DISTRIBUTED ITERATE-COLLAPSE INVERSION (DICI) ALGORITHM FOR $L-$BANDED MATRICES

Usman A. Khan and José M. F. Moura

Carnegie Mellon University
Department of Electrical and Computer Engineering
Pittsburgh, PA 15213
{ukhan, moura}@ece.cmu.edu

ABSTRACT

In this paper, we present a distributed algorithm to invert $L-$banded matrices that are symmetric positive definite (SPD), when the submatrices in the band are distributed among several processing nodes. We provide a distributed iterate-collapse inversion (DICI) algorithm that converges, at each node, to the corresponding submatrices in the inverse of the $L-$banded matrix. The computational complexity of the DICI algorithm to invert an SPD $L-$banded $n \times n$ matrix can be shown at each node to be independent of the size, n, of the matrix. Local information exchange is carried out after each iteration to guarantee convergence. We apply this algorithm to invert the information matrices in a computationally efficient distributed implementation of the Kalman filter and show its application towards inverting arbitrary sparse SPD matrices.

Index Terms— Sparse matrices, Distributed algorithms, Matrix inversion, Kalman Filtering, Large-scale systems

1. INTRODUCTION

Banded matrices are found frequently in signal processing, e.g., in the context of discretization of partial differential equations governing a random field and autoregressive or moving average image modeling. When they are constrained to be symmetric positive definite (SPD) matrices, they are the inverses of the covariance matrices of causal or non-causal Gauss-Markovian random processes [1]. Furthermore, in linear algebra, solving a sparse large linear system of equations is a well studied problem, where a sparse matrix inverse is to be calculated. By employing matrix reordering algorithms [2], we can convert sparse SPD matrices to banded SPD matrices. Hence, computing the inverse of banded matrices efficiently is an important problem both in signal processing and linear algebra.

The direct inverse of banded matrices can be computed centrally but that requires extensive storage, communication, and computation. Algorithms to compute direct inverses include Gauss-Jordan elimination. Most inversion algorithms for SPD matrices involve a Cholesky factorization that is efficient on a single processor implementation as long as computation power and memory requirements are within limits. Incomplete Cholesky factorization is also an important method for solving large sparse SPD linear systems [3].

This work was partially supported by the DARPA DSO Advanced Computing and Mathematics Program Integrated Sensing and Processing (ISP) Initiative under ARO grant # DAAD 19-02-1-0180, by NSF under grants # ECS-0225449 and # CNS-0428404, by ONR under grant # MURI-N000140710747, and by an IBM Faculty Award.

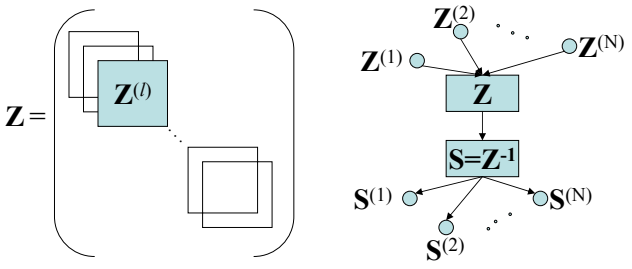


Fig. 1. Composition of the $L-$band of $\mathbf{Z}$ from the local matrices, $\mathbf{Z}^{(l)}$, shown in the left figure. Centralized Implementation of $\mathbf{S} = \mathbf{Z}^{-1}$, shown in the right figure.

Recursive inversion of banded matrices can be found in [4, 5]. In [4], a forward-backward recursion algorithm is provided to compute the inverse of a tridiagonal matrix, which involves a forward recursion to start from the first node and reach the last node, and a backward recursion that proceeds in the opposite direction. Since, the iterations involve serial communication of the local matrices among all the nodes, the associated latency is impractical, besides requiring an inordinate amount of communication.

