# Square Root Propagation

**Andrew G. Howard**
Department of Computer Science
Columbia University
New York, NY 10027
ahoward@cs.columbia.edu

**Tony Jebara**
Department of Computer Science
Columbia University
New York, NY 10027
jebara@cs.columbia.edcu

## Abstract

We propose a message propagation scheme for numerically stable inference in Gaussian graphical models which can otherwise be susceptible to errors caused by finite numerical precision. We adapt square root algorithms, popular in Kalman filtering, to graphs with arbitrary topologies. The method consists of maintaining potentials and generating messages that involve the square root of precision matrices. Combining this with the machinery of the junction tree algorithm leads to an efficient and numerically stable algorithm. Experiments are presented to demonstrate the robustness of the method to numerical errors that can arise in complex learning and inference problems.

## 1 Introduction

The formalism of probabilistic graphical models has led to a unification of methods from diverse fields and has allowed improvements and transfer of many inference tools. For instance, approximate inference techniques from statistical physics such as sampling methods, and variational mean field algorithms have migrated into the learning community. The control literature has also been a source of novel inference tools. Expectation propagation, for example, has its roots in assumed density filtering. In this article, we propose (and generalize) another technique from the control and estimation literature, square-root propagation, which resolves numerical issues that can emerge during inference in Gaussian graphical models. Numerical errors in the ubiquitous Kalman filter model have been well studied for years. The long chain of Gaussian random variables through which messages must be propagated can give rise to numerical inaccuracies in the estimated covariance matrices. The cumulative effects of round off error eventually causes estimated covariances to drift away from symmetry and to possibly develop negative eigenvalues. One solution for this is to propagate only half the values in the covariance in a so called *square root matrix*. This technique was first proposed in [9] and was used in the Apollo manned missions to the moon. Square root algorithms have since become a well understood tool [6] and (arguably) the preferred method for implementing Kalman filters for real world applications. The main contribution of this paper is to generalize the square root algorithm of Kalman filtering to arbitrary Gaussian graphical models through the use of the junction tree algorithm, loopy belief propagation, as well as variational approximations. Of particular interest is the application of square root propagation to the long chains that arise in variational approximation methods applied to hybrid dynamic Bayes nets [8, 4] and variational Bayesian inference

for linear dynamical systems [1]. We present the first square root filter and smoother for these complicated hybrid models and additionally a much simplified square root smoother in comparison to previous algorithms reported in [6].

The paper is organized as follows. Section 2 reviews inference and approximate inference in general graphical models. Section 3 describes various ways to parameterize a multivariate Gaussian and introduces the notation for square root form. Section 4 reviews local message passing computations in Gaussian models. This is the basis for square root propagation which is introduced in section 5. In section 6 experiments demonstrate that square root propagation behaves better than previous message propagation schemes. We conclude in section 7 with a discussion and future work.

## 2 Inference in Graphical Models

In this section we briefly review exact inference in graphical models using the junction tree algorithm. We also discuss approximate inference methods such as loopy belief propagation and variational structured mean field algorithms. These all employ local operations to perform inference in a model. These local operations (also known as message passing or belief propagation) essentially involve *sum-product* steps because they perform marginalization and multiplication of local probabilities or potential functions. We summarize these operations here and then describe them in the special case of Gaussian graphical models and finally go on to develop square root propagation to improve their numerical accuracy.

### 2.1 Junction Tree Algorithm

The junction tree algorithm (JTA) is a key inference tool in the graphical modeling community and essentially involves sum-product operations. In the graphical modeling framework, we are given a joint distribution over a set of random variables $x$. This joint distribution can be written in terms of a product of non-negative potential functions over all cliques $C$ of random variables in $x$ as follows:

$$P(x) = \frac{1}{Z} \prod_{c \in C} \psi_c(x_c).$$

If such cliques are connected in a tree topology which satisfies the so-called running intersection property, the JTA can be called upon to efficiently compute marginal distributions over all the cliques starting from their potential functions $\psi(x_c)$ in two steps. The JTA computes local marginal distributions by passing messages between clique potential functions $\psi(x_c)$ via separator potential functions $\phi(x_i)$. These separator potential functions are maintained between pairs of connected cliques and have a domain $x_i$ which is the set of random variables the pair of cliques have in common. The first step of the junction tree algorithm (the collect step) propagates messages from a downstream clique $c$ (starting at the leaves of a tree) to upstream cliques $c'$ as it progress toward a chosen root. The various clique and separator potential functions are updated along the way as follows:

$$\phi^*(x_i) = \int_{c/x_i} \psi_c(x_c) \qquad \psi^*_{c'}(x_{c'}) = \phi^*(x_i)\psi_{c'}(x_{c'}).$$

