Journal of Machine Learning Research 20 (2019) 1-37 Submitted 1/18; Revised 12/19; Published 12/19

On the Convergence of Gaussian Belief Propagation with Nodes of Arbitrary Size

Francois Kamper FRANCOISK@SUN.AC.ZA
Sarel J. Steel SJST@SUN.AC.ZA
Department of Statistics and Actuarial Science
Stellenbosch University
Stellenbosch, South Africa

Johan A. du Preez DUPREEZ@SUN.AC.ZA
Department of Electrical and Electronic Engineering
Stellenbosch University
Stellenbosch, South Africa

Editor: David Barber

Abstract

This paper is concerned with a multivariate extension of Gaussian message passing