_a$ in (18)

$$\mathbf{x}^H\mathbf{R}_{f,N}^{-1}\mathbf{x} = \|\mathbf{x}^*_{1:N} + \mathbf{M}_1(\widehat{\mathbf{a}}_{FS} + \boldsymbol{\delta}_a)\|^2 - \|\mathbf{M}_2(\widehat{\mathbf{a}}_{FS} + \boldsymbol{\delta}_a)\|^2. \tag{20}$$

It should be remembered that we are interested in the minimization of the cost function $J(\boldsymbol{a})$ given in (11) which is composed of two terms. Previously, we have approximated the first term of the sum as a quadratic relation of reflection coefficients. Eq. (20) gives the quadratic approximation of the second term in terms of filter coefficients. In order to combine the approximations to two terms of the cost function, we need to express $\boldsymbol{\delta}_a$ in terms of reflection coefficients.

The first order approximation of $\boldsymbol{\delta}_a$ on the variation of reflection coefficients can be written as $\boldsymbol{\delta}_a = \mathbf{G}\boldsymbol{\delta}_k + \mathbf{G}_c\boldsymbol{\delta}_k^*$ where $\mathbf{G}$ and $\mathbf{G}_c$ are the $P\times P$ Jacobian matrices with the ith row and jth column entry of $\frac{\partial a_i}{\partial k_j}$ and $\frac{\partial a_i}{\partial k_j^*}$ evaluated at the expansion point of $\mathbf{a}_{FS}$, respectively.

Note that the matrices $\mathbf{G}$ and $\mathbf{G}_c$ are *not* complex conjugates of each other. This stems from the fact that the relation between filter coefficients and reflection coefficients are in general complex-valued functions of complex variables, [17].

As an example, the synthesis filter coefficients for an AR(2) process can be written as $a_1 = k_1 + k_1^* k_2$ and $a_2 = k_2$ in terms of the reflection coefficients, [1, p. 234]. The $\mathbf{G}$ and $\mathbf{G}_c$ matrices can be written as

$$\mathbf{G} = \begin{bmatrix} 1 & k_1^* \\ 0 & 1 \end{bmatrix}, \quad \mathbf{G}_c = \begin{bmatrix} k_2 & 0 \\ 0 & 0 \end{bmatrix}.$$

where $[\mathbf{G}]_{ij} = \frac{\partial a_i}{\partial k_j}$ and $[\mathbf{G}_c]_{ij} = \frac{\partial a_i}{\partial k_j^*}$.

The elements of the Jacobian matrices can be efficiently calculated from the inverse Levinson-Durbin recursion. The critical expression of the recursion enabling the calculation is $a_i^{j+1} = a_i^j + k_{j+1}(a_{j-i+1}^j)^*$, $1 \le j \le P$, $1 \le i \le j$, [1, Table 5.4]. Here a_i^j shows the ith filter coefficient at the jth stage of the recursion. This expression states that the nth reflection coefficient (k_n) affects the nth and following stages, i.e. the recursion stages with the index $j \ge n$. Further details of the Jacobian calculation are given in Algorithm 2.

Algorithm 2: Returns Jacobian matrices for the filter coefficients with respect to reflection coefficients evaluated at a given point.

```
1  function [Jr,Ji,G,Gc]=jacobian-of-a-wrt-gamma(gamma);
   Input  : gamma : Reflection coefficients of the expansion
            point
   Output: Jr, Ji : Jacobian matrix of filter coef. wrt
            real/imaginary parts of reflection coef. at the
            expansion point
            G, Gc : Jacobian matrix of filter coef. wrt
            reflection coef and its conjugate at
            the expansion point
2  a=1; p=length(gamma);
3  avecs = cell(1,p);
4  for jind=2:p+1 do
5  |  a=[a;0] + gamma(jind-1)*[0;conj(flipud(a))];
6  |  avecs{jind-1} = a;
7  end
8  Jr = zeros(p+1,p); Ji = zeros(p+1,p);
9  for gammaind=1:p do
10    if gammaind==1, vec = 1;
11    else vec = avecsgammaind-1; end;
12    vecr = [0; conj(flipud(vec))];
13    veci = 1i*vecr;
14    for jind=(gammaind+1):p;
15    vecr=[vecr;0] + gamma(jind)*[0;conj(flipud(vecr))];
16    veci=[veci;0] + gamma(jind)*[0;conj(flipud(veci))];
17    end;
18    Jr(:,gammaind) = vecr;
19    Ji(:,gammaind) = veci;
20 end
21 G = Jr/2 - 1i*Ji/2/1;
22 Gc = Jr/2 + 1i*Ji/2/1;
```

Substituting $\boldsymbol{\delta}_a = \mathbf{G}\boldsymbol{\delta}_k + \mathbf{G}_c\boldsymbol{\delta}_k^*$ into (20), we get

$$\mathbf{x}^H\mathbf{R}_{f,N}^{-1}\mathbf{x} = \|\mathbf{b}_1 + \mathbf{M}_1(\mathbf{G}\boldsymbol{\delta}_k + \mathbf{G}_c\boldsymbol{\delta}_k^*)\|^2 - \|\mathbf{b}_2 + \mathbf{M}_2(\mathbf{G}\boldsymbol{\delta}_k + \mathbf{G}_c\boldsymbol{\delta}_k^*)\|^2, \tag{21}$$

where $\mathbf{b}_1 = \mathbf{x}_{1:N}^* + \mathbf{M}_1\widehat{\mathbf{a}}_{FS}$ and $\mathbf{b}_2 = \mathbf{M}_2\widehat{\mathbf{a}}_{FS}$ are constant vectors. Here $\mathbf{M}_1$ and $\mathbf{M}_2$ matrices given by (19) and $\widehat{\mathbf{a}}_{FS}$ is the estimate of the first stage.