Here we integrate (sum) over all variables in a downstream clique $c$ except for $x_i$ to get updated separator potentials and then multiply (product) the separator potential with the potential function of an upstream clique $c'$. Subsequently (during the distribute step of JTA) a return set of messages are sent downstream from the root to the leaves to finish updating the potentials as follows

$$\phi^{**}(x_i) = \int_{c'/x_i} \psi^*_{c'}(x_{c'}) \qquad \psi^{**}_c(x_c) = \frac{\phi^{**}(x_i)}{\phi^*(x_i)} \psi^*_c(x_c).$$

When the junction tree algorithm terminates, the cliques become proportional to true marginals of the variables, in other words, $\psi_c(x_c) \propto P(x_c)$ and $\phi(x_i) \propto P(x_i)$. The computational efficiency of the JTA crucially depends on the complexity of the sum and product operations above which grow exponentially in the clique size (the dimensionality of the random variables in each clique).

### 2.2 Loopy Belief Propagation and Variational Inference

In the case where a inference using a full junction is too costly, an iterative message passing scheme can be used to approximate exact inference. Loopy belief propagation, send messages $\phi^*(x_i)$ iteratively. It uses the same sum and product rules as exact inference but only yields approximate marginals.

A competitor approach to loopy belief propagation is variational inference. This method approximates the true posterior by a more factored distribution $Q$. It procedes my iteratively optimizing $Q$ to minimize its Kullback-Leibler (KL) divergence to the true posterior distribution $KL(Q\|P) = \int_x Q(x) \ln Q(x)/P(x)$. At each iteration, computing exact marginals of $Q$ using the JTA becomes a sub step of each iteration.

## 3 Gaussian Graphical Models

We now focus on the sum-product message passing scheme in the particular case of Gaussian random variables where each $\vec{x}_i$ is a continuous multivariate random variable with a multivariate Gaussian distribution and the overall distribution over all variables $P(\vec{x})$ is a multivariate Gaussian. Message passing (be it for tree, loopy or variational distributions) applies as before in the Gaussian graphical modeling case. However, we note that there are many ways to implement message passing in practice. The actual message passing schemes we will implement depend on the way the Gaussian models are parametrized and we will show that they can have very different numerical properties and numerical performance, particularly in the case where the various Gaussian messages and covariances start to exhibit poor condition numbers.

We assume we have a Gaussian graphical model describing a multivariate random variable $\vec{x}$ that is jointly Gaussian. We will discuss three possible ways of parameterizing the Gaussian (although more are possible). In this article, we consider multivariate Gaussians $P(\vec{x})$ parameterized in the moment parametrization $N(\vec{x}|\mu, \Sigma)$, the canonical parametrization $C(\vec{x}|K, h)$ or the root parametrization $R(\vec{x}|\hat{K}, \hat{h})$. These are theoretically equivalent but will have different numerical behavior. Each is detail below with its parameterization (a vector and a matrix that determine the quadratic form to be exponentiated):

$$\begin{aligned} N(\vec{x}|\mu, \Sigma) &\propto e^{(-\frac{1}{2}(\vec{x}-\mu)^T \Sigma^{-1} (\vec{x}-\mu))} \\ C(\vec{x}|K, h) &\propto e^{(-\frac{1}{2}\vec{x}^T K \vec{x} + h^T \vec{x})} \\ R(\vec{x}|\hat{K}, \hat{h}) &\propto e^{(-\frac{1}{2}\vec{x}^T \hat{K} \hat{K}^T \vec{x} + \hat{h}^T \hat{K}^T \vec{x})}. \end{aligned}$$

It is straightforward to convert between these different parameterizations as follows:

$$h = \Sigma^{-1}\mu \qquad K = \Sigma^{-1} \qquad \hat{K} = K^{1/2} \qquad \hat{h} = \hat{K}^{-1} h.$$

We will assume that these joint distributions obey a Gaussian graphical model and factorize according to a graph $G$ taking on the form of a product of cliques. Message passing takes advantage of this factorization to allow efficient computations of quantities of interest such as marginal and posterior distributions with the overall Gaussian model $P(\vec{x})$. Each message being passed is proportional to a Gaussian model. For each of the above possible Gaussian parameterizations, we can use the message passing scheme of the previous section

as long as we can succinctly enumerate the possible sum-product rules for each message passing. In theory, only one sum-product algorithm for Gaussians is necessary since we can convert back to the desired parametrization once message passing converges or interleave conversions within the message passing. However, such additional conversions in practice incur numerical errors that can accumulate and can even cause divergence. In fact, not all sum-product algorithmic steps are equivalent and we shall demonstrate that the root parameterization and its sum-product procedures for the root parametrization has much better numerical properties.

