Proceedings of the 41st IEEE
Conference on Decision and Control
Las Vegas, Nevada USA, December 2002

An Efficient Implementation of Maximum Likelihood Identification of LTI State-Space Models by Local Gradient Search

N. H. Bergboer*), V. Verdult**), M. H. G. Verhaegen*)

*) Faculty of Applied Physics, University of Twente
P.O. Box 217, NL-7500 AE Enschede, The Netherlands
**) Faculty of Information Technology and Systems, Delft University of Technology
P.O. Box 5031, NL-2600 GA Delft, The Netherlands
Fax: (+31) 15 27 86679, Email: V.Verdult@its.tudelft.nl

Abstract

We present a numerically efficient implementation of the nonlinear least squares and maximum likelihood identification of multivariable linear time-invariant (LTI) state-space models. This implementation is based on a local parameterization of the system and a gradient search in the resulting parameter space. The output error identification problem is discussed, and its extension to maximum likelihood identification is explained. We show that the maximum likelihood framework yields parameter errors that converge to the Cramer-Rao bound. Furthermore, the implementation is shown to be fast and able to handle large sample size problems.

1 Introduction

In this paper, we present an efficient implementation of the least squares and maximum likelihood identification of linear time-invariant (LTI) state-space systems, which have the following structure:

$$x(k+1) = \underline{A}x(k) + \underline{B}u(k), \quad (1)$$

$$y(k) = \underline{C}x(k) + \underline{D}u(k) + v(k). \quad (2)$$

The system's unknown state is represented by $x(k) \in \mathbb{R}^n$, $u(k) \in \mathbb{R}^m$ is the input, $y(k) \in \mathbb{R}^\ell$ is the output and $v(k) \in \mathbb{R}^\ell$ is an unknown perturbation of the output measurements that is independent of $u(k)$. A standing assumption throughout this paper is that the system is stable, so that a bounded $u(k)$ results in a bounded $y(k)$. Given N input and output samples, the goal is to determine the system matrices $\underline{A}$, $\underline{B}$, $\underline{C}$ and $\underline{D}$. This identification problem is solved by using a local parameterization of the LTI state-space system, and by performing a gradient search in the resulting parameter space. This method is a special case of the one presented by Lee and Poolla [3].

The maximum likelihood identification of LTI models is an important tool that is used extensively [4, Sec. 7.4]. The implementation can be based on either predefined model structures such as ARX or Box-Jenkins, or on fully parameterized state-space models in which a projection is used to compensate for the over-parameterization [3, 5, 11]. In this paper we will use the latter, both because of numerical robustness [5], and because it removes the need for choosing a canonical model structure. However, it should be stressed that the method we develop can equally well be applied to predefined model structures.

It is our opinion that numerical implementation issues concerning identification have received less attention in literature than they deserve. Efficient large-scale identification packages are of considerable interest to both research groups and the industry to solve practical problems.

The main contribution of this paper is the description of a computationally efficient and numerically accurate implementation of maximum likelihood identification of LTI state-space systems. The method is based on a weighted nonlinear least squares optimization in which the weighting matrix is based on the characteristics of the noise. The optimization problem is solved by a local gradient search. This paper addresses both the efficient computation of the weighting matrix and the efficient computation of the gradients and the projection needed in the local gradient search.

The maximum likelihood framework discussed in this paper is applicable to any system which can be suitably parameterized. In this paper, we will consider only LTI systems, but many of the numerical efficiency issues are applicable to linear parameter-varying (LPV) and bilinear systems [9], [10] as well.

This paper is organized as follows: Section 2 describes the output error identification problem and Section 3 describes the numerical efficiency issues for this problem. Here, we effectively assume a white $v(k)$. If $v(k)$ is nonwhite, a maximum likelihood optimization is required, which is discussed in Section 4 together with implementation details. Section 5 provides some simulation results.

