APPROXIMATE LEAST SQUARES PARAMETER ESTIMATION WITH STRUCTURED OBSERVATIONS

Atulya Yellepeddi

Electrical Eng./Applied Ocean Physics and Eng.
MIT-WHOI Joint Program
email: atulya@mit.edu

James C. Preisig

Applied Ocean Physics and Engineering
Woods Hole Oceanographic Institution
email: jpreisig@whoi.edu

ABSTRACT

The solution of inverse problems where the parameter being estimated has a known structure has been widely studied. In this work, we consider the situation where it is not appropriate to assume a structure for the parameter, but the observations on which the estimate are based are structured; specifically, when the observations are parametrized by a decomposable graphical model. This translates to structured sparsity of the inverse covariance matrix for Gaussian distributed observation vectors. We present an approximate least squares method which takes advantage of the structure to reduce the complexity of least squares. The approximate least squares method can be implemented recursively for even lower complexity. It is shown that the proposed method is asymptotically equivalent to least squares parameter estimation for a large number of observations. The properties of the algorithm are verified by simulation.

Index Terms— Least squares methods, graphical models, adaptive algorithms

1. INTRODUCTION

Linear inverse or parameter estimation problems are amongst the most important challenges in signal processing. Detailed descriptions of various problems that fall into this framework may be found in, for instance, [1, 2].

The basic linear parameter estimation problem is that of finding a vector $\boldsymbol{h}$ from a number of observations of $\boldsymbol{x}(n)$ and $y(n)$, where

$$y(n) = \boldsymbol{h}^{\dagger}\boldsymbol{x}(n) + v(n),\ n = 1, 2, \ldots, N. \tag{1}$$

Note that † represents the Hermitian operation, and $v(n)$ is noise. [1]

In recent years, there has been much interest in taking advantage of known properties of the system in order to either improve the performance or reduce the computational complexity of the estimation algorithm. The case when the parameter $\boldsymbol{h}$ is sparse has been the subject of special interest, both for the model of (1) and generalizations [3–6]. In [7], the problem of linear inference for multi-resolution models with a sparse inverse covariance structure is considered. Sparse recovery with graphical model prior constraints is considered in [8–10] and references therein. All of these methods exploit constraints on the parameter that is being estimated.

In this work, a related but different problem is considered. We consider the case in which $\boldsymbol{h}$ is not constrained, but the observations[2] $\boldsymbol{x}(n)$ are structured. The assumption is that it is hard to explicitly constrain the parameter that is being estimated, but the system has structure, which manifests itself as structured observations.

An example of a motivating situation is the problem of estimating equalizer coefficients. In this case, the random vectors $\boldsymbol{x}(n)$ correspond to outputs of a channel. Depending upon the channel, therefore, they may be quite structured. However, based on this information, it may be hard to impose structure on the equalizer coefficients. The proposed model thus generalizes the problem of constraining the parameter to exploiting constraints that are imposed in a less obvious manner.

Specifically, the structure assumed is that each vector $\boldsymbol{x}(n)$ is Gaussian distributed, and that the distribution factors according to an undirected graphical model. For a Gaussian vector, this implies a structured sparsity constraint on the inverse covariance matrix [11]. Since the inverse covariance matrix is central to (linear) estimation, we are able to exploit such structured sparsity in developing the proposed method.

A common method of solving problems of the form of (1) is *least squares*, since it performs fairly well in practice. Moreover, it is possible to cast least squares into a computationally efficient recursive form, termed *recursive least squares*, which can be used (with exponential windowing) to track $\boldsymbol{h}$ as it changes over time. The reader is referred to [12] for a discussion of these and other properties of Least Squares.

In this paper we present an algorithm that utilizes the structure of the covariance matrix of the observations to form an approximate least squares estimate that is significantly less computationally intensive than traditional least squares and is also easy to parallelize. It is easy to recast the algorithm as a recursive algorithm similar to the RLS algorithm. It is shown that in the large data limit, as $N \to \infty$, the approximate method converges to the least squares solution.