The conditional probabilities defined by a directed graph in moment form are called linear Gaussians and can be written in moment form as $P(\vec{x}|pa(\vec{x})) = N(\vec{x}|\ b + \sum_{i \in pa(\vec{x})} \beta_i \vec{x}_i, \Sigma)$ where each $\vec{x}$ is a vector and each $\beta$ is a matrix. For propagating messages around the Gaussian graphical model, however, the canonical parameterization which we detail is more elegant. Subsequently we derive square root propagation which maintains this elegance but also exhibits much better numerical properties.

## 4 Canonical Propagation

The canonical form of a Gaussian was defined earlier and is more specifically $C(\vec{x}|K,h) = \exp\left(-\frac{1}{2}\vec{x}^T K \vec{x} + h^T \vec{x} + g\right)$ where $g$ is the normalizer. The canonical form can represent more than just a Gaussian density. In fact, it represents a valid Gaussian density if and only if $K$ is non-singular. We will use canonical potentials as our basic building block for the sum-product procedures in, for example, a junction tree algorithm. When propagating messages in a junction tree made up of canonical potentials only $h$ and $K$ need to be computed because all potentials will be valid Gaussians when the algorithm terminates. The following local computations on potentials are needed to perform inference and are quite easy using the canonical form.

### 4.1 Canonical Product (and Division)

Multiplication of potentials is necessary when updating a potential with a message from its neighbor. With canonical Gaussians, it is simply addition of the canonical parameters.

$$C(\vec{x}|K_1,h_1)C(\vec{x}|K_2,h_2) \ \propto\ C(\vec{x}|K_1+K_2,h_1+h_2).$$

Division similarly involves a subtraction.

### 4.2 Canonical Sum (Marginalization)

To form a message, it is necessary to marginalize a canonical potential as follows:

$$C\left(\begin{bmatrix}\vec{x}_1\\ \vec{x}_2\end{bmatrix} | \begin{bmatrix}K_{11} & K_{12}\\ K_{21} & K_{21}\end{bmatrix}, \begin{bmatrix}h_1\\ h_2\end{bmatrix}\right) \rightarrow C\left(\vec{x}_1|K_{11}-K_{12}K_{22}^{-1}K_{21}, h_1-K_{12}K_{22}^{-1}h_2\right)$$

giving us the parameters of a canonical Gaussian over only the $\vec{x}_1$ random variable.

### 4.3 Canonical Entering of Evidence (Conditioning)

In a Gaussian graphical model, evidence from an observed variable must be entered into every potential in which it appears. Given the joint canonical Gaussian $C(\vec{x}_1,\vec{x}_2)$ above, the conditional potential $C(\vec{x}_1|\vec{x}_2)$ becomes:

$$C\left(\begin{bmatrix}\vec{x}_1\\ \vec{x}_2\end{bmatrix} | \begin{bmatrix}K_{11} & K_{12}\\ K_{21} & K_{21}\end{bmatrix}, \begin{bmatrix}h_1\\ h_2\end{bmatrix}\right) \ \rightarrow\ C\left(\vec{x}_1|K_{11}, h_1-K_{12}\vec{x_2}\right).$$

Thus, we have straightforward manipulations for the canonical parameters for the various building blocks (sum and product and evidence) for message passing in Gaussian graphical models. We next mirror these derivations for the square root Gaussian parametrization.

## 5 Square Root Propagation

This section details square root algorithms to address the numerical problems that can accrue and propagate in large Gaussian graphical networks, namely errors in the propagated covariance matrix i.e. asymmetry and non positive definiteness. Square root algorithms were originally proposed for a special case of Gaussian graphical models, the Kalman filter or state space model. The idea is to only propagate half of the covariance matrix as a so-called square root matrix. The form that these types of algorithms take is to create a matrix of known quantities and apply a unitary transformation. The resulting matrix will then contain the answer. The transformations that are used induce zeros in the solution matrix and can be found by using an extended QR decomposition or, to leverage already existing zero patterns, Givens rotations. A square root Gaussian potential , $R(\vec{x}|K^{1/2}, \hat{h})$, will be defined as in section 3. The sum-product sub algorithms will mirror those of the previous sections except that the local computations will be computed differently and will maintain better numerical behavior.