Finally, by the Taylor series expansion of $\log(A + Bx) \approx \log(A) + \frac{B}{A}x$ for $|x| \ll 1$, we can approximate the second term of the cost function as

$$\log \mathbf{x}^H\mathbf{R}_{f,N}^{-1}\mathbf{x} \approx \log(A) + \frac{\|\mathbf{b}_1 + \mathbf{M}_1(\mathbf{G}\boldsymbol{\delta}_k + \mathbf{G}_c\boldsymbol{\delta}_k^*)\|^2 - \|\mathbf{b}_1\|^2}{A} - \frac{\|\mathbf{b}_2 + \mathbf{M}_2(\mathbf{G}\boldsymbol{\delta}_k + \mathbf{G}_c\boldsymbol{\delta}_k^*)\|^2 - \|\mathbf{b}_2\|^2}{A} \tag{22}$$

where $A = \|\mathbf{b}_1\|^2 - \|\mathbf{b}_2\|^2$.

Note that $A = \|\mathbf{b}_1\|^2 - \|\mathbf{b}_2\|^2$ is the value when $\boldsymbol{\delta}_k$ is replaced by all zeros vector in (21). Hence, A can be expressed as $A = \mathbf{x}^H\mathbf{R}_{f,N}^{-1}\mathbf{x}$ for the $\mathbf{R}_{f,N}$ matrix generated by the $\mathbf{a}$ estimate formed from the first stage estimate $\widehat{\mathbf{a}}_{FS}$. The value of A can be efficiently calculated, even without forming $\mathbf{R}_{f,N}$ matrix, via the algorithm given in Algorithm 1. This concludes the Taylor series expansion of the second term at the expansion point.

Combining Quadratic Approximations For Both Terms: Using earlier results, the cost function of $J(\mathbf{a}) = \frac{1}{N}\log|\mathbf{R}_{f,N}| + \log(\mathbf{x}^H\mathbf{R}_{f,N}^{-1}\mathbf{x})$. can be approximated around the operating point of $\mathbf{a} = \widehat{\mathbf{a}}_{FS}$, or equivalently around the reflection coefficient vector corresponding to $\mathbf{a} = \widehat{\mathbf{a}}_{FS}$, as follows

$$J^a(\boldsymbol{\delta}_k, \boldsymbol{\delta}_k^*) = -\frac{1}{N}\sum_{i=1}^{P} if^a(k_i + \delta_{k_i}, k_i^* + \delta_{k_i}^*) - \frac{1}{A}\sum_{i=1}^{2}(-1)^i\|\mathbf{b}_i + \mathbf{M}_i(\mathbf{G}\boldsymbol{\delta}_k + \mathbf{G}_c\boldsymbol{\delta}_k^*)\|^2. \tag{23}$$

Here k_i for $i = \{1, \ldots, P\}$ are the reflection coefficients of the all-pole filter with coefficients $\widehat{\mathbf{a}}_{FS}$. The first and second summation in (23) are the quadratic approximation to $\frac{1}{N}\log|\mathbf{R}_{f,N}|$ and $\log \mathbf{x}^H\mathbf{R}_{f,N}^{-1}\mathbf{x}$, given by (16) and (22), respectively.

Optimizing The Approximation To The Likelihood: The quadratic cost function given by (23) is a real-valued function of the complex-valued vector $\boldsymbol{\delta}_k$; hence the gradient of the quadratic cost function with respect to $\boldsymbol{\delta}_k$ and $\boldsymbol{\delta}_k^*$ are complex conjugates of each other, [17]. Therefore, the calculation of the gradient with respect to $\boldsymbol{\delta}_k^*$ is sufficient for the optimization.

From (17), the gradient of the first summation in (23), with respect to $\boldsymbol{\delta}_k^*$, can be written as

$$\nabla_{\boldsymbol{\delta}_k^*}\left\{-\frac{1}{N}\sum_{i=1}^{P} if^a(k_i + \delta_{k_i}, k_i^* + \delta_{k_i}^*)\right\} = \begin{bmatrix}\mathbf{Q}_1 & \mathbf{Q}_2\end{bmatrix}\begin{bmatrix}\boldsymbol{\delta}_k \\ \boldsymbol{\delta}_k^*\end{bmatrix} + \mathbf{r}_1, \tag{24}$$

where $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are diagonal matrices with diagonal entries of $-i/N \times f_{zz^*}(k_i, k_i^*)$ and $-i/N \times f_{z^*z^*}(k_i, k_i^*)$, $i = \{1, 2, \ldots, P\}$, respectively.

By combining gradient with respect to $\boldsymbol{\delta}_k$ and $\boldsymbol{\delta}_k^*$ together, the gradient with respect to $[\boldsymbol{\delta}_k \quad \boldsymbol{\delta}_k^*]^T$ can be written as

$$\nabla_{[\boldsymbol{\delta}_k \quad \boldsymbol{\delta}_k^*]^T}\left\{-\frac{1}{N}\sum_{i=1}^{P} if^a(k_i + \delta_{k_i}, k_i^* + \delta_{k_i}^*)\right\} = \begin{bmatrix}\mathbf{Q}_1 & \mathbf{Q}_2 \\ \mathbf{Q}_2^* & \mathbf{Q}_1^*\end{bmatrix}\begin{bmatrix}\boldsymbol{\delta}_k \\ \boldsymbol{\delta}_k^*\end{bmatrix} + \begin{bmatrix}\mathbf{r}_1 \\ \mathbf{r}_1^*\end{bmatrix}. \tag{25}$$