2. PRELIMINARIES

Before developing the algorithm, it is necessary to state a few assumptions and results. Note that boldface lowercase represents vectors and boldface uppercase denotes matrices; and a subscript for a vector represents the component of the vector (e.g., $\boldsymbol{h}_1$ means the first component of $\boldsymbol{h}$).

Work supported by ONR Grant #N00014-11-10426, NSF Grant #ECCS-1102156, and the WHOI Academic Programs office

[1] In (1), $y(n)$ is assumed to be scalar and $\boldsymbol{x}(n)$ is a vector of the same dimension as $\boldsymbol{h}$, although it is easy to extend this to the case of vectors and matrices of suitable dimension, respectively.

[2] In the literature, $\boldsymbol{x}(n)$ is sometimes referred to as the "input" to the system and $y(n)$ as the observation. However, from the point of view of the estimation algorithm, they are both observations based on which the parameter is estimated. Thus, in this work, we interpret $\boldsymbol{x}(n)$ as observations.

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

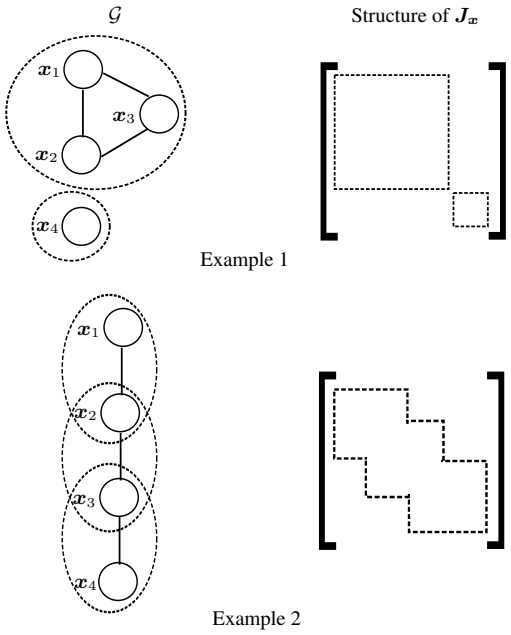

Fig. 1. Relationship between a graph $\mathcal{G}$ and the inverse covariance matrix $\boldsymbol{J}_{\boldsymbol{x}}$. Entries of the $\boldsymbol{J}_{\boldsymbol{x}}$ outside the dotted lines are 0. Dashed circles around nodes of $\mathcal{G}$ represent maximal cliques, corresponding to each of which a non-zero block is present in $\boldsymbol{J}_{\boldsymbol{x}}$.

First, with reference to (1), the length of $\boldsymbol{h}$ (and of $\boldsymbol{x}(n)$ for each n) is M. We assume that the vectors $\boldsymbol{x}(n)$ are independent at different times, and each vector has a multivariate complex normal distribution with zero mean and a positive definite covariance matrix $\boldsymbol{R}_{\boldsymbol{x}}$, i.e., $\boldsymbol{x}(n) \overset{\text{iid}}{\sim} \mathcal{CN}(\boldsymbol{0}, \boldsymbol{R}_{\boldsymbol{x}})$. We further denote $\boldsymbol{J}_{\boldsymbol{x}} = \boldsymbol{R}_{\boldsymbol{x}}^{-1}$ as the inverse covariance matrix of $\boldsymbol{x}(n)$. Finally, it is assumed that $v(n)$ is zero-mean white noise with covariance σ^2 and that it is uncorrelated with $\boldsymbol{x}(n)$.

The constraint that is introduced on the vectors $\boldsymbol{x}(n)$ is that the probability density function of each $\boldsymbol{x}(n)$ factors according to a decomposable undirected graphical model $\mathcal{G}$ of M nodes. In other words, the joint distribution of the vector $\boldsymbol{x}(n)$ can be written as the product of potentials over the cliques of the graph $\mathcal{G}$ (divided by a suitable normalization factor). When this happens, we say that the vectors $\boldsymbol{x}(n)$ are parameterized by the graph $\mathcal{G}$. The underlying graph $\mathcal{G}$ is decomposable if (and only if) it is chordal. These concepts are discussed in detail in [13, 14].