2 Output Error Identification

Given N samples of the input $u(k)$ and the output $y(k)$ of the real system (1)–(2), we wish to identify this system. To this end, we parameterize matrices $A(\theta)$, $B(\theta)$, $C(\theta)$ and

0-7803-7516-5/02/$17.00 ©2002 IEEE

$D(\theta)$ so that the output of the model

$$\widehat{x}(k+1;\theta) = A(\theta)\widehat{x}(k;\theta) + B(\theta)u(k), \quad (3)$$
$$\widehat{y}(k;\theta) = C(\theta)\widehat{x}(k;\theta) + D(\theta)u(k), \quad (4)$$

matches the output of the real system sufficiently well. This is achieved by minimizing the output-error cost function with respect to the parameters θ:

$$V_N(\theta) \equiv \frac{1}{N}\sum_{k=1}^{N} \|y(k) - \widehat{y}(k;\theta)\|_2^2 = \frac{1}{N} E_N(\theta)^T E_N(\theta). \quad (5)$$

Here, $E_N(\theta) = [e(1)^T\ e(2)^T\ \cdots\ e(N)^T]^T \in \mathbb{R}^{N\ell}$ denotes the error vector in which $e(k) := y(k) - \widehat{y}(k;\theta)$. The model matrices A, B, C and D are fully parameterized, that is, the parameter vector θ is given by

$$\theta := \mathrm{vec}\left(\begin{bmatrix} A & B \\ C & D \end{bmatrix}\right). \quad (6)$$

This parameterization implies a nonlinear relation between θ and $\widehat{y}(\theta)$, which results in a nonlinear and nonconvex optimization problem. A good initial estimate is of great importance for such an optimization problem, and can be obtained using LTI subspace identification (e.g. PI-MOESP [12]).

2.1 Local Gradient Search

There does not exist a unique point θ^* for which the cost function (5) has a global minimum; the input-output behavior of LTI state-space models is invariant under a nonsingular similarity transformation $T \in \mathbb{R}^{n\times n}$ of the state basis. This nonuniqueness can be dealt with by deriving, at each point θ, a set of only those directions in which the cost function changes. This set is given by the left null space of the following matrix [3]:

$$M(\theta) = \begin{bmatrix} I_n \\ 0_{m\times n} \end{bmatrix} \otimes \begin{bmatrix} A \\ C \end{bmatrix} - \begin{bmatrix} A^T \\ B^T \end{bmatrix} \otimes \begin{bmatrix} I_n \\ 0_{\ell\times n} \end{bmatrix}. \quad (7)$$

Here, $\otimes$ denotes the Kronecker matrix product. The left null space of a matrix is usually obtained using a singular value decomposition. However, since $M(\theta)$ is of full rank for minimal systems [3], we can use a QR factorization, which is computationally faster:

$$M(\theta) = \begin{bmatrix} \mathcal{Q}_1(\theta) & \mathcal{Q}_2(\theta) \end{bmatrix} \begin{bmatrix} \mathcal{R}_1(\theta) \\ 0 \end{bmatrix}. \quad (8)$$

The columns of the matrix $\mathcal{Q}_2$ form an orthonormal basis for the left null space of $M(\theta)$ [2, p. 229]. A gradient search will be performed in this resulting local parameter subspace.

There exists a multitude of gradient search algorithms, but most implementations are incapable of handling local parameterizations and effectively use $\mathcal{Q}_2 = I$. Therefore, we will use a dedicated trust region implementation of the Levenberg-Marquardt algorithm [6]. After integrating the local parameterization into this algorithm, subsequent iterations update the parameters as $\theta^{(i+1)} = \theta^{(i)} + d(\theta^{(i)}, \lambda^{(i)})$, in which $d(\theta^{(i)}, \lambda^{(i)})$ is given by

$$d(\theta,\lambda) = -\mathcal{Q}_2(\theta)\Phi(\theta,\lambda)^{-1}\mathcal{Q}_2(\theta)^T\Psi_N(\theta)^T E_N(\theta), \quad (9)$$

with

$$\Phi(\theta,\lambda) := \mathcal{Q}_2(\theta)^T\Psi_N(\theta)^T\Psi_N(\theta)\mathcal{Q}_2(\theta) + \lambda I. \quad (10)$$