Similarly, the gradient of second summation in (23) can be calculated as

$$\nabla_{[\boldsymbol{\delta}_k \quad \boldsymbol{\delta}_k^*]^T}\left\{-\frac{1}{A}\sum_{i=1}^{2}(-1)^i\|\mathbf{b}_i + \mathbf{M}_i(\mathbf{G}\boldsymbol{\delta}_k + \mathbf{G}_c\boldsymbol{\delta}_k^*)\|^2\right\} = \begin{bmatrix}\widetilde{\mathbf{Q}}_1 & \widetilde{\mathbf{Q}}_2 \\ \widetilde{\mathbf{Q}}_2^* & \widetilde{\mathbf{Q}}_1^*\end{bmatrix}\begin{bmatrix}\boldsymbol{\delta}_k \\ \boldsymbol{\delta}_k^*\end{bmatrix} + \begin{bmatrix}\widetilde{\mathbf{r}}_1 \\ \widetilde{\mathbf{r}}_1^*\end{bmatrix}, \tag{26}$$

where

$$\widetilde{\mathbf{Q}}_1 = \frac{\mathbf{G}^H\mathbf{P}\mathbf{G} + \left(\mathbf{G}_c^H\mathbf{P}\mathbf{G}_c\right)^*}{A}; \; \widetilde{\mathbf{Q}}_2 = \frac{\mathbf{G}^H\mathbf{P}\mathbf{G}_c + \left(\mathbf{G}_c^H\mathbf{P}\mathbf{G}\right)^*}{A};$$
$$\widetilde{\mathbf{r}}_1 = \frac{\mathbf{G}^H\mathbf{v} + \left(\mathbf{G}_c^H\mathbf{v}\right)^*}{A}. \tag{27}$$

and $\mathbf{P} = \mathbf{M}_1^H\mathbf{M}_1 - \mathbf{M}_2^H\mathbf{M}_2$, $\mathbf{v} = \mathbf{M}_1^H\mathbf{b}_1 - \mathbf{M}_2^H\mathbf{b}_2$.

Using the results given, the gradient of $J^a(\delta_{k_i}, \delta_{k_i}^*)$ can be expressed by summing the right hand sides of the Eqs. (25) and (26). By equating the gradient to the zero vector, we get the following linear equation system for the optimal perturbation vector that locally maximizes the likelihood around the operating point

$$\begin{bmatrix} \mathbf{Q}_1 + \widetilde{\mathbf{Q}}_1 & \mathbf{Q}_2 + \widetilde{\mathbf{Q}}_2 \\ \mathbf{Q}_2^* + \widetilde{\mathbf{Q}}_2^* & \mathbf{Q}_1^* + \widetilde{\mathbf{Q}}_1^* \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta}_k \\ \boldsymbol{\delta}_k^* \end{bmatrix} = - \begin{bmatrix} \mathbf{r}_1 + \widetilde{\mathbf{r}}_1 \\ \mathbf{r}_1^* + \widetilde{\mathbf{r}}_1^* \end{bmatrix}. \tag{28}$$

The end result of the second stage is the updated reflection coefficients from k_i to $k_i + \delta_{k_i}$ where k_i is the reflection coefficients corresponding to the estimate generated by the first stage. By running the well known step-up recursion or inverse Levinson recursion, we can convert the optimized reflection coefficients to the filter coefficients, [1]. For further details on the second stage of the proposed method, you can examine the ready-to-use MATLAB code given in [14].

We would like to underline that the implementation of the second stage can be simplified for real-valued processes. For real-valued processes, the reflection coefficients are also real-valued, i.e. $\boldsymbol{\delta}_k = \boldsymbol{\delta}_k^*$. Substituting this condition in (28) results in a lower dimensional matrix equation from which the optimal perturbation vector can be found at the halved dimension of complex-valued case:

$$\left(\mathbf{Q}_1 + \mathbf{Q}_2 + \widetilde{\mathbf{Q}}_1 + \widetilde{\mathbf{Q}}_2\right)\boldsymbol{\delta}_k = -(\mathbf{r}_1 + \widetilde{\mathbf{r}}_1). \tag{29}$$

As a final note, we would like to point the possibility of running the proposed algorithm iteratively. That is, the second stage result of an earlier iteration can be taken as the initial condition of the next iteration. In the numerical results section, the proposed method and its iterative version are compared with other methods to illustrate the cases in which the iterations can be useful.

4. Numerical results

We present a comparison of the likelihood values attained by several autoregressive model parameter estimation methods. The comparisons are given in two sets. In the first set, the experiments are conducted on AR processes with specific synthesis filters that have been previously utilized in similar performance comparisons, [6]. In the second set, the likelihood value comparison is given for AR processes with randomly selected filter coefficients.

4.1. First comparison set

The comparisons are limited to AR(1), AR(2) and AR(4) processes in this set. We would like to remind that, with the exception of AR(1) and AR(2) processes, *exact* maximum likelihood parameter estimation for AR processes is known to be infeasible to implement, [9,15]. In the cases examined in this set, the poles of the synthesis filter are close to the unit circle. This choice increases the value of the determinant term in the compressed likelihood expression in (3) and puts the methods ignoring this term at a disadvantage in comparison to true maximum likelihood estimator. Due to the complexity of the maximum likelihood estimator, all practical methods, including the proposed one, exhibit different degrees of sensitivities to the ignored terms. Our goal is to compare Burg's method, conventional forward-backward prediction method (also known as modified covariance method), covariance method (also known as forward prediction method), correlation method (also known as Yule-Walker method) and Kay's RMLE method [6] with the suggested method.

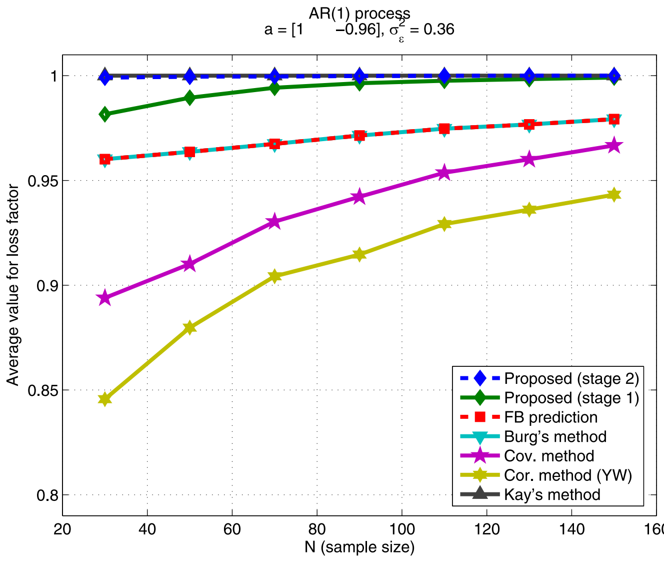


Fig. 2. Variation of the average loss factor with respect to N (sample size) for AR(1) process.