For multivariate Gaussian vectors which are parameterized by a graph, the inverse covariance matrix of the vectors have entries of 0 wherever there is no edge between the corresponding nodes in the graph [11], i.e., $J_{x_{ij}} = 0 \iff (i,j) \notin \mathcal{G}$ (by convention, every node is connected to itself, so that $(i,i) \in \mathcal{G}$). This is demonstrated in Figure 1. Thus, the graphical model constraint on the random vectors is equivalent to a sparsity constraint on the inverse covariance matrix. It is assumed that this sparsity pattern is known (though the matrix itself is not known).

The objective of the problem is to estimate $\boldsymbol{h}$ based on N observations of $\boldsymbol{x}(n)$ and $y(n)$. The baseline algorithm to estimate $\boldsymbol{h}$ is the least squares algorithm. The least squares estimate based on N observations, provided $N \geq M$, is given by [12]:

$$\hat{\boldsymbol{h}}_{\text{LS}}(N) = \boldsymbol{R}_{\boldsymbol{x}}^{-1}(N)\boldsymbol{r}_{\boldsymbol{x}y}(N) \tag{2}$$

where:

$$\boldsymbol{R}_{\boldsymbol{x}}(N) = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}(n)\boldsymbol{x}^{\dagger}(n) \tag{3a}$$

$$\boldsymbol{r}_{\boldsymbol{x}y}(N) = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}(n)y^{*}(n) \tag{3b}$$

With this background, we develop a low-complexity algorithm that uses the graphical model, or equivalently, the sparsity of the inverse covariance matrix, to form an approximate least squares estimate of $\boldsymbol{h}$.

3. PROPOSED METHOD

Observe that, in the absence of other constraints on $\boldsymbol{x}(n)$, $\boldsymbol{R}_{\boldsymbol{x}}(N)$ is the Maximum Likelihood estimate of $\boldsymbol{R}_{\boldsymbol{x}}$ from N samples of $\boldsymbol{x}$. Similarly, $\boldsymbol{r}_{\boldsymbol{x}y}(N)$ is the ML estimate of the cross correlation between $\boldsymbol{x}(n)$ and $y(n)$. Moreover it is a simple matter to show that $\boldsymbol{R}_{\boldsymbol{x}}^{-1}(N)$ is the ML estimate of $\boldsymbol{J}_{\boldsymbol{x}}$. Hence, the Least Squares algorithm multiplies the ML estimate of the inverse covariance matrix with the ML estimate of the cross-correlation between $\boldsymbol{x}(n)$ and $y(n)$.

In [15], the authors presented expressions for the ML estimate for the inverse covariance matrix when the vector is parameterized by a decomposable graphical model. The ML estimate can be computed by combining local estimates at different cliques.

Specifically, a decomposable graph $\mathcal{G}$ can be broken into a set of maximal cliques $\mathcal{C}$ and separators $\mathcal{S}$[3]. For any $c \in \mathcal{C}$, let $\boldsymbol{x}_c$ be the vector of variables in that clique. Then define

$$\boldsymbol{R}_{\boldsymbol{x}_c}(N) = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_c(n)\boldsymbol{x}_c^{\dagger}(n) \tag{4a}$$

$$\boldsymbol{r}_{\boldsymbol{x}_c y}(N) = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_c(n)y^{*}(n) \tag{4b}$$

and similar quantities for a separator. Then, the ML estimate of $\boldsymbol{J}_{\boldsymbol{x}}$ based on N observations, subject to parameterization by the graph $\mathcal{G}$ is denoted $\tilde{\boldsymbol{J}}_{\boldsymbol{x}}(N)$, and given by [15]:

$$\tilde{\boldsymbol{J}}_{\boldsymbol{x}}(N) = \sum_{c\in\mathcal{C}} \left[\boldsymbol{R}_{\boldsymbol{x}_c}^{-1}(N)\right]_0 - \sum_{s\in\mathcal{S}} \left[\boldsymbol{R}_{\boldsymbol{x}_s}^{-1}(N)\right]_0 \tag{5}$$

where $[\cdot]_0$ is an operator that takes the matrix in its argument and places it at the appropriate location in a conformable matrix, padding the rest of the resulting matrix with zeros. The location corresponds to the position of the variables of that particular clique in the matrix. For instance, if the clique has $\boldsymbol{x}_1, \boldsymbol{x}_2$ and $\boldsymbol{x}_3$, then the matrix in the argument would be placed in the corresponding 3×3 sub-matrix.