The Levenberg-Marquardt regularization parameter $\lambda^{(i)}$ is determined in each iteration and depends on the linearity of the cost function in the vicinity of the point $\theta^{(i)}$. The Jacobian of the error vector $E_N(\theta)$ is given by:

$$\Psi_N(\theta) := \frac{\partial E_N(\theta)}{\partial \theta^T}. \quad (11)$$

2.2 Calculating the Iterative Parameter Update

In the previous section the calculation of $\mathcal{Q}_2(\theta)$ for the parameter update rule (9) was discussed. In order to calculate the search direction, we require $E_N(\theta)$ and $\Psi_N(\theta)$ as well. Calculating $E_N(\theta)$ is done by simulating the LTI system (3)–(4) corresponding to $\theta^{(i)}$, which yields $\widehat{y}(k;\theta^{(i)})$. It also yields the state sequence $\widehat{x}(k;\theta^{(i)})$, which can be used to simulate $\Psi_N(\theta)$ efficiently [3, 11], using the related LTI state-space system

$$\mathcal{X}_i(k+1;\theta) = A\mathcal{X}_i(k;\theta) + \frac{\partial A}{\partial \theta_i}\widehat{x}(k;\theta) + \frac{\partial B}{\partial \theta_i}u(k), \quad (12)$$
$$\frac{\partial \widehat{y}(k;\theta)}{\partial \theta_i} = C\mathcal{X}_i(k;\theta) + \frac{\partial C}{\partial \theta_i}\widehat{x}(k;\theta) + \frac{\partial D}{\partial \theta_i}u(k), \quad (13)$$

with

$$\mathcal{X}_i(k;\theta) := \frac{\partial \widehat{x}(k;\theta)}{\partial \theta_i}. \quad (14)$$

3 Efficient Implementation

In section 2 we described the output error optimization algorithm. In this section, we will discuss the numerical efficiency issues associated with this algorithm.

3.1 Calculating the Search Direction

It is well-known that computing the search direction using (9) is inefficient and inaccurate. The QR factorization can be used to avoid the formation and direct inversion of (10). The Levenberg-Marquardt algorithm needs to compute several candidate search directions —corresponding to different values of λ— in each iteration i; this would require several QR factorizations. However, only one QR factorization per iteration suffices if we compute the following economy size QR factorization:

$$\begin{bmatrix} \Psi_N\mathcal{Q}_2 & E_N \end{bmatrix} = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix}. \quad (15)$$

Only the R factor is needed, so that the Q factor does not need to be calculated and stored, which significantly reduces the number of calculations [2, p. 225].

The size of the matrix R_{11} does not depend on the —possibly large— number of samples N. The singular value decomposition of this matrix:

$$R_{11} = U\Sigma V^T.$$

can be used to compute (9) for any $\lambda^{(i)}$ as

$$d(\theta, \lambda^{(i)}) = -\mathcal{Q}_2 V \mathrm{diag}\left(\frac{\sigma_j}{\sigma_j^2 + \lambda^{(i)}}\right) U^T R_{12},$$

where σ_j is the jth singular value of R_{11} and $\mathrm{diag}(\beta_j)$ denotes a diagonal matrix of which the jth diagonal element equals β_j. Note that the singular value decomposition of R_{11} and the matrix products $\mathcal{Q}_2 V$ and $U^T R_{12}$ can be precalculated and reused for all values for $\lambda^{(i)}$.

3.2 Obtaining the Local Parameter Subspace

The left null space of the similarity map in (7) is calculated using the QR factorization (8). The reason for using a QR factorization rather than a singular value decomposition is efficiency; only one set of left Householder rotations needs to be calculated [2, p. 223]. Furthermore, given the Householder vectors, the matrix $\mathcal{Q}_2$ can be calculated directly, without calculating $\mathcal{Q}_1$, by applying all Householder rotations to $[0 \quad I]^T$. Efficient numerical implementations exist for this [2, p. 211].

3.3 Search Directions in a Local Parameter Subspace

The search direction (9) can be calculated efficiently using the QR factorization (15). However, it should be noted that for systems having a large order compared to the number of inputs and outputs, the matrix $M(\theta) \in \mathbb{R}^{(n+\ell)(n+m)\times n^2}$ is almost square so that the number of columns in $\Psi_N \mathcal{Q}_2$ is vastly smaller than the number of columns in Ψ_N.