Case 1: AR(1) Process The parameters generating the AR(1) process are selected as $\mathbf{a} = [1 \quad -0.96]^T$ and $\sigma_\epsilon^2 = 0.36$. The synthesis filter is excited with white Gaussian noise and N consecutive samples of the filter output is taken as the observation vector. Specifically for the AR(1) process, the methods proposed by Kay [6] and Whorter et al. [9] generate the exact maximum likelihood estimate. All other methods generate an approximation to the true maximum likelihood value.

As a comparison metric, the ratio of attained likelihood value by a method over the maximum likelihood value is taken. We denote this ratio as the loss factor. Fig. 2 shows the average of loss factor as the observation vector length (N), i.e. the number of observations, varies. The average is calculated over 5000 Monte Carlo runs in all comparisons.

From Fig. 2, we can see that the second stage result of the proposed method is virtually identical to the maximum likelihood estimator of Kay and Whorter et al. In addition, the first stage output of the proposed method, (the initial condition of the second stage) is also a good estimate on its own, especially as N increases. For AR(1) processes, Burg's method and forward-backward prediction method present identical results, in line with the theoretical expectations.

As a side observation, the conventional wisdom on the goodness of practical methods for AR parameter estimation, i.e. forward-backward prediction followed by Burg's method which is followed by covariance and correlation methods, is exactly reflected in Fig. 2. More importantly, the proposed weighted forward-backward prediction gives much better results in terms of likelihood maximization than all other schemes that are computationally competitive.

Case 2: AR(2) Process In this case, the poles of the synthesis filter are located at $0.96\exp(j\pi/4)$ and its conjugate on the complex plane. Among all methods, the method by Whorter at al. gives the true maximum likelihood estimate for AR(2) processes [9]. Other methods are approximations to the maximum likelihood estimator.

From Fig. 3, we can observe that the results for the second stage of the proposed method coincides with the maximum like-

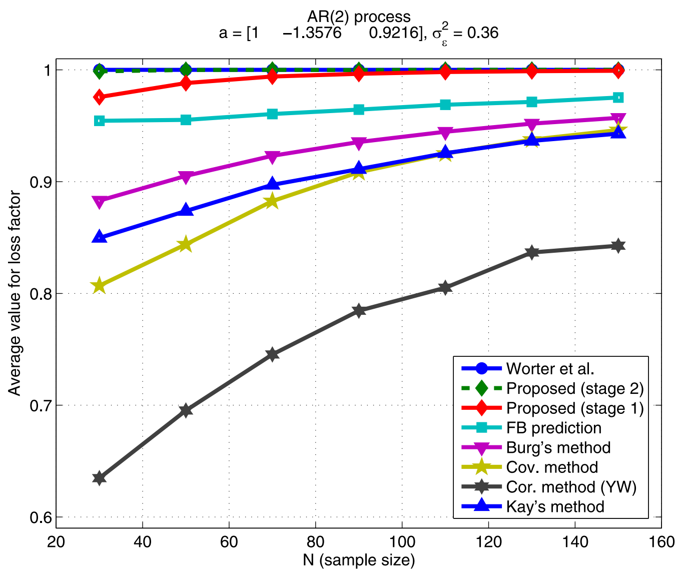


Fig. 3. Variation of the average loss factor with respect to N (sample size) for AR(2) process.

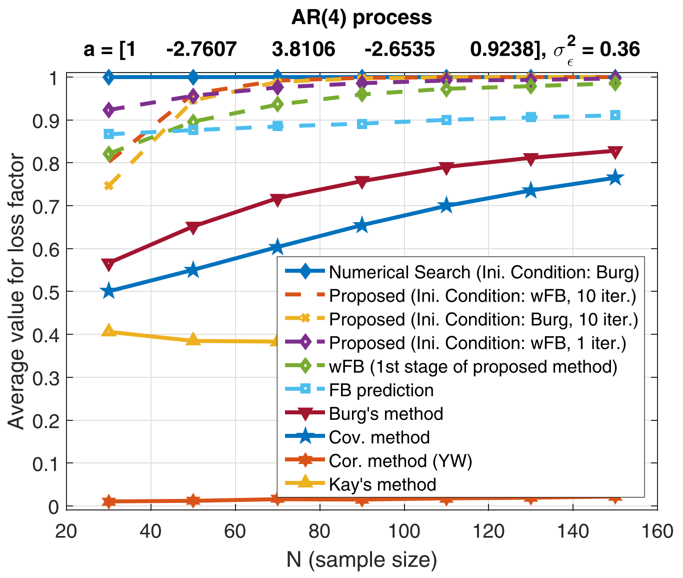


Fig. 4. Variation of the average loss factor with respect to N (sample size) for AR(4) process.

lihood estimator of Whorter at al. for all N values. Similar to the AR(1) case, the first stage of proposed method (weighted forward-backward prediction) performs best among all linear prediction based methods in the literature. It is interesting to note that the gap between conventional forward-backward prediction method and its weighted version is not closed even at large samples sizes. The performance gaps between the weighted forward-backward prediction method and other two methods (Burg's method and covariance method) are even larger.

Case 3: AR(4) Process The parameters of the synthesis filter generating AR(4) process are $\mathbf{a} = [1 \quad -2.7607 \quad 3.8106 \quad -2.6535 \quad 0.9238]^T$ and $\sigma_\epsilon^2 = 0.36$. The random process generated by this filter is utilized as a benchmark for parameter estimation in many studies since the initial study of Kay, [6]. Since the exact maximum likelihood value is not exactly known for this case, we have implemented a Quasi-Newton method based numerical search for the likelihood maximization. The numerical search method uses the estimate of Burg's method as the initial condition. For this comparison, the loss factor is calculated by normalizing the likelihood values by the maximum of the attained values for that Monte Carlo run.