Our algorithm essentially replaces the "unconstrained" ML estimate (3a) with the estimate of (5), which is the true ML estimate with the graph parameterization constraint. Of course, it seems that we should also represent $\boldsymbol{r}_{\boldsymbol{x}y}(N)$ in a similar manner. However, it is simple to verify by linearity for a decomposable graphical model,

$$\boldsymbol{r}_{\boldsymbol{x}y}(N) = \sum_{c\in\mathcal{C}} \left[\boldsymbol{r}_{\boldsymbol{x}_c}(N)\right]_0 - \sum_{s\in\mathcal{S}} \left[\boldsymbol{r}_{\boldsymbol{x}_s}(N)\right]_0 \tag{6}$$

[3]While we can break any graph into sets of its cliques and separators, in this work, it is required that the sets $\mathcal{C}$ and $\mathcal{S}$ satisfy the running intersection property, which is true if, and only if, $\mathcal{G}$ is decomposable.

so $\boldsymbol{r}_{\boldsymbol{x}y}(N)$ is already in the right form.[4]

Hence, the estimate using the structured version of Least Squares is given by:

$$\hat{\boldsymbol{h}}_{\text{sLS}}(N) = \tilde{\boldsymbol{J}}_{\boldsymbol{x}}(N)\boldsymbol{r}_{\boldsymbol{x}y}(N) \tag{7}$$

A few important things should be said about the proposed structured LS algorithm.

3.1. Combination of Local Estimates:

Using (5) in (7)

$$\begin{aligned}\hat{\boldsymbol{h}}_{\text{sLS}}(N) &= \left(\sum_{c\in\mathcal{C}}\left[\boldsymbol{R}_{\boldsymbol{x}_c}^{-1}(N)\right]_0 - \sum_{s\in\mathcal{S}}\left[\boldsymbol{R}_{\boldsymbol{x}_s}^{-1}(N)\right]_0\right)\boldsymbol{r}_{\boldsymbol{x}y}(N) \\ &= \sum_{c\in\mathcal{C}}\left[\boldsymbol{R}_{\boldsymbol{x}_c}^{-1}(N)\boldsymbol{r}_{\boldsymbol{x}_c y}(N)\right]_0 - \sum_{s\in\mathcal{S}}\left[\boldsymbol{R}_{\boldsymbol{x}_s}^{-1}(N)\boldsymbol{r}_{\boldsymbol{x}_s y}(N)\right]_0\end{aligned} \tag{8}$$

Thus, the structured LS method combines the least squares estimates at each clique (and subtracts separators to avoid double counting). This means it is *parallelizable*, and it can be implemented in a *distributed* fashion, if the application so demands.

3.2. Recursive Implementation:

The Least Squares estimate of (2) can be implemented in a recursive form [12], called the Recursive Least Squares filter. This is a highly efficient implementation when the data is received in an online manner. The RLS update step from time (data point) $N-1$ to N is:

$$\begin{aligned}\hat{\boldsymbol{h}}_{\text{LS}}(N) &= \hat{\boldsymbol{h}}_{\text{LS}}(N-1) \\ &\quad + \boldsymbol{R}_{\boldsymbol{x}}^{-1}(N)\boldsymbol{x}(N)\left[y(N) - \hat{\boldsymbol{h}}_{\text{LS}}^{\dagger}(N-1)\boldsymbol{x}(N)\right]^*\end{aligned} \tag{9}$$

Define $e(N) = y(N) - \hat{\boldsymbol{h}}_{\text{LS}}^{\dagger}(N-1)\boldsymbol{x}(N)$ as the *innovation* at time N. It can be shown using the matrix inversion lemma that the update equation above can be run in $\mathcal{O}\left(M^2\right)$ time.