We study the banded matrix inversion when the band of the matrix is distributed among several processing nodes, see figure 1 (left), and hence a distributed algorithm with local communication is intrinsic to the nature of the problem. This arises in problems where the information is distributed in a large geographical region, e.g., through distributed sensing, and hence the matrix to be inverted is distributed among the sensors in the system [6]. On the other hand, we may be interested in solving a very large linear system of equations on a multiprocessor machine where parallelizing the algorithm is essential in load balancing and its real-time implementation, and hence the matrix is distributed among different available processors. Typically for such problems $L \ll n$.

Consider $\mathbf{Z}$ to be an $L-$banded matrix (we refer to a matrix as an L-banded matrix ($L \geq 0$), if the elements outside the band defined by the Lth upper and the Lth lower diagonal are zero) and we are interested in computing $\mathbf{S} = \mathbf{Z}^{-1}$, when the non-zero submatrices along the main diagonal of $\mathbf{Z}$ are distributed among N processing nodes. This is shown in figure 1 (left). The lth node has a diagonal submatrix, $\mathbf{Z}^{(l)}$, termed as the *local* matrix at node l. We note here that the $L-$band of $\mathbf{Z}$ should be preserved among all the nodes, i.e., each element in the $L-$band of $\mathbf{Z}$ must be a member of at least one local matrix, $\mathbf{Z}^{(l)}$. Collecting the complete matrix at a central

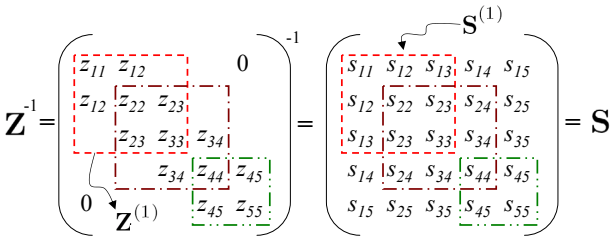


Fig. 2. DICI algorithm on a 5×5 $L = 1-$banded matrix $\mathbf{Z}$.

location, as shown in figure 1 (right), is infeasible because of the extensive computation and communication requirements.

We present a distributed iterate-collapse inversion algorithm, named DICI (pronounced die-see to sound like spicy), for $L-$banded SPD matrices. We point out here that the DICI algorithm will give the diagonal submatrices, $\mathbf{S}^{(l)}$, that lie in the $L-$band of $\mathbf{S}$ at the appropriate nodes. For the non $L-$band elements in $\mathbf{S}$, we provide a theorem to calculate them, with only local communication, from the elements in the $L-$band of $\mathbf{S}$. This process is also shown in figure 2 for a 5×5 $L = 1-$banded matrix $\mathbf{Z}$.

The DICI algorithm is divided into an iterate step and a collapse step. The iterate step is implemented on the elements inside the $L-$band. We then employ a non-linear collapse step on the non $L-$band elements that exploits the structure of the matrix $\mathbf{Z}$ and reduces the computation requirements of the ordinary Jacobi algorithm. It is noteworthy here that when Jacobi or Gauss-Seidel type (block) iterative algorithms [7, 8] are used to solve linear system of matrix equations ($\mathbf{ZS} = \mathbf{B}$, for $\mathbf{S}, \mathbf{B}$ matrices), they implement n linear system of vector equations ($\mathbf{Zs} = \mathbf{b}$, for $\mathbf{s}, \mathbf{b}$ vectors). On the other hand, the DICI algorithm employs a non-linear collapse step by exploiting the structure of the matrix $\mathbf{Z}$. This collapse step makes the computation complexity of the DICI algorithm independent of the size, n, of the matrix. The algorithm scales efficiently when compared to $O(n^3)$ direct inversion algorithms and $O(n^2)$ fast inversion algorithms, e.g., [4].

We summarize the rest of the paper. Section 2 presents the Jacobi algorithm with its extensions to matrices, section 3 gives the distributed implementation of the Jacobi algorithm. We present the DICI algorithm in section 4, and its application to distributed Kalman filter in subsection 5.1 and to sparse matrix inversion in subsection 5.2. Section 6 concludes the paper.