This can be exploited by calculating $\Psi_N \mathcal{Q}_2$ directly, rather than first calculating Ψ_N and then multiplying by $\mathcal{Q}_2$. This represents a large saving in both memory usage and calculation time. The columns of $\Psi_N \mathcal{Q}_2$ are the directional derivatives of E_N in the directions specified by the columns of $\mathcal{Q}_2$,

$$\Omega_N := \Psi_N \mathcal{Q}_2 = \frac{\partial E_N}{\partial \theta^T} \mathcal{Q}_2.$$

The kth element of the jth column of Ω_N can be written as

$$\Omega_N(k,j) = \sum_{i=1}^{p} \frac{\partial \hat{y}(k;\theta)}{\partial \theta_i} \mathcal{Q}_2(i,j). \tag{16}$$

where p is the number of parameters in θ. This sum can be obtained by simulating only one dynamic system: as differentiation is a linear operation, the matrix derivatives in (12)–(13) can be replaced by similar weighted sums over i, which can be obtained from

$$\begin{aligned}
&\mathrm{vec}\left(\begin{bmatrix} \sum_{i=1}^{p} \frac{\partial A}{\partial \theta_i} \mathcal{Q}_2(i,j) & \sum_{i=1}^{p} \frac{\partial B}{\partial \theta_i} \mathcal{Q}_2(i,j) \\ \sum_{i=1}^{p} \frac{\partial C}{\partial \theta_i} \mathcal{Q}_2(i,j) & \sum_{i=1}^{p} \frac{\partial D}{\partial \theta_i} \mathcal{Q}_2(i,j) \end{bmatrix}\right) \\
&= \sum_{i=1}^{p} \frac{\partial}{\partial \theta_i} \mathrm{vec}\left(\begin{bmatrix} A & B \\ C & D \end{bmatrix}\right) \mathcal{Q}_2(i,j) \equiv \sum_{i=1}^{p} \frac{\partial \theta}{\partial \theta_i} \mathcal{Q}_2(i,j) \\
&= \sum_{i=1}^{p} \mathbf{e}_i \mathcal{Q}_2(i,j) = \mathcal{Q}_2(:,j),
\end{aligned} \tag{17}$$

in which $\mathbf{e}_i$ denotes a vector which contains zeros, except for the ith component, which equals one.

3.4 Optimization Problems with Huge Data Lengths

For optimization problems that have an exceedingly large number of samples N, the approach outlined so far is impractical. Even if memory is saved by calculating $\Psi_N \mathcal{Q}_2$ directly, the memory requirements may be too high.

Memory usage can be reduced by exploiting the fact that the Levenberg-Marquardt optimization function requires only the R factor of (15). This R factor can be built up recursively using a suitably small number N_b block rows of the Jacobian at a time. For the first N_b block rows, (15) is calculated. Subsequently, for the jth set of block rows, the $R^{(j)}$ factor is updated as

$$\begin{bmatrix} R_{11}^{(j)} & R_{12}^{(j)} \\ \Psi_{N_b} \mathcal{Q}_2 & E_{N_b} \end{bmatrix} = \begin{bmatrix} Q_1^{(j+1)} & Q_2^{(j+1)} \end{bmatrix} \begin{bmatrix} R_{11}^{(j+1)} & R_{12}^{(j+1)} \\ 0 & R_{22}^{(j+1)} \end{bmatrix}.$$

After the last update, the final R_{11} and R_{12} matrices can be used to obtain the search direction. Note that this recursive update is possible since we simulate the Jacobian columns; the next set of block rows can be calculated based on the stored final state of the previous set.

3.5 Fast Simulation

The simulations for E_N and Ψ_N using (12)–(13) constitute a substantial amount of the total number of calculations required to perform one iteration. Therefore, an efficient implementation of the state-trajectory calculation (12) has been written in C. This function is comparable to MATLAB's `ltitr`, but it is faster.

4 Maximum Likelihood Identification

If the output measurements are perturbed by colored noise, a nonlinear least squares identification will yield biased estimates [4, p. 178]. However, if information on the noise covariance matrix Σ_v is available, a maximum likelihood identification is possible. The cost function (5) is then weighted as

$$\begin{aligned}
V_N(\theta) = \frac{1}{N} E_N^T \Sigma_v^{-1} E_N &= \frac{1}{N} (\Sigma_v^{-1/2} E_N)^T (\Sigma_v^{-1/2} E_N) \\
&= \frac{1}{N} \widetilde{E}_N^T \widetilde{E}_N.
\end{aligned} \tag{18}$$

This implies that if a Cholesky factor $\Sigma_v^{-1/2}$ of the inverse covariance matrix (ICM) is known, an existing nonlinear least squares optimization framework can be used by setting $\widetilde{E}_N = \Sigma_v^{-1/2} E_N$ and $\widetilde{\Psi}_N = \Sigma_v^{-1/2} \Psi_N$. The multiplication of E_N and Ψ_N can be performed efficiently since $\Sigma_v^{-1/2}$ is sparse, as shown below. Numerical software can take this sparsity into account. This maximum likelihood weighting works independently on the different columns of Ψ_N. Therefore, its use is independent of the type of parameterization or type of system. This implies that it can be used for LPV and bilinear systems as well.