From (8), the structured LS algorithm is the combination of a least squares algorithm for each clique. Hence, it seems clear that each of these local LS algorithms can be run recursively using an update equation similar to (9), with $\boldsymbol{x}(N)$ replaced with $\boldsymbol{x}_c(N)$, and $\boldsymbol{R}_{\boldsymbol{x}}(N)$ replaced with $\boldsymbol{R}_{\boldsymbol{x}_c}(N)$. However, there is an important change to note. As (8) indicates, the local estimates have to be combined to form a global estimate of $\boldsymbol{h}$, which is thereafter used to filter the overall input $\boldsymbol{x}(n)$. In other words the filtering operation is global. Therefore, the innovation term, which is computed using the output of the filter, is computed globally. Hence, when implemented recursively, the update equations for the structured RLS algorithm are given by:

$$\begin{aligned}\hat{\boldsymbol{h}}_{\text{sLS},c}(N) &= \hat{\boldsymbol{h}}_{\text{sLS},c}(N-1) \\ &\quad + \boldsymbol{R}_{\boldsymbol{x}_c}^{-1}(N)\boldsymbol{x}_c(N)\left[y(N) - \hat{\boldsymbol{h}}_{\text{sLS}}^{\dagger}(N-1)\boldsymbol{x}(N)\right]^*\end{aligned} \tag{10}$$

and

$$\hat{\boldsymbol{h}}_{\text{sLS}}(N) = \sum_{c\in\mathcal{C}}\left[\hat{\boldsymbol{h}}_{\text{sLS},c}(N)\right] - \sum_{s\in\mathcal{S}}\left[\hat{\boldsymbol{h}}_{\text{sLS},s}(N)\right]_0 \tag{11}$$

[4] However, see 4.2.2.

Thus, the structured Least Squares algorithm can be implemented in a recursive form. Moreover, each clique can run its own RLS algorithm, i.e., the recursive processing can be done in parallel or distributed form, and only the innovation needs to be computed globally and shared. These are valuable properties that show how to parallelize the least squares algorithm effectively when the observations are parameterized by a decomposable graph.

The advantage of a recursive form is that it can be used to track time-varying systems, with an exponential weighting factor. This leads to some interesting observations. For instance, the structured RLS form can track the overall coefficient vector with different averaging intervals for cliques whose coefficients vary at different rates.

4. PERFORMANCE CHARACTERIZATION

4.1. Computation Complexity

When the least squares algorithm is implemented in a non-recursive form, the slowest step is the direct matrix inversion, which has a complexity of $\mathcal{O}\left(M^{2.8}\right)$ (faster algorithms are known, but are practical only for enormous matrices; see [16] and references therein). Conventional least squares thus has a complexity of $\mathcal{O}\left(M^{2.8}\right)$.

Defining $|c|$ as the number of variables in clique c, the structured least squares has a complexity of $\mathcal{O}\left(\sum_{c\in\mathcal{C}}|c|^{2.8} + \sum_{s\in\mathcal{S}}|s|^{2.8}\right)$, since a matrix inversion is required for each clique and separator (assuming no parallelization). The number of cliques is obviously upper bounded by M. If we further assume that $\max_{c\in\mathcal{C}}|c| \leq K$ (for a decomposable graph, the largest separator has to be smaller than the largest clique), then it follows that the complexity is $\mathcal{O}\left(K^{2.8}M\right)$, for a purely serial implementation using matrix inversion. The performance could be improved by a factor of M with parallelization.

A similar argument shows that using the recursive implementation of 3.2, the complexity of the structured algorithm is $\mathcal{O}\left(K^2M\right)$ (at each time), as opposed to $\mathcal{O}\left(M^2\right)$ for conventional RLS.