2. JACOBI ALGORITHM

The Jacobi algorithm [7] is an iterative algorithm to solve

$$\mathbf{Zs} = \mathbf{b} \tag{1}$$

for $\mathbf{s}$, by successive substitution, where $\mathbf{s} \in \mathbb{R}^n$ and $\mathbf{b} \in \mathbb{R}^n$ are vectors, and $\mathbf{Z} \in \mathbb{R}^{n \times n}$ is an $L-$banded matrix. Let $\mathbf{M} = \text{diag}(\mathbf{Z})$, i.e., the diagonal matrix with the main diagonal of $\mathbf{Z}$ and zero elsewhere. The Jacobi algorithm [7] is given by

$$\mathbf{s}_{k+1} = \mathbf{P}\mathbf{s}_k + \mathbf{M}^{-1}\mathbf{b}, \qquad k \geq 0 \tag{2}$$

where k is the iteration number and the multiplier matrix, $\mathbf{P}$, is

$$\mathbf{P} = \mathbf{M}^{-1}(\mathbf{M} - \mathbf{Z}). \tag{3}$$

Note that since $\mathbf{Z}$ is $L-$banded, the multiplier matrix, $\mathbf{P}$, in (3), is also $L-$banded. It can be shown that the vector $\mathbf{s}$ is the fixed point solution of the iteration in (2). We can extend (2) to solve

$$\mathbf{ZS} = \mathbf{B} \tag{4}$$

for $\mathbf{S}$, where $\mathbf{S} \in \mathbb{R}^{n \times n}$ and $\mathbf{B} \in \mathbb{R}^{n \times n}$ are now matrices, by writing (2) in matrix form as follows.

$$\mathbf{S}_{k+1} = \mathbf{P}\mathbf{S}_k + \mathbf{M}^{-1}\mathbf{B} \tag{5}$$

Solving for,

$$\mathbf{S} = \mathbf{Z}^{-1}, \tag{6}$$

is equivalent to letting $\mathbf{B}$ to be an $n \times n$ identity matrix, $\mathbf{I}$. The iterative algorithm for the inverse of the matrix $\mathbf{Z}$ now becomes,

$$\mathbf{S}_{k+1} = \mathbf{P}\mathbf{S}_k + \mathbf{M}^{-1}. \tag{7}$$

3. DISTRIBUTED JACOBI ALGORITHM

When the matrix $\mathbf{Z}$ is $L-$banded and its local matrices, $\mathbf{Z}^{(l)}$, are distributed among N nodes, as shown in figure 1 (left) , the Jacobi algorithm, (7), can be distributed as follows. The iteration for the ij-th element, s_{ij}, in $\mathbf{S}_{k+1}$ can be written at time $k+1$ as

$$s_{ij} = \mathbf{p}_i \mathbf{s}_k^j \qquad (i \neq j) \tag{8}$$

$$s_{ij} = \mathbf{p}_i \mathbf{s}_k^j + m_i \qquad (i = j) \tag{9}$$

where $\mathbf{p}_i$ is the ith row of the multiplier matrix, $\mathbf{P}$, and $\mathbf{s}_k^j$ is the jth column of the matrix $\mathbf{S}_k$, and m_i is the ith element at the diagonal in $\mathbf{M}^{-1}$. Since, the multiplier matrix, $\mathbf{P}$, is $L-$banded, we note that the only non-zero elements in the ith row, $\mathbf{p}_i$, of $\mathbf{P}$ are located at most at the $i-L, \ldots, i, \ldots, i+L$ locations and can be represented by $\{\mathbf{p}_i\}_q$, where q goes from $i-L, \ldots, i, \ldots, i+L$. These non-zero elements pick the corresponding elements, $\{\mathbf{s}_k^j\}_q$, in the jth column, $\mathbf{s}_k^j$, of $\mathbf{S}_k$, from (8) or (9). Based on the dimensions of the submatrix at the lth node, $\mathbf{S}^{(l)}$, the elements $\{\mathbf{s}_k^j\}_q$ lie in the submatrices $\mathbf{S}^{(l-1)}$, $\mathbf{S}^{(l)}$, and $\mathbf{S}^{(l+1)}$ and hence can be communicated to node l from the neighboring nodes $l-1$ and $l+1$, see figure 2.