In general, the covariance matrix is a full $N\ell \times N\ell$ matrix, of which the formation and inversion requires a prohibitive amount of memory. However, recent work by David and Bastin [1] provides a solution since it yields an analytic and sparse expression for the ICM, if the noise model is assumed to be dth order auto-regressive (AR):

$$v(k) = \vec{e}\ (k) - \sum_{i=1}^{d} \vec{A}_i\ v(k-i) = \overleftarrow{e}\ (k) - \sum_{i=1}^{d} \overleftarrow{A}_i\ v(k+i). \tag{19}$$

In this equation, the matrices $\vec{A}_i, \overleftarrow{A}_i \in \mathbb{R}^{\ell \times \ell}$ describe causal and anti-causal AR models. The innovations $\vec{e}\ (k)$ and $\overleftarrow{e}\ (k)$ are zero-mean white-noise sequences with covariance matrices $\vec{\Sigma} = E[\vec{e}\ (k)\ \vec{e}\ (k)^T]$ and $\overleftarrow{\Sigma} = E[\overleftarrow{e}\ (k)\ \overleftarrow{e}\ (k)^T]$.

An expression for the ICM will first be derived. After this, a way to integrate the problem into a nonlinear least squares framework will be discussed.

4.1 The Inverse Covariance Matrix

David [1] takes the Gohberg-Heinig inversion theorem as a starting point in order to prove that the ICM of the AR process described by (19) is given by

$$\Sigma_v^{-1} = U^T (I_N \otimes \vec{\Sigma})^{-1} U - V^T (I_N \otimes \overleftarrow{\Sigma})^{-1} V. \tag{20}$$

Here, the U matrix is a block Toeplitz matrix for the causal model and the V matrix is a block Toeplitz matrix for the anti-causal model:

$$U = \text{toep}\left(I_\ell, \vec{A}_1, \ldots, \vec{A}_d, 0_{(N-d-1)\ell \times \ell}\right), \tag{21}$$

$$V = \text{toep}\left(0_{(N-d)\ell \times \ell}, \overleftarrow{A}_d, \ldots, \overleftarrow{A}_1\right). \tag{22}$$

The notation $\text{toep}(B_1, \ldots, B_N)$ has been adopted to denote a block Toeplitz matrix in terms of its impulse response kernel as

$$\text{toep}(B_1, \cdots, B_N) := \begin{bmatrix} B_1 & & & \\ B_2 & B_1 & & \\ \vdots & \vdots & \ddots & \\ B_N & B_{N-1} & \cdots & B_1 \end{bmatrix}.$$

It is crucial to realize that (20) is positive definite despite the fact that it is a subtraction of two terms; failure to realize this and applying (20) directly results in a cost function that contains both positive and negative terms. Such a cost function cannot be minimized using a nonlinear least squares framework, and one would have to resort to much slower methods such as the simplex method.

Although the ICM (20) is a structured matrix, an analytic Cholesky factor is not readily obtained. However, the structure does allow the efficient numerical calculation of a Cholesky factor.

4.2 A Cholesky Factor of the ICM

The first notion in the efficient numerical calculation of the ICM's Cholesky factor, is that the ICM is a block diagonal matrix that has just d block super-diagonals. Sparse storage schemes can thus result in a large saving in memory.

Second, the expression for Σ_v^{-1} can be simplified. Given the Cholesky factorization of the innovation covariances, $\vec{\Sigma}^{-1} = \vec{T}^T\ \vec{T}$ and $\overleftarrow{\Sigma}^{-1} = \overleftarrow{T}^T\ \overleftarrow{T}$, the filter coefficients are premultiplied as $\vec{\mathcal{A}}_i = \vec{T}\vec{A}_i$ and $\overleftarrow{\mathcal{A}}_i = \overleftarrow{T}\overleftarrow{A}_i$. Straightforward derivation now shows that (20) can be written without block diagonal weighting as $\Sigma_v^{-1} = \mathcal{U}^T\mathcal{U} - \mathcal{V}^T\mathcal{V}$, where $\mathcal{U}$ and $\mathcal{V}$ are based on $\vec{\mathcal{A}}_i$ and $\overleftarrow{\mathcal{A}}_i$ in analogy to (21) and (22).