This indicates the regime in which the algorithm is of the most interest: when $M \gg K$, i.e., when the largest clique only contains a small fraction of the variables. In this regime, the complexity is essentially linear in M, as opposed to nearly cubic for conventional least squares, since the higher order terms are not important.

4.2. Convergence Behavior and Estimation Error

4.2.1. Large N

First consider the case when $N \to \infty$. Practically, this corresponds to the case where the $N \gg M$.

It was mentioned in [15] that as $N \to \infty$, $\tilde{\boldsymbol{J}}_{\boldsymbol{x}}(N) \xrightarrow{\text{a.s.}} \boldsymbol{J}_{\boldsymbol{x}}$. Indeed this follows from the fact that $\tilde{\boldsymbol{J}}_{\boldsymbol{x}}(N)$ is the *true* ML estimate of $\boldsymbol{J}_{\boldsymbol{x}}$ given the graph constraints, and the ML estimator is consistent. Combined with the fact that $\boldsymbol{r}_{\boldsymbol{x}y}(N) \xrightarrow{\text{a.s.}} \boldsymbol{r}_{\boldsymbol{x}y}$ (due to the strong law of large numbers), it is clear that $\hat{\boldsymbol{h}}_{\text{s LS}}(N) \xrightarrow{\text{a.s.}} \boldsymbol{R}_{\boldsymbol{x}}^{-1}\boldsymbol{r}_{\boldsymbol{x}y} = \hat{\boldsymbol{h}}_{\text{MMSE}}$ almost surely as $N \to \infty$. Thus, the structured LS estimate converges to the Minimum Mean Square Error (MMSE) estimate.

The conventional least squares estimate is also known to converge to the MMSE estimate. Hence, the structured Least Squares is asymptotically equivalent to the true least squares estimate. This should not be a surprise, as both $\boldsymbol{R}_{\boldsymbol{x}}^{-1}(N)$ and $\tilde{\boldsymbol{J}}_{\boldsymbol{x}}(N)$ converge to $\boldsymbol{J}_{\boldsymbol{x}}$ as $N \to \infty$.

Finally note that $\hat{\boldsymbol{h}}_{\text{MMSE}} \to \boldsymbol{h}$ as $\sigma \to 0$ (as the noise vanishes). Thus, the structured LS and conventional LS are both consistent asymptotically for $N \to \infty$ *and* $\sigma^2 \to 0$.

4.2.2. *Moderate N*

When $N \geq M$, but $N \sim M$, using the results of [17], it is possible to show that the conventional Least Squares estimate is consistent as $\sigma^2 \to 0$, i.e., that $\hat{\boldsymbol{h}}_{\text{LS}}(N) \xrightarrow{\text{a.s.}} \boldsymbol{h}$.[5] However, for the structured LS algorithm, it can be shown that:

$$\boldsymbol{h} - \hat{\boldsymbol{h}}_{\text{sLS}}(N) = \left(\boldsymbol{I} - \tilde{\boldsymbol{J}}_{\boldsymbol{x}}(N)\boldsymbol{R}_{\boldsymbol{x}}(N)\right)\boldsymbol{h} + \frac{\tilde{\boldsymbol{J}}_{\boldsymbol{x}}(N)}{N}\sum_{n=1}^{N}\boldsymbol{x}(n)v^*(n) \tag{12}$$

The first term is noise independent, and in general, is non-zero for finite N, so the structured LS algorithm is *not* consistent for finite N and as $\sigma^2 \to 0$. This is one instance in which the performance deteriorates over using conventional least squares.

It should be understood, however, that the loss of performance is not due to a "compromise." Indeed, as shown in [15], $\tilde{\boldsymbol{J}}_{\boldsymbol{x}}(N)$ is a *better* estimate of the true inverse covariance matrix than is $\boldsymbol{R}_{\boldsymbol{x}}^{-1}(N)$, which is only to be expected since it explicitly accounts for the structure. Rather, the reason for the degradation in performance is that merely writing the cross correlation as a linear combination of the cross correlation term for cliques as in (6) is not sufficient to exploit the graphical model structure when estimating the cross correlation term. A more sophisticated approach is needed to incorporate the structure into the cross-correlation term.