The dashed and solid lines in Fig. 4 show the results of proposed method and other methods, respectively. Different from earlier comparisons, the second stage of the proposed method is also initialized with the estimate of Burg's method. In addition, the performance after 10 iterations of the second stage is also given.

From Fig. 4, we can see that among all conventional methods that are computationally competitive, both stages of the method (shown by dashed lines with different colors) yields the best performance. A careful consideration of Fig. 4 also yields that the initial condition for the second stage can also be taken as the Burg's estimate with a minor degradation in the loss factor. The results of this comparison may also imply that running the proposed method for 10 iterations has a little return in comparison to running it only once. Even though this comment is correct for this case, the conclusion is reversed for higher order AR processes that are studied in the second comparison set. Again the weighted forward-backward prediction is the best method among all linear prediction based methods. It is also interesting to note that the correlation method (Yule-Walker method) performs very poorly in this case, which is known to have serious problems for ill-conditioned covariance matrices [18].

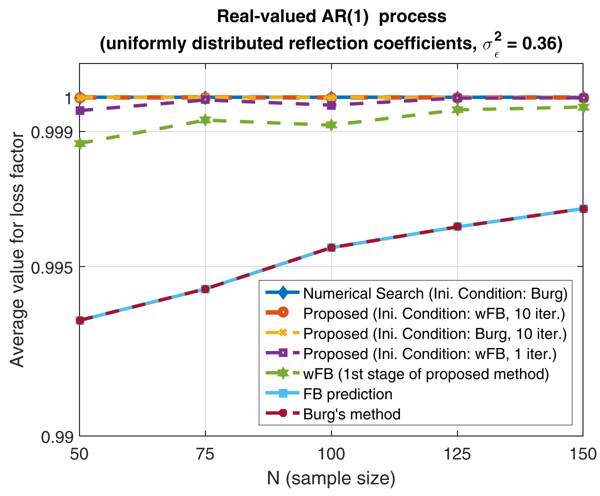
4.2. Second comparison set

In this comparison set, we study AR processes whose synthesis filters have randomly selected reflection coefficients. The randomization of the filters is achieved by independently sampling each reflection coefficient from the uniform distribution over the unit circle in the complex plane, i.e. the stability region of the filter. Since the comparisons are not specific to a synthesis filter, results represent the average over the ensemble of stable filters of a given order. For clarity, we do not present the results for the correlation, covariance and Kay's method which perform poorly in comparison to other methods.

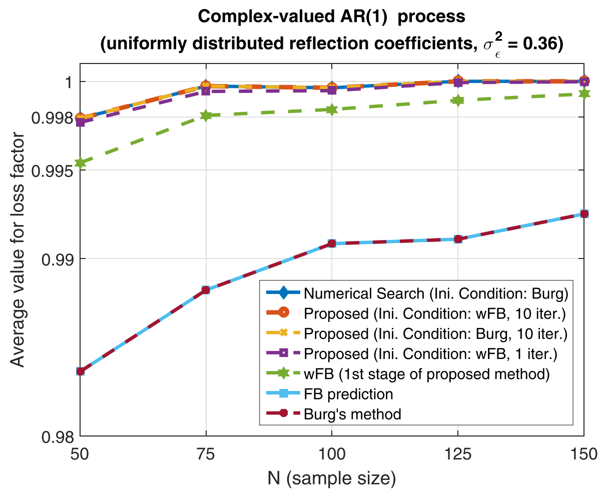
Case 1: AR(1), AR(2) Processes Fig. 5 shows the average loss factor comparisons for real/complex valued AR(1) and AR(2) processes. For real-valued processes, the reflection coefficients of the synthesis filter are sampled from the uniform distribution in $(-1, 1)$ interval, leading to a stable filter with real-valued coefficients. The filter is excited with real-valued white Gaussian noise to generate Gaussian AR process. Finally, N consecutive filter output samples are concatenated to form the observation vector.

From Fig. 5, it can be seen that the proposed method (dashed lines) presents results very close to the numerical search in this case. A single iteration of the second stage is sufficient to practically reach the likelihood value of the numerical search. Again, the first stage of the proposed method (green dashed line) yields much better results than the conventional prediction based methods, consistent with earlier results.

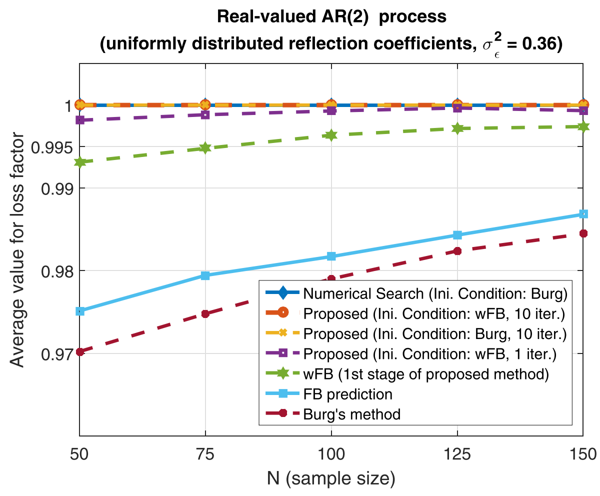
Case 2: Higher Order AR Processes Fig. 6 shows the average loss factor comparisons for AR(4), AR(6), AR(8) and AR(10) processes. The results for the proposed method is very close to the numerical search in all cases, except the case of AR(10) for small sample sizes. In addition, different from the earlier comparisons, the iterative application of the proposed method for 10 iterations yields a significant improvement in the performance. This is essentially due to increasingly rugged nature of the objective function with the model order increase. As the sample size N increases, the performance of all methods improve. The rate of improvement is small for Burg's and conventional forward-backward prediction methods. The second stage of the suggested method becomes an important tool to improve the likelihood value at a much lower computational cost than the numerical search operation. Again, the first stage of