In addition, the calculation of Σ_v^{-1} can be simplified further by exploiting the block Toeplitz structure in $\mathcal{U}$ and $\mathcal{V}$. And finally, only the part of the band above the diagonal needs to be calculated since a Cholesky factorization of a real matrix needs only that part.

Because of the block Toeplitz structure of $\mathcal{U}$, the block elements depend only on the distance to the block diagonal. Apart from the lower right $\ell d \times \ell d$ corner, the block elements of the ith block super-diagonal are given by

$$(\mathcal{U}^T\mathcal{U})_{\text{ith block super-diagonal}} = \sum_{k=0}^{d-i} \vec{\mathcal{A}}_{k+i}{}^T\ \vec{\mathcal{A}}_k\ .$$

This implies that only d such blocks need to be precalculated and that these cached copies can be used to fill a large part of $\mathcal{U}^T\mathcal{U}$. The lower-right $\ell d \times \ell d$ part is small and is most easily calculated explicitly as

$$(\mathcal{U}^T\mathcal{U})_{((N-d)\ell+1:\,N\ell,(N-d)\ell+1:\,N\ell)} = \text{toep}(\vec{\mathcal{A}}_1, \ldots, \vec{\mathcal{A}}_d)^T \text{toep}(\vec{\mathcal{A}}_1, \ldots, \vec{\mathcal{A}}_d).$$

Since $\mathcal{V}^T\mathcal{V}$ has nonzero entries in only its upper-left $\ell d \times \ell d$ part, only this upper-left part has to be calculated. This part is equal to

$$(\mathcal{V}^T\mathcal{V})_{(1:\,\ell d,1:\,\ell d)} = \text{toep}(\overleftarrow{\mathcal{A}}_d, \ldots, \overleftarrow{\mathcal{A}}_1)^T \text{toep}(\overleftarrow{\mathcal{A}}_d, \ldots, \overleftarrow{\mathcal{A}}_1).$$

The implication of these facts is that one does not have to form $\mathcal{U}$ or $\mathcal{V}$ explicitly. The matrix Σ_v^{-1} can be calculated directly, and only a very small part of it actually has to be calculated. The Cholesky factor of the resulting upper triangular block band matrix can be calculated using a sparse Cholesky factorization, e.g. in MATLAB.

5 Experiments

In this section the following steam-engine model is used in a Monte-Carlo simulation study to show that the proposed implementation is both efficient and accurate:

$$A = \begin{bmatrix} 0.3274 & 0.4685 & -0.0608 & -0.3670 \\ 0.0379 & 0.7852 & -0.7436 & 0.1048 \\ 0.1169 & -0.0469 & 0.7596 & -0.2108 \\ 0.1272 & 0.0300 & 0.0660 & -0.3001 \end{bmatrix},$$

$$B = \begin{bmatrix} -0.0084 & -0.4142 \\ -0.0590 & 0.1041 \\ 0.0508 & 0.0516 \\ -0.0092 & -0.0949 \end{bmatrix}, \quad D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix},$$

$$C = \begin{bmatrix} -0.8937 & 0.1707 & 0.0906 & 0.0550 \\ -0.0862 & -0.4803 & -0.5923 & -0.4631 \end{bmatrix}.$$

The input signal $u(k)$ is a unit-variance Gaussian white-noise sequence. The output disturbance $v(k)$ is generated as

$$v_1(k) = e_1(k) + 0.8e_2(k) + 0.9v_1(k-1) - 0.95v_1(k-2),$$
$$v_2(k) = 0.8e_1(k) + e_2(k) + 0.8v_2(k-1) - 0.85v_2(k-2),$$

in which $e_1(k)$ and $e_2(k)$ are two uncorrelated unit-variance Gaussian white-noise sequences.

5.1 Attaining the Cramer-Rao Bound

In the Monte-Carlo simulations we simulate the system for several sample sizes and subsequently add the output disturbance which has been described above.

The PI-MOESP subspace method [12] is used to generate initial estimates of the system matrices. A block size $s = 16$ is used in this algorithm. These matrices are used as an initial starting point for the output error optimization algorithm described in Section 2.