Thus, for $L-$banded matrices, $\mathbf{Z}$, the iterative algorithm to find its inverse in (7) can be distributed with communication only from the neighboring nodes. To initialize the algorithm, we have

$$\mathbf{P}^{(l)} = \left(\mathbf{M}^{(l)}\right)^{-1}\left(\mathbf{M}^{(l)} - \mathbf{Z}^{(l)}\right), \tag{10}$$

$$\mathbf{S}_0^{(l)} = \left(\mathbf{Z}^{(l)}\right)^{-1}. \tag{11}$$

Drawbacks for distributed Implementation: Since the inverse, $\mathbf{S}$, of the $L-$banded matrix, $\mathbf{Z}$, is, in general, full, we are required to iterate on the elements that do not lie in the L-band of $\mathbf{S}$ as can be seen from (7). This can be shown by writing out the iteration on the $L-$band element $s_{45,k+1}$ from (8), see figure 2 for $L = 1-$banded matrix, $\mathbf{Z}$,

$$s_{45,k+1} = p_{43}s_{35,k} + p_{44}s_{45,k} + p_{45}s_{55,k}. \tag{12}$$

Equation (12) shows that the iterations on an $L-$band element s_{45} requires a non $L-$band element s_{35}. The iterations on the non $L-$band element s_{35} from (7) can be written as

$$s_{35,k+1} = p_{32}s_{25,k} + p_{33}s_{35,k} + p_{34}s_{45,k}. \tag{13}$$

The computation in (13) involves $s_{25,k}$ that lies in the non $L-$band of $\mathbf{S}_k$, iterating on which, in turn, requires another non $L-$band element, $s_{15,k}$, and so on. Computing the elements outside the $L-$band,

thus, requires iterating on all the elements in a single row of $\mathbf{S}$, at the node corresponding to that row. Hence, a single iteration of the algorithm, although requiring only local communication, sweeps the entire rows in $\mathbf{S}$ at the corresponding nodes and the complexity of this approach scales with the size, n, of the linear system.

4. DISTRIBUTED ITERATE-COLLAPSE INVERSION (DICI) ALGORITHM

To overcome iterating on the entire row of $\mathbf{S}_k$ at the corresponding nodes, we present the DICI algorithm. The algorithm is a $2-$step algorithm with an *iterate step* and a *collapse step*. Before explaining the steps, we present the following theorem for $L = 1$, [5]. Its generalization to $L > 1$ can be found in Theorem 3 in [5].

Theorem 1. *Let $\mathbf{Z}$ be $L = 1-$banded and $\mathbf{S} = \mathbf{Z}^{-1}$. Then any non $L-$band element, s_{ij}, $|i-j| > L$, can be written as a function of the elements inside the $L-$band.*

$$s_{ij} = s_{i,j-1} \left(s_{i+1,j-1}\right)^{-1} s_{i+1,j} \tag{14}$$

If any of the elements in (14) is a non $L-$band element, then it is written first in terms of $L-$banded elements using (14). The DICI algorithm is divided into the following two steps.

Iterate step: The iterate step is implementing (8)–(9) but only for the elements inside the $L-$band of $\mathbf{S}_k$, i.e., ij-th element when $|i-j| \leq L$. Each node l iterates on the elements of its own local matrix, $\mathbf{S}^{(l)}$. To accomplish this, they need elements of $\mathbf{S}^{(l-1)}$ and $\mathbf{S}^{(l+1)}$ and so they are required to communicate with the neighboring nodes (node $l-1$ and node $l+1$).