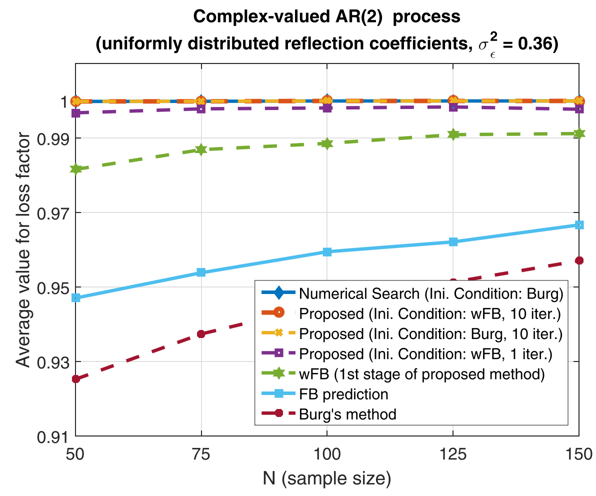
(a) Real-valued AR(1) process



(b) Complex-valued AR(1) process



(c) Real-valued AR(2) process



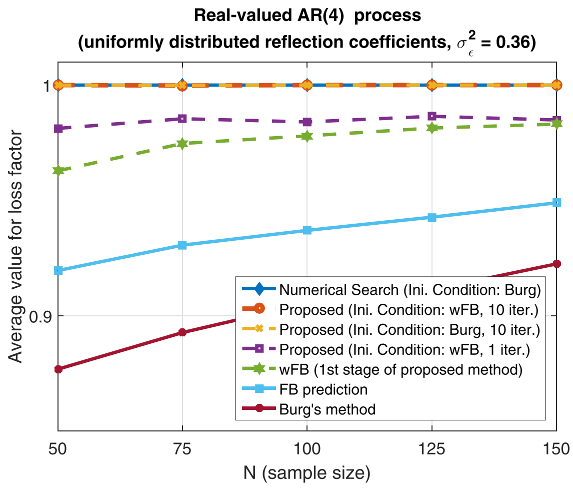
(d) Complex-valued AR(2) process

Fig. 5. Average loss factor for AR(1) and AR(2) processes whose synthesis filters have randomized reflection coefficients.

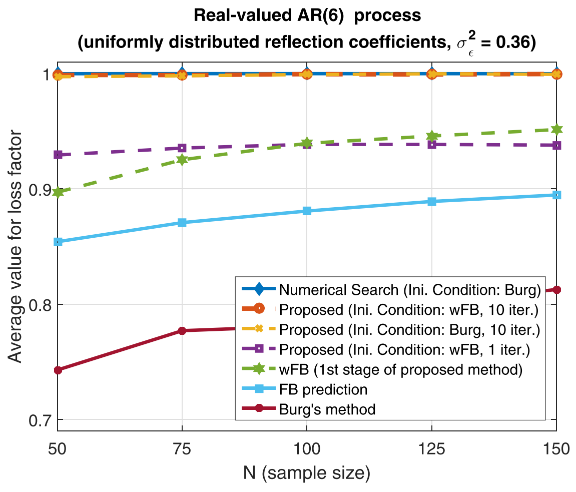
the proposed method, the weighted forward-backward prediction method, yields better results than other linear prediction based methods in all cases.

Computational Complexity Considerations: Fig. 7 shows the performance of the proposed method for different numbers of second stage iterations. This experiment, apart from fixing the iteration number, is identical to the one given for AR(8) and AR(10) processes in Fig. 6; but different from earlier comparisons, the numerical search is also initialized with the first stage output (weighted forward-backward prediction method) to observe the effectiveness of the proposed method. The solid black line shows the result of the numerical search without any limitations on the iteration number. From Fig. 7, we observe that running the second stage for 5 iterations is sufficient to get a good approximation to the final result of the numerical search. Surprisingly, increasing the number of iterations to 10, causes a minor reduction in the likelihood value due to previously mentioned rugged optimization space for high ordered AR processes and also due to the lack of positive definiteness guarantee on the matrix to be inverted in the second stage of the proposed method. Typically, the numerical search methods implement checks to avoid the reduction in the objective function value at each iteration. Similar early-termination checks can also be utilized in the second-stage of the proposed method. An important observation from Fig. 7 is that 5 iterations of the suggested method yields a better performance than 10 iterations of the numerical search method. Furthermore, to achieve the performance of the proposed method (with 5 iterations) more than 20 and 30 iterations of Quasi-Newton based numerical search are required on the average for AR(8) and AR(10) processes, respectively. Hence, we observe that the numerical search requires approximately 5 fold number of iterations in comparison to the proposed method to attain a similar performance in this experiment.

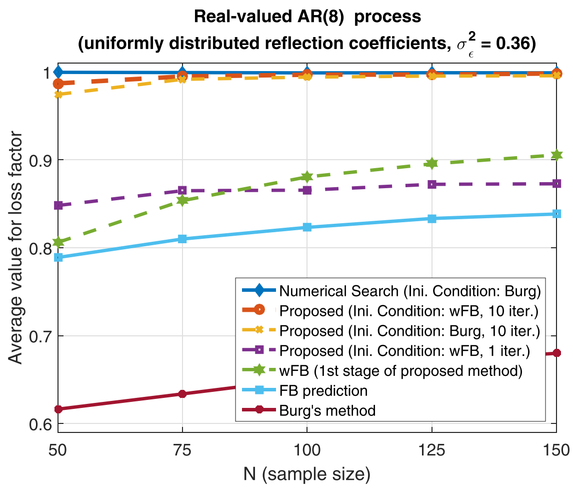
To further compare the computational load of the proposed method with the numerical search, we have used the compressed likelihood function given in (3) as the objective function of the numerical search routine and utilized the efficient implementation of the quadratic form $\mathbf{x}^H\mathbf{R}_{f,N}^{-1}\mathbf{x}$ (given in the Algorithm 1 listing) and log-determinant $(\log|\mathbf{R}_{f,N}| = -\sum_{i=1}^{P} i\log(1-|k_i|^2)$ evaluation via the step-down recursion [1, p.236]. With the provided efficient implementations, the computational complexity for the evaluation of objective function is dominated by the evaluation complexity of the quadratic form, which is on the order of $N\times P$ multiplications. The numerical search needs the evaluation of the objective function P times for the estimation of the gradient vector, a Hessian update and the inversion of a $P\times P$ matrix at each iteration. In com-