The residual that results after the output error optimization is used to estimate a multivariable AR model according to [1]. Based on this AR model, a maximum likelihood optimization is carried out as described in Section 4. The residual that results after this step is used to estimate an improved AR model. With this improved AR model a second and final maximum likelihood optimization is performed.

We repeat this identification experiment 1000 times using different realizations for $e(k)$. In order to make a comparison between these different models possible, they are transformed into the output normal form [7], which provides a unique minimal parameterization. When denoting the output normal parameters of a state-space model for the ith experiment as $\widehat{\theta}_{\mathrm{ON}}^{(i)}$, the mean square error (MSE) of the parameters is defined as

$$\mathrm{MSE} = \frac{N}{1000} \sum_{i=1}^{1000} (\widehat{\theta}_{\mathrm{ON}}^{(i)} - \theta)^T (\widehat{\theta}_{\mathrm{ON}}^{(i)} - \theta).$$

This MSE is assessed by comparing it to the trace of the Cramer-Rao lower bound covariance matrix, which is the lowest possible attainable MSE. It can be calculated based on the true parameters θ^* as

$$\Sigma_{\mathrm{CR}} = N \left(\Psi_N(\theta^*)^T \Sigma_v^{-1} \Psi_N(\theta^*) \right)^{-1}.$$

Figure 1 shows the MSE for the subspace, output error and maximum likelihood identifications in relation to the Cramer-Rao bound for different sample sizes. From these results it is clear that subspace identification does not converge to the Cramer-Rao bound, a result which is consistent with [8]. For the maximum likelihood results, on the other hand, the Cramer-Rao bound is attained.

It is important to note that knowledge of the true noise model is never used in the maximum likelihood identification; the noise model is derived from measured residuals in an iterative fashion. This makes the method usable in practice.

5.2 Numerical Efficiency

This section will illustrate the effects of the numerically efficient implementation discussed in Section 3. The efficient implementation is compared to the nonlinear optimization function `leastsq` from the MATLAB Optimization Toolbox, which was adapted to incorporate the local gradient search. The cost function used with the modified `leastsq` function, uses MATLAB's internal `ltitr` state trajectory calculation function. The efficient implementation, on the other hand, uses a fast C-implementation.

The same model and input signals as in the previous section are used. However, the experiment is performed in a noise-free setting, since we merely wish to show the performance gain of our implementation. The initial estimate that is given to the optimization function consists of the actual system matrices, where random variables with standard deviation 0.01 are added to each element.

This identification experiment is repeated 100 times, and the optimization time and memory usage is recorded. Figure 2 shows the optimization times for the MATLAB `leastsq` function and our efficient implementation. For sample sizes above 10 000, our implementation uses the Huge Data Length extension discussed in Section 3.4 with $N_b = 10\,000$. It is clear that the efficient implementation is about a factor 3 faster than the MATLAB implementation for moderate sample sizes. For sample sizes 100 000 and 200 000, the MATLAB implementation becomes exceedingly slower, because the Jacobian does not fit into physical memory anymore. In contrast, our implementation requires only about 60 Megabytes for $N = 200\,000$.

6 Conclusions

An efficient implementation of the nonlinear least squares and maximum likelihood identification of LTI state-space models has been described. It has been shown that the maximum likelihood identification can be efficiently implemented as a suitable weighting of the nonlinear least squares problem. Since well-established and efficient algorithms exist for the latter, this is an advantageous circumstance.

The maximum likelihood weighting has been shown to be an output weighting. This implies that it can be used for any kind of system with a suitable parameterization for which an output-error description can be made. Examples of these for which maximum likelihood identification will be implemented in the near future, are bilinear and LPV systems.

The nonlinear optimization itself consists of a local parameterization of the state-space model and a subsequent gradient search in the resulting local parameter space. For high-order systems, the calculation of this local parameter subspace becomes the bottleneck as far as memory requirements are concerned; it involves the calculation of the left null space of a large sparse matrix with a nontrivial structure. It is therefore recommended that the structure of this matrix is exploited in future research.

Monte-Carlo simulations have shown that the maximum likelihood framework attains the Cramer-Rao lower bound when PI-MOESP is used to obtain an initial model estimate.