### 5.1 Root Product (and Division)

We wish to update a single root Gaussian potential via the product of two known root Gaussian potentials as follows:

$$R(\vec{x}|\hat{K}_1, \hat{h}_1)R(\vec{x}|\hat{K}_2, \hat{h}_2) \ \propto \ R(\vec{x}|\hat{K}, \hat{h})$$

Thus, $\hat{K}$ is computed (without conversion) via a numerically reliable linear algebra procedure. This is done by creating a matrix of known values and inducing zero patterns to create the target solution matrix. The answer can then be found in the new matrix.

$$\begin{bmatrix} \hat{K}_1 & \hat{K}_2 \\ \hat{h}_1 & \hat{h}_2 \end{bmatrix} \theta = \begin{bmatrix} \hat{K} & 0 \\ \hat{h} & \alpha \end{bmatrix}.$$

To perform addition we zero out the appropriate entries on the left hand equation to find the matrix on the right by either explicitly or implicitly finding a unitary transformation $\theta$. To perform subtraction (division), instead of a standard unitary transformation, we find a J-unitary transformation where $\theta_J \theta_J^T = J$ and $J$ is an identity matrix with some of the entries on the diagonal are $-1$. A hyperbolic QR decomposition or Cholesky downdate algorithm can be used. See [7] for a thorough treatment.

### 5.2 Root Sum (Marginalization)

To form a message, it is necessary to marginalize a root potential as follows:

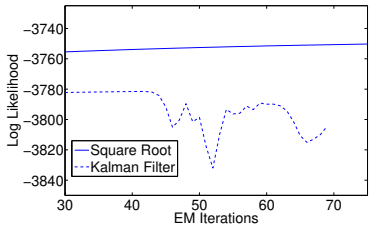
$$R\left(\begin{bmatrix} \vec{x}_1 \\ \vec{x}_2 \end{bmatrix} | \begin{bmatrix} \hat{K}_{11} & K_{12} \\ K_{21} & \hat{K}_{22} \end{bmatrix}, \begin{bmatrix} \hat{h}_1 \\ \hat{h}_2 \end{bmatrix}\right) \ \rightarrow \ R\left(\vec{x}_1 | \hat{K}_1^m, \hat{h}_1^m\right)$$

where the parameters of the marginal root Gaussian over only the $\vec{x}_1$ random variable $\hat{K}$ and $\hat{h}$ are computed as follows. First we solve a triangular system of equations for the term G = $\hat{K}_{12}\hat{K}_{22}^{-1}$ which is necessary for future computations.

$$solve : G\hat{K}_{22} = K_{12}, \qquad \begin{bmatrix} \hat{K}_{11} & G \\ \hat{h}_1 & \hat{h}_2 \end{bmatrix} \theta_J = \begin{bmatrix} \hat{K}_1^m & 0 \\ \hat{h}_1^m & \alpha \end{bmatrix}$$

### 5.3 Root Entering of Evidence (Conditioning)

In the square root parameterization, entering evidence simply requires solving a triangular system of equations.




| | Diverged | $\%\lambda < 0$ | $\|\|P_t - P_t^T\|\|_F^2$ |
|---|---|---|---|
| KF | 38 | 1.21% | 2.32 |
| IF | 100 | – | – |
| SR | 0 | 0% | $7.89 \cdot 10^{-20}$ |

Figure 1: a) The training log likelihood when the Kalman filtering is one step from divergence b) Results for EM training

$$R\left(\begin{bmatrix} \vec{x}_1 \\ \vec{x}_2 \end{bmatrix} | \begin{bmatrix} \hat{K}_{11} & K_{12} \\ K_{21} & \hat{K}_{22} \end{bmatrix}, \begin{bmatrix} \hat{h}_1 \\ \hat{h}_2 \end{bmatrix}\right) \rightarrow R\left(\vec{x}_1 | \hat{K}_1^c, \hat{h}_1^c\right)$$

$$solve: \hat{K}_{11}\hat{h}_1^c = K_{12}\vec{x}_2 \qquad \hat{K}_1^c = \hat{K}_{11}$$

# 6 Experiments

We now demonstrate the robustness of square root propagation in the context of learning and inference compared with other Gaussian propagation methods. Experiments for learning parameters of dynamical systems as well as approximate inference using structured mean field are shown and provide evidence for the usefulness of square root propagation.