To start the iterate step, we need the initial conditions that are provided in (10)–(11). We also set the non $L-$band elements required in implementing the iterate step, see (12), to be zero, i.e.,

$$\{s_{ij}\}_{|i-j|>L} = 0. \tag{15}$$

Each lth node performs an initial communication step where the relevant elements of the neighboring multiplier matrices, $\mathbf{P}^{(l-1)}$ and $\mathbf{P}^{(l+1)}$, required for the iterate step are communicated. Note that these elements remain fixed throughout the rest of the algorithm.

Collapse step: The collapse step is to use (14) for the computation of non L-band elements required to implement (8) of the iterate step. When we use equation (14), we are not required to do the entire row sweep, hence distributing the algorithm completely.

Theorem 1 is only valid when $\mathbf{S}$ is the inverse of an $L-$banded matrix. But at each k, $\mathbf{S}_k$ is not the inverse of an $L-$banded matrix, instead it is converging to $\mathbf{S}$, which is the inverse of an $L-$banded matrix. Thus we call this step a collapse step, since it collapses $\mathbf{S}_k$ to the space of matrices having an $L-$banded inverse.

Convergence: We note that the centralized Jacobi algorithm for matrix inversion, in (7), has the error process, $\mathbf{E}_{k+1}$, given by

$$\begin{aligned} \mathbf{E}_{k+1} &= \mathbf{S}_{k+1} - \mathbf{S}, \\ &= \mathbf{P}\mathbf{E}_k. \end{aligned} \tag{16}$$

Thus, the error process goes to 0 if the spectral radius of $\mathbf{P}$ is less than 1, $||\mathbf{P}|| < 1$. Since the distributed Jacobi algorithm has no approximation involved in going from (7) to (8)–(9), the convergence criteria are the same.

Let $\widehat{\mathbf{E}}_{\mathbf{k+1}}$ be the error process of the DICI algorithm. We show the convergence of the DICI algorithm numerically by plotting the difference error process, $\Delta\mathbf{E}_{k+1}$, defined as

$$\Delta\mathbf{E}_{k+1} = \widehat{\mathbf{E}}_{k+1} - \mathbf{E}_{k+1}, \tag{17}$$

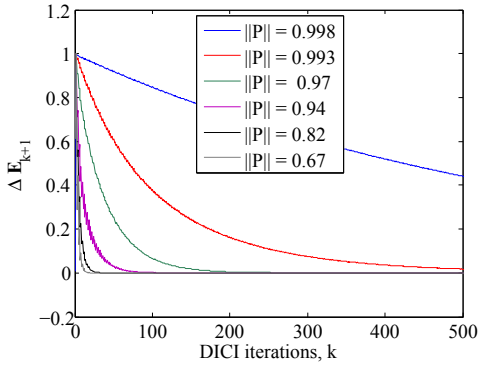


Fig. 3. Trace of $\Delta\mathbf{E}_{k+1}$, plotted versus the DICI iterations for several different $||\mathbf{P}||$ (curves are plotted in the same order from top to bottom as $||\mathbf{P}||$ appears in the legend).

in figure 3. Since $\Delta\mathbf{E}_{k+1}$ goes to zero (we simulated for several values of $||\mathbf{P}||$), from figure 3, the DICI algorithm converges whenever $\mathbf{E}_{k+1}$ goes to zero.

5. APPLICATIONS

We provide two applications for the DICI algorithm below.