4.2.3. *Small N*

Computing the Least Squares solution (without regularization) requires that $N \geq M$ so that the matrix $\boldsymbol{R}_{\boldsymbol{x}}(N)$ is invertible almost surely. The largest matrix whose inverse is needed by the structured least squares algorithm, however, is $\max_{c\in\mathcal{C}}|c|$.

It has been noted in 4.1 that computationally, structured LS algorithm has the most utility when the size of the largest clique is much smaller than M. Assuming that this is regime of operation, structured LS usually needs much less data to provide a numerically stable estimate. Of course, as discussed in 4.2.2, the estimate so obtained is not necessarily very good.

4.3. Verification by Simulation

To verify the results and properties described, the conventional and structured LS algorithms were implemented to estimate an unknown channel of length $M = 10$, based on a number of observations N and at different noise levels (signal-to-noise ratios). The metric is the mean squared error, which is the normalized squared-ℓ_2 norm of the estimation error. The input vectors $\boldsymbol{x}(n)$ have a graphical model structure which is the same as the one shown in Example 2 of Figure 1 extended to 10 nodes (a Markov chain structure).

The results as a function of SNR for some values of N are shown in Figure 2. These results demonstrate the concepts discussed in 4.2.2. The structured LS algorithm saturates to an error floor with increasing SNR, where the performance of the conventional algorithm continues to improve.

Figure 3 has the results as a function of N for SNR$=$ 10dB. These demonstrate that for large N, the errors of both conventional and structured LS go to 0, i.e., both estimators are consistent, as discussed in 4.2.1. Moreover the small N effects discussed in 4.2.3 are also shown in the figure; for $N \leq 10$, the MSE of the conventional method blows up, but the structured LS does not.

[5] We don't need [17] to prove that if $\sigma^2 = 0$, then $\hat{\boldsymbol{h}}_{\text{LS}}(N) = \boldsymbol{h}$, but that fact alone does not guarantee "graceful" convergence.

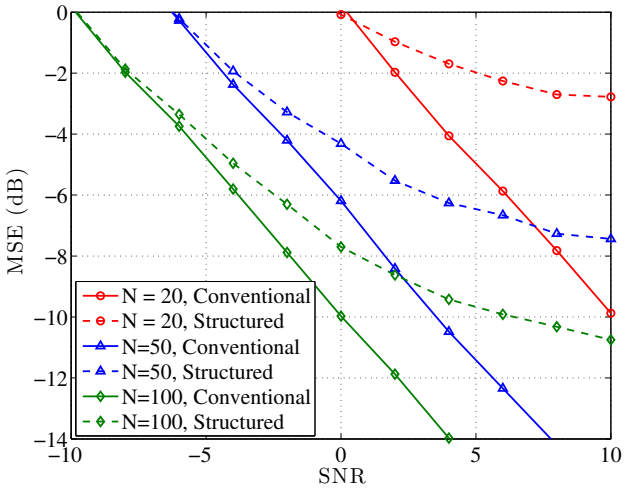

Fig. 2. Mean Square Error of Conventional and Structured LS as a function of SNR for various N

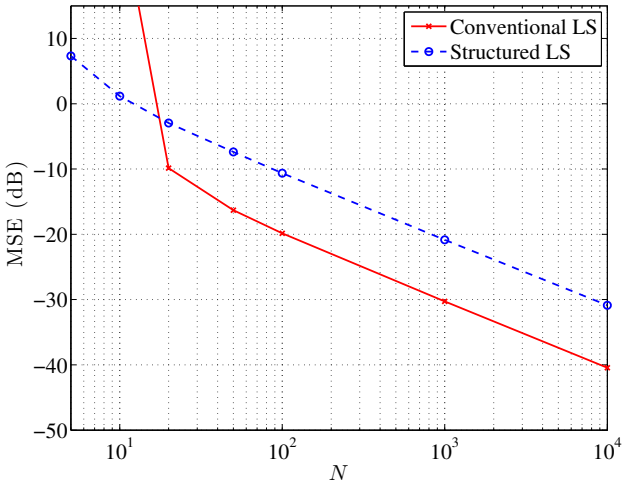

Fig. 3. Mean Square Error of Conventional and Structured LS as a function of N, SNR$=$ 10dB