## 6.1 Learning with EM

In this experiment we demonstrate the effect numerical errors can have on the inference step of an Expectation-Maximization (EM) algorithm and on the overall maximum likelihood parameter estimation problem. We will learn a linear dynamical system using EM from simulated data. The linear dynamical system model is defined as follows

$$\vec{x}_t = A\vec{x}_{t-1} + \vec{w}_t \qquad \vec{y}_t = C\vec{x}_t + \vec{v}_t$$

We choose a very simple model to learn with $\vec{x}_t$ and $\vec{y}_t$ two dimensional, $A$ and $C$ identity transformations, and $\vec{w}_t \sim N(0, Q)$ and $\vec{v}_t \sim N(0, R)$ with $Q$ and $R$ as identity. The model was simulated for 1000 times steps with an initial state, $\vec{x}_1 \sim N(0, I)$. To create a situation in which we would expect to find an ill conditioned covariance matrix, we propose to learn the linear dynamical system model [2] with a state space of 8 dimensions. The experiments were run 100 times with a different set of samples. We tried 3 different algorithms for the inference problem, the standard Kalman filter and RTS smoother using a moment parameterization, a potential based information filter based on the junction tree algorithm using canonical parameterization, and square root propagation using the root Gaussian parameterization. Square root propagation is the square root form of the information filter used in the experiment allowing for direct comparison of the effects of the square root form. The Kalman filter implementation was from Kevin Murphy's Kalman filter toolbox allowing for comparison to a widely used implementation. In the experiment we run EM for 100 steps or until convergence.

Figure 1b) summarizes our results. Column one shows the number of tests that an algorithm diverged (NaN errors caused by numerics). Figure 1a) shows the training log likelihood of

| $N = \lambda_{max}/\lambda_{min}$ | $10^6$ | $10^7$ | $10^8$ | $10^9$ | $10^{10}$ | $10^{11}$ |
|---|---|---|---|---|---|---|
| Moment | 0 | 0 | 0 | 36 | 83 | 93 |
| Canonical | 0 | 27 | 77 | 95 | 98 | 92 |
| Square Root | 0 | 0 | 0 | 15 | 38 | 32 |

Figure 2: Divergence in SLDS inference as condition number increases

the Kalman filter one step before divergence. Column two of figure 1b) lists the percentage of eigenvalues that are negative in the the inferred covariance matrices, $P_t$, at the completion of EM for the cases that do converge. Furthermore, in column three we report a measure in the divergence from symmetry in all $P_t$, the average of $\|P_t - P_t^T\|_F^2$, where $\|\cdot\|_F^2$ is the squared Frobenius norm. Theoretically, the log likelihood should converge monotonically, $P_t$ should never have negative eigenvalues and $\|P_t - P_t^T\|_F^2 = 0$. However, in our experiments these were violated by the algorithms. The information filter performed worst, diverging in all cases. This poor performance was due to having to switch between moment parameters to update the M-step and canonical parameters to perform inference. The Kalman filter which computed all steps in moment form diverged 38 times, and had negative eigenvalues and errors in symmetry when it did perform properly. Square root propagation converged in all tests. However, in 10 cases, it did not converge monotonically. The other algorithms though, diverged in all those cases. The eigenvalues and symmetry were correctly preserved in the square root case.

### 6.2 Variational inference in hybrid dynamic Bayes nets

In this experiment we demonstrate square root propagation in a complex inference task. A popular method for performing approximate inference in hybrid[8, 4] and fully Bayesian[1] dynamic Bayes nets is variational structured mean field [5]. In typical setups, part of the approximate variational distribution will be chosen to be a Gaussian chain. It can be seen that this distribution forms canonical potentials that factor over a chain.

We will be working with a switching linear dynamical system defined as follows:

$$\vec{x}_t = A_i \vec{x}_{t-1} + \vec{w}_t \qquad \vec{y}_t = C\vec{x}_t + \vec{v}_t$$

Where a discrete variable $s_t$ has been introduced to switch the dynamics and noise model of the state space. $s_t$ evolves according to a Markov chain with parameters $\pi(i) = p(s_1 = i)$ and $\Pi(i,j) = P(s_t = i | s_{t-1} = j)$. $\vec{v}_t \sim N(0, R)$ as before and $\vec{w}_t \sim N(0, Q_i)$ depending on the state of $s_t$.