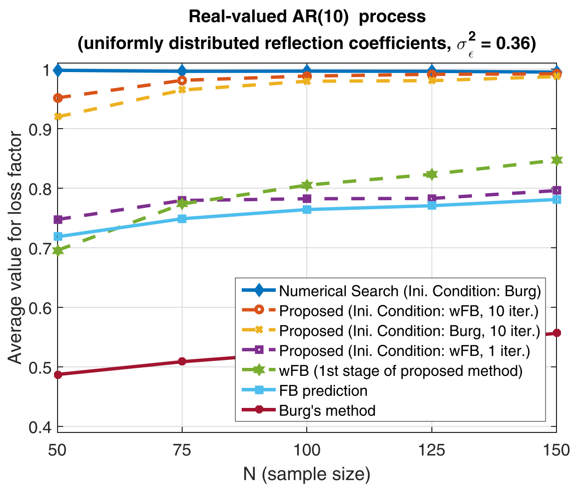
(a) AR(4) process



(b) AR(6) process

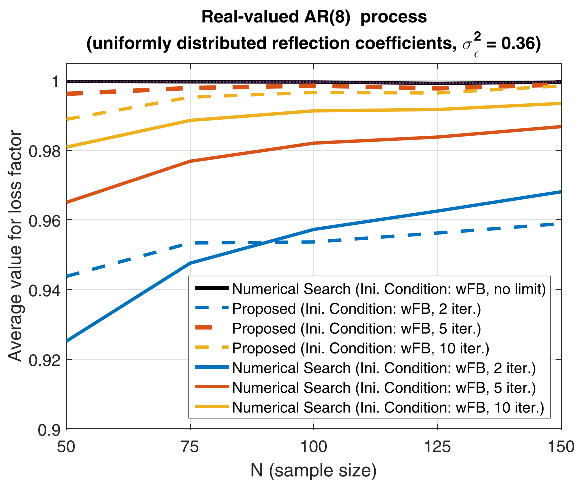


(c) AR(8) process

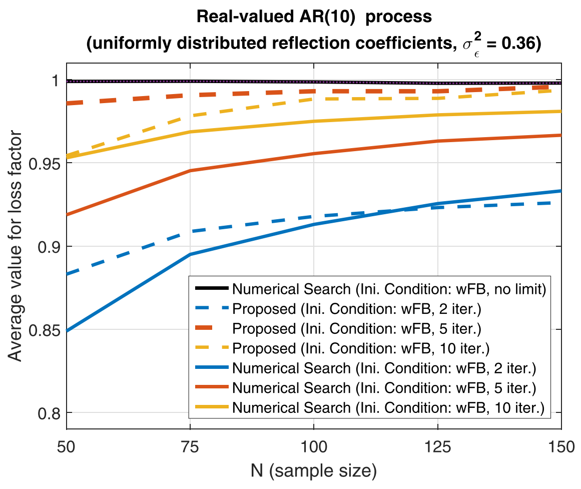


(d) AR(10) process

Fig. 6. Average loss factor for real-valued AR processes of different orders. The filter reflection coefficients are sampled from uniform distribution in $(-1, 1)$ at each Monte Carlo run.



(a) AR(8) process



(b) AR(10) process

Fig. 7. Average loss factor for AR(8) and AR(10) processes for different numbers of iterations.

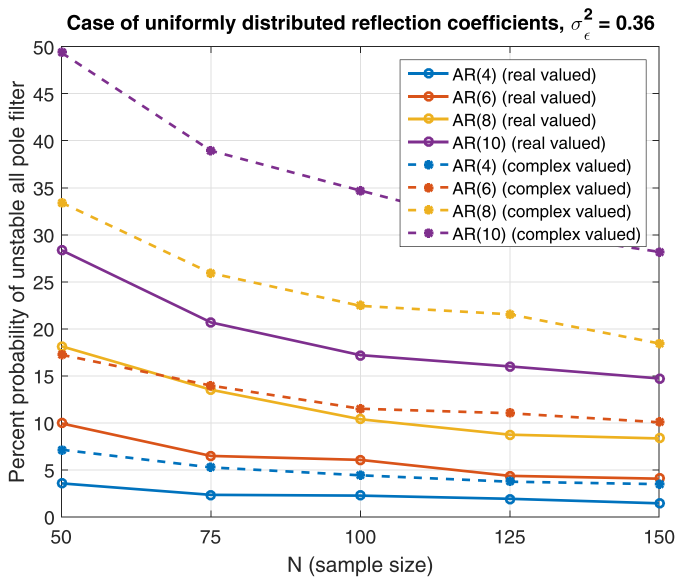


Fig. 8. Probability of having an unstable synthesis filter for the experiments in Fig. 6 and its complex valued version when the proposed method is initialized with the weighted forward-backward prediction method.

parison, the proposed method basically requires the inversion of a $P \times P$ matrix at each iteration. It has been observed that total CPU time of the numerical search is 2–3 times of the proposed method at each iteration. Considering the number of iterations, the overall computation load of the proposed method is 10–15 folds reduced in comparison to the numerical search in this experiment. We have noted a similar CPU time reduction, about an order of magnitude, in other experiments.

Stability Considerations: A major set-back for the first stage of the proposed method is the lack of stability guarantee for the designed synthesis filter. It is well known that all-pole filter designs by the correlation method and Burg's method are guaranteed to be stable [1]. Yet, all other methods including the covariance method, forward-backward prediction method and the suggested method, which is a variant of forward-backward prediction, do not have a stability guarantee. Fig. 8 shows the percentage of unstable synthesis filters for the weighted forward-backward prediction method for the experiment in Fig. 6.