5. CONCLUSIONS

In this work, an algorithm was introduced to solve the problem of parameter estimation with structured observations. The observations were assumed to be parameterized by a decomposable graphical model, which led to a structured inverse covariance matrix. Exploiting the structure of this matrix, a Structured Least Squares algorithm was developed. The useful properties of this algorithm were shown to include lowered complexity, the ability to be parallelized and distributed, the existence of a recursive structure and asymptotic consistency. The known drawbacks and error floors were discussed.

Various questions still remain. It has been stated that the performance degradation in certain regimes is due to the naiveté of the method of exploiting structure in computing the cross correlation. Exact methods for this could be developed. An analysis to better understand the error floor (i.e., the performance degradation) would be instructive. More research into the behavior of the algorithm in tracking applications is needed.

6. REFERENCES

[1] R. C. Aster, B. Borchers, and C. H. Thurber, *Parameter estimation and inverse problems*, 2nd ed. Academic Press, 2012.

[2] A. Tarantola, *Inverse Problem Theory: Methods for Data Fitting and Model Parameter Estimation*, 2nd ed. Elsevier Science, 1994.

[3] S. F. Cotter, B. D. Rao, K. Engan, and K. Kreutz-Delgado, "Sparse solutions to linear inverse problems with multiple measurement vectors," *Signal Processing, IEEE Transactions on*, vol. 53, no. 7, pp. 2477–2488, 2005.

[4] E. Herrholz and G. Teschke, "Compressive sensing principles and iterative sparse recovery for inverse and ill-posed problems," *Inverse Problems*, no. 12, Dec.

[5] A. S. Gupta and J. Preisig, "A geometric mixed norm approach to shallow water acoustic channel estimation and tracking," *Physical Communication*, no. 2, pp. 119–128.

[6] J. A. Tropp and S. J. Wright, "Computational methods for sparse solution of linear inverse problems," *Proceedings of the IEEE*, no. 6, pp. 948–958, Jun.

[7] M. Choi, V. Chandrasekaran, and A. Willsky, "Exploiting sparse Markov and covariance structure in Multiresolution models," in *Machine Learning, Proceedings of the 26th Annual International Conference on*. Association for Computing Machinery, 2009, pp. 1–8.

[8] D. L. Donoho, A. Maleki, and A. Montanari, "Message-passing algorithms for compressed sensing." *Proceedings of the National Academy of Sciences of the United States of America*, vol. 106, no. 45, Nov. 2009.

[9] V. Chandar, D. Shah, and G. W. Wornell, "A simple message-passing algorithm for compressed sensing," in *2010 IEEE International Symposium on Information Theory*. IEEE, Jun. 2010, pp. 1968–1972.

[10] A. Montanari, "Graphical model concepts in compressed sensing," in *Compressed Sensing, Theory and Applications*. Cambridge University Press, 2012, ch. 9, pp. 394–438.

[11] S. L. Lauritzen, *Graphical models*. Oxford University Press, 1996.

[12] S. Haykin, *Adaptive Filter Theory*, 4th ed. Pearson Education, 2002.

[13] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*, 1st ed. The MIT Press, 2009.

[14] M. I. Jordan, *Introduction to Probabilistic Graphical Models*. (In preparation), 2003.

[15] A. Wiesel, Y. Eldar, and A. Hero, "Covariance estimation in decomposable Gaussian graphical models," *Signal Processing, IEEE Transactions on*, vol. 58, no. 3, pp. 1482–1492, 2010.

[16] V. V. Williams, "Breaking the Coppersmith-Winograd barrier," *Unpublished Manuscipt*, 2011.

[17] M. Pajovic and J. C. Preisig, "Performance analysis of the least squares based LTI channel identification algorithm using random matrix methods," in *2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, no. 8. IEEE, Sep. 2011, pp. 516–523.