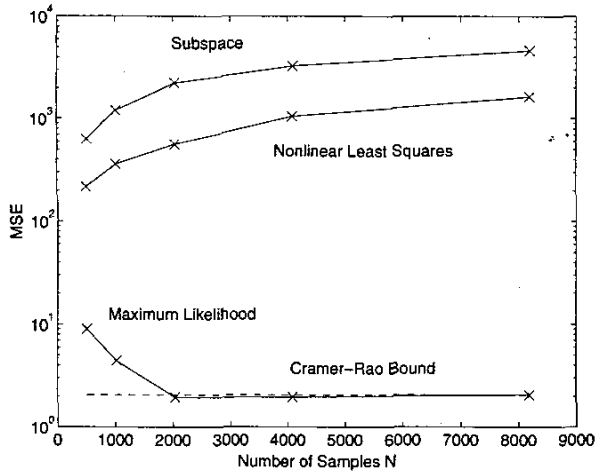

Figure 1: Mean Square Error of estimated parameters as a function of the sample size N for different identification methods.

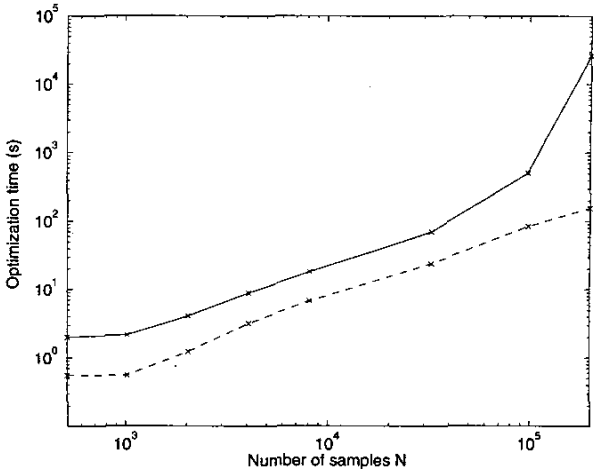

Figure 2: Optimization time as a function of the sample size N for the modified MATLAB `leastsq` function (solid) and the efficient implementation discussed in this paper (dashed).

Furthermore, the efficient implementation discussed in this paper is shown to be faster than the MATLAB Optimization Toolbox while requiring less memory.

References

[1] B. David. *Parameter Estimation in Nonlinear Dynamical Systems with Correlated Noise.* PhD thesis, Université Catholique de Louvain, November 2001.

[2] G. H. Golub and C. F. van Loan. *Matrix Computations.* The John Hopkins University Press, Baltimore, 3 edition, 1996.

[3] L. H. Lee and K. Poolla. Identification of linear parameter-varying systems using nonlinear programming. *Journal of Dynamic Systems, Measurement and Control*, 121(1):71–78, March 1999.

[4] L. Ljung. *System Identification: Theory for the User.* Prentice Hall, Englewood Cliffs, New Jersey, 1987.

[5] T. McKelvey. *Identification of State-Space Model from Time and Frequency Data.* PhD thesis, Linköping University, Linköping, Sweden, 1995.

[6] J. J. Moré. The Levenberg-Marquardt algorithm: Implementation and theory. In G.A. Watson, editor, *Numerical Analysis*, volume 630 of *Lecture Notes in Mathematics*, pages 106–116. Springer Verlag, 1978.

[7] R. Ober. Balanced realizations: Canonical form, parametrization, model reduction. *International Journal of Control*, 46(2):643–670, 1987.

[8] K. Peternell, W. Scherrer, and M. Deistler. Statistical analysis of novel subspace identification methods. *Signal Processing*, 52:161–177, 1996.

[9] V. Verdult. *Nonlinear System Identification: A State-Space Approach.* PhD thesis, University of Twente, Faculty of Applied Physics, Enschede, The Netherlands, 2002.

[10] V. Verdult, N. H. Bergboer, and M. Verhaegen. Maximum likelihood identification of multivariable bilinear state-space systems by projected gradient search. In *Proceedings of the 41st IEEE Conference on Decision and Control*, Las Vegas, Nevada, December 2002.

[11] V. Verdult and M. Verhaegen. Identification of multivariable LPV state space systems by local gradient search. In *Proceedings of the European Control Conference 2001*, Porto, Portugal, September 2001.

[12] M. Verhaegen. Subspace model identification part 3: Analysis of the ordinary output-error state-space model identification algorithm. *International Journal of Control*, 56(3):555–586, 1993.