Given the lack of stability guarantee for the weighted forward-backward prediction method, we suggest to initialize the second stage with the weighted forward-backward prediction estimate only if the initial estimate corresponds to a stable filter. If the weighted forward-backward prediction estimate results in an unstable filter, the second stage can be initialized with the Burg's estimate. For high ordered processes, it is also recommended to run the second stage for a small number of iterations, say 5 to 10 iterations.

5. Discussions and conclusions

A low complexity parameter estimation method for the maximum likelihood estimation of AR(P) process parameters is given. The method consists of two stages. The first stage of the method is the weighted version of the conventional forward-backward prediction scheme. The weighted version comes no additional implementation cost and gives better likelihood values in comparison to other classical methods such as correlation method, covariance method, Burg's method etc. Different from similar efforts, the proposed weights are not given in an ad-hoc manner, as in [19,20]; but derived from the likelihood metric partially explaining the observed improvement. Unfortunately, the weighted forward-backward prediction suffers from the stability problem at small sample sizes which is a common problem for some other linear prediction methods in this class such as covariance, conventional forward-backward prediction methods etc.

The second stage of the proposed method approximates the likelihood function around an initial parameter estimate with a quadratic function. By solving the linear equation system associated with the optimization of quadratic approximation, the final estimate is generated. The conducted numerical results show that the second stage result is almost as good as Quasi-Newton based numerical search operation with a proper initialization. We recommend to initialize the second stage with the estimate produced by the weighted forward-backward prediction scheme due to its superior performance, in terms of likelihood value, in comparison to all other methods. If the estimate produced by the weighted forward-backward scheme results in an unstable filter, the estimate via Burg's method can also be used. Depending on the AR model order, the second stage is recommended to run a few number of iterations, say 5–10 iterations.

In many applications, the process to be modeled is observed in the presence of additive white noise, further complicating the AR modeling problem [21]. For these problems, the proposed method can be used as a sub-component of the expectation maximization (EM) or Alternating Direction Method of Multipliers (ADMM) frameworks taking into account the deviation from the exact AR model in parameter estimation. As a final note, a ready-to-use MATLAB codes of the suggested method is made available in [14].

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] M.H. Hayes, Statistical Digital Signal Processing and Modeling, Wiley, 1996.
[2] M. Leonard, B. Kennett, Multi-component autoregressive techniques for the analysis of seismograms, Elsevier Phys. Earth Planetary Inter. 113 (1) (1999) 247–263.
[3] P. Wang, H. Li, B. Himed, A bayesian parametric test for multichannel adaptive signal detection in nonhomogeneous environments, IEEE Signal Process. Lett. 17 (4) (2010) 351–354.
[4] A. Schlögl, A comparison of multivariate autoregressive estimators, Signal Process. 86 (9) (2006) 2426–2429.
[5] P.M.T. Broersen, Automatic Autocorrelation and Spectral Analysis, Springer-Verlag, 2006.
[6] S. Kay, Recursive maximum likelihood estimation of autoregressive processes, IEEE Trans. Acoust. Speech Signal Process 31 (1) (1983) 56–65.
[7] M.L. Vis, L.L. Scharf, A note on recursive maximum likelihood for autoregressive modeling, IEEE Trans. Signal Process 42 (10) (1994) 2881–2883.
[8] P.D. Tuan, Maximum likelihood estimation of the autoregressive model by relaxation on the reflection coefficients, IEEE Trans. Acoust. Speech Signal Process 36 (8) (1988) 1363–1367, doi:10.1109/29.1667.
[9] L.T. Mc Whorter, L.L. Scharf, Nonlinear maximum likelihood estimation of autoregressive time series, IEEE Trans. Signal Process 43 (12) (1995) 2909–2919, doi:10.1109/78.476434.
[10] K.J. Sohn, H. Li, B. Himed, Parametric GLRT for multichannel adaptive signal detection, IEEE Trans. Signal Process 55 (11) (2007) 5351–5360, doi:10.1109/TSP.2007.896068.
[11] A. El-Jaroudi, J. Makhoul, Discrete all-pole modeling, IEEE Trans. Signal Process 39 (2) (1991) 411–423, doi:10.1109/78.80824.
[12] A.M. Sykulski, S.C. Olhede, A.P. Guillaumin, J.M. Lilly, J.J. Early, The debiased whittle likelihood, Biometrika 106 (2) (2019) 251–266, doi:10.1093/biomet/asy071.
[13] D.W. Tufts, R. Kumaresan, Estimation of frequencies of multiple sinusoids: making linear prediction perform like maximum likelihood, Proc. IEEE 70 (9) (1982) 975–989.
[14] C. Candan, Making linear prediction perform like maximum likelihood in Gaussian autoregressive model parameter estimation (MATLAB code), 2019. https://codeocean.com/capsule/2779767/.

[15] X. Zhou, S.M. Kay, A comparison between robust information theoretic estimator and asymptotic maximum likelihood estimator for misspecified model, Proceedings of the SPIE 10646 (2018) 106460N 1–11. doi:10.1117/12.2304550.
[16] P. Stoica, R. Moses, Spectral Analysis of Signals, Prentice Hall, 2005.
[17] C. Candan, Properly handling complex differentiation in optimization and approximation problems, IEEE Signal Process. Mag 36 (2) (2019) 117–124.
[18] M. De-Hoon, T.V.d. Hagen, H. Schoone-welle, H. Van-Dam, Why Yule-Walker should not be used for autoregressive modeling, Ann. Nucl. Energy 23 (15) (1996) 1219–1228.
[19] M. Kaveh, G. Lippert, An optimum tapered burg algorithm for linear prediction and spectral analysis, IEEE Trans. Acoust., Speech, and Signal Process 31 (2) (1983) 438–444, doi:10.1109/TASSP.1983.1164082.
[20] W. Campbell, D.N. Swingler, Frequency estimation performance of several weighted Burg algorithms, IEEE Trans. Signal Process 41 (3) (1993) 1237–1247.
[21] L. Weruaga, L. Dimitrov, The spectral nature of maximum likelihood noise compensated linear prediction, IEEE Audio Speech Lang. Process 21 (8) (2013) 1760–1765.