Variational inference in this model alternates between performing inference in the discrete chain and the Gaussian chain until convergence. For our experiment, we randomly generated SLDS models with 2 dimensional $\vec{x}_t$ and $\vec{y}_t$ and 2 switching states. The state space covariance $Q$ was created as a full covariance that was ill conditioned. In the experiment we varied the condition number, $N = \lambda_{max}/\lambda_{min}$, of $Q$ from $10^{-6}$ to $10^{-11}$ by setting the smallest eigenvalue, $\lambda_{min} = 1/N$ and $\lambda_{max} = 1$. Three algorithms were tested for computing marginals in the Gaussian chain, a moment based algorithm proposed in [4] as an improvement over the algorithm in [8], a canonical form junction tree algorithm similar to the algorithm proposed in [1], and a square root version of the canonical algorithm. The variational inference was iterated 25 times in the experiment.

Figure 2 show the results for the various algorithms at increasing condition numbers. An algorithm is considered to have diverged if it reaches 25 iterations with imaginary or non monotonic likelihood. The canonical based algorithm again suffers from having to switch between parameterization. The updates to the discrete chain require a conversion to moment parameters. The moment based algorithm holds up better but starts to break down

as the condition number increases. The square root algorithm does diverge as condition number increases, but not nearly as much as the other two algorithms.

## 7 Discussion

We have proposed square root propagation as a numerically robust message passing scheme for inference in arbitrary Gaussian graphical models. In the experiments we demonstrated improved numerical stability in complicated problems. Another area that we will be applying square root propagation is for loopy belief propagation in Gaussian Markov random fields (GMRFs). Loopy belief propagation has been provably shown to converge under some cases yielding exact means in a GMRF [11]. Square root propagation would be useful to guarantee that a lack of convergence is not caused by numerical errors.

We briefly discuss the numerical properties of square root propagation, the running time and numerical stability. The running time of square root propagation is asymptotically equivalent to canonical propagation. For each algorithm a message takes $O(n^3)$ flops in total. However, the product of potentials substep of message passing is more expensive in square root propagation at $O(n^3)$ as opposed to $O(n^2)$ for matrix addition in canonical propagation. Each of the numerical linear algebra algorithms used in square root propagation have numerical stability results. QR decomposition, Givens rotations, and solving linear equations with back substitution are backward stable [3]. Hyperbolic QR is known to have algorithms that are mixed forward-backward stable, and in some cases backward stable [7]. It has also been shown that a sequence of Cholesky updates and downdates are mixed forward-backward stable [10]. This result, if generalized to interleaving a solution of a system of triangular equations in the sequence could be used to show the mixed forward-backward stability of our algorithm. This will be the subject of future work.

## References


[1] M. J. Beal and Z. Ghahramani. The variational kalman smoother. Technical report, Gatsby Computational Neuroscience Unit, 2001. GCNU TR 2001-03.

[2] G.E. Ghahramani, Z.and Hinton. Parameter estimation for linear dynamical systems. Technical report, Univsersity of Toronto, 1996. CRG-TR-96-2.

[3] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, second edition, 2002.

[4] A. Howard and T. Jebara. Dynamical systems trees. In *In Proc. 20th conf. on Uncertainty in Artificial Intelligence*, 2004.

[5] T. Jaakkola. Tutorial on variational approximation methods. In *Advanced Mean Field Methods: Theory and Practice*. MIT Press, 2000.

[6] T. Kailath, A. H. Sayed, and B. Hassibi. *Linear Estimation*. Prentice Hall, Upper Saddle River, NJ, first edition, 2000.

[7] H. Patel. *Solving the Indefinite Least Squares Problem*. PhD thesis, University of Manchester, 2002.

[8] V. Pavlovic, J. M. Rehg, and J. MacCormick. Learning switching linear models of human motion. In *Neural Information Processing Systems 13*, pages 981–987, 2000.

[9] J.E. Potter and R.G. Stern. Statistical filtering of space navigation measurements. In *AIAA Guidance Contr. Conference*, 1963.

[10] G. W. Stewart. On the stability of sequential updates and downdates. *IEEE Transactions on Signal Processing*, 43(11):2642–2648, 1995.

[11] Y. Weiss and W. T. T. Freeman. Correctness of belief propagation in gaussian graphical models of arbitrary topology. *Neural Computation*, 13(10):2173–2200, 2001.