5.1. Distributed Kalman filtering

Distributed Kalman filters are very important especially when we have to estimate large $n-$dimensional dynamical systems and their observations are distributed geographically, e.g., US power grid monitored by a sensor network. With n of the order of 10^6 to 10^8 conventional distributed approaches become practically infeasible. A computationally efficient Kalman filter is implemented by spatially decomposing the system into n_l-dimensional local node (or sensor)-driven models, see [9], where $n_l \ll n$. Local Kalman filters of much lower dimension, n_l, are now implemented at each node (sensor) l. Shared observations and estimates across different local models are fused using bipartite fusion graphs, see [6]. For the local prediction step in the local Kalman filters, it can be shown that we are required to compute the local error covariance matrices, $\mathbf{S}^{(l)}$, from the local information matrices, $\mathbf{Z}^{(l)}$, see [6]. The relationship between their centralized counterparts (centralized error covariance matrix, $\mathbf{S}$, and the centralized information matrix, $\mathbf{Z}$) for $n = 5$ is depicted in figure 2, where the information matrices, $\mathbf{Z}$, are approximated to be $L-$banded (equivalent to approximating the Gaussian error process of the Kalman filter to be Gauss-Markovian of Lth order, see [4]). We employ the DICI algorithm to compute $\mathbf{S}^{(l)}$ from $\mathbf{Z}^{(l)}$. The simulation results for the distributed Kalman filter implementation with a $5-$dimensional system and nearest neighbor model dynamics employing DICI algorithm to compute the inverse of the local information matrices is shown in figure 4. Convergence of the DICI algorithm can be verified as the trace of the error covariance matrix in the centralized Kalman filter with $L = 1-$banded approximation on the centralized information matrix can be seen to be exactly overlapping with the trace of the error covariance matrices in the distributed Kalman filter implementation.

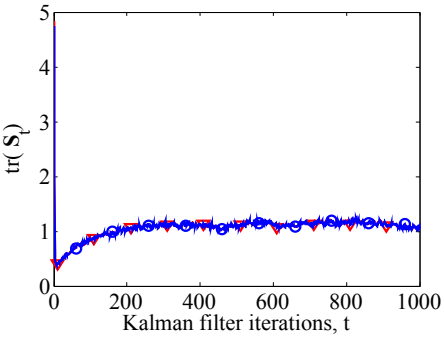
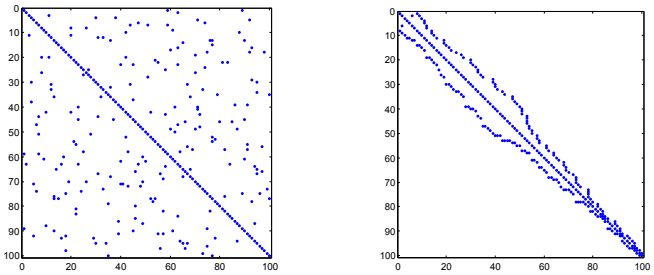


Fig. 4. Kalman filter with $L = 1-$banded approximation on the information matrices, $\mathbf{Z}_t$. Trace of the error covariance matrix, $\mathbf{S}_t$, is plotted for the centralized Kalman filter (-▽-) and the distributed Kalman filter (-◯-).



(a) A random sparse SPD matrix, $\overline{\mathbf{Z}}$, with sparseness density 0.03. Non-zero elements are shown in black.

(b) $L = 12-$banded reordering of $\overline{\mathbf{Z}}$, shown in figure 5(a), using RCM algorithm [2].

Fig. 5. RCM algorithm

5.2. Sparse Matrix Inversion

We show the extension of the DICI algorithm to invert sparse SPD matrices after applying matrix reordering algorithms to the sparse SPD matrices. These algorithms apply matrix bandwidth reduction methods, e.g., Reverse Cuthill Mckee (RCM) algorithm reordering [2], such that the sparse SPD matrices are converted to banded matrices by permutation of rows and columns.

Consider $\overline{\mathbf{Z}}$ to be an arbitrary sparse SPD matrix. We can apply the RCM algorithm to convert $\overline{\mathbf{Z}}$ into an $L-$banded matrix, $\mathbf{Z}$. The general reordering looks like

$$\mathbf{Z} = \mathbf{P}\overline{\mathbf{Z}}\mathbf{P}^T. \tag{18}$$

The inverse of $\overline{\mathbf{Z}}$ is given by

$$\overline{\mathbf{Z}}^{-1} = \mathbf{P}^T\mathbf{Z}^{-1}\mathbf{P}. \tag{19}$$

We can parallelize the computation of $\mathbf{Z}^{-1}$ on a multiprocessor machine using the DICI algorithm and computing $\overline{\mathbf{Z}}^{-1}$ reduces to low order computation at each processor (node) l, and two matrix multiplications. The matrix $\mathbf{P}$ is a permutation matrix and multiplying by it is a permutation of rows (and columns).

Remarks: It may seem that pre- and post-multiplication with the permutation matrix, $\mathbf{P}$, in (19), has to be implemented at a central location. In fact, this step can also be distributed by realizing that the permutation of rows and columns can be implemented by imposing a communication graph on the nodes using the structure of the permutation matrix, $\mathbf{P}$. Hence, $\mathbf{P}$, determines the communication topology required by the nodes to communicate the appropriate elements among the nodes.

We show the result of the RCM algorithm on a 100×100 sparse SPD matrix, $\overline{\mathbf{Z}}$, shown in figure 5(a), which is converted to a $L = 12-$band matrix, $\mathbf{Z}$, shown in figure 5(b), by the permutation matrix given by the RCM algorithm [2]. Depending on the number of nodes, the $L = 12-$banded matrix, $\mathbf{Z}$, shown in figure 5(b), is divided into overlapping local matrices. Note that the minimum size of the local matrix is $L + 1 \times L + 1$, which in the case ($L = 12$) is a 13×13 matrix.

6. CONCLUSIONS

We present a distributed inversion algorithm, DICI, for banded SPD matrices that is distributed both in terms of communication and computations. Each node does local communication and performs computation of order $O(L^4t)$ with only local matrices, where $L << n$ and t is the number of iterations of the DICI algorithm. The algorithm has significant importance when applied to problems where partial information about a global phenomenon is available at the nodes and where parallelized solutions are sought under resource constraints for load balancing.

7. REFERENCES

[1] N. Balram and J. M. F. Moura, "Noncausal Gauss Markov random fields: Parameter structure and estimation," *IEEE Trans. on Information Theory*, vol. 39, no. 4, pp. 1333–1355, Jul. 1993.

[2] E. Cuthill J. McKee, "Reducing the bandwidth of sparse symmetric matrices," in *Proceedings of the 24th National Conference*, New York, 1969, pp. 157–172.

[3] G. Golub and C. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, MD, 1996.

[4] A. Kavcic and J. M. F. Moura, "Matrices with banded inverses: Inversion algorithms and factorization of Gauss-Markov processes," *IEEE Trans. on Information Theory*, vol. 46, no. 4, pp. 1495–1509, Jul. 2000.

[5] A. Asif and J. M. F. Moura, "Inversion of block matrices with L-block banded inverse," *IEEE Trans. on Sig. Proc.*, vol. 53, no. 2, pp. 630–642, Feb. 2005.

[6] U. A. Khan and J. M. F. Moura, "Distributing the Kalman filters for large-scale systems," Submitted to IEEE Trans. on Signal Processing, http://arxiv.org/pdf/0708.0242.

[7] D. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computations*, Prentice Hall, Englewood Cliffs, NJ, 1989.

[8] V. Delouille, R. Neelamani, and R. Baraniuk, "Robust distributed estimation using the embedded subgraphs algorithm," *IEEE Trans. on Sig. Proc.*, vol. 54, pp. 2998–3010, Aug. 2006.

[9] U. A. Khan and J. M. F. Moura, "Model distribution for distributed Kalman filters: A graph theoretic approach," in *41st Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, Nov. 2007, accepted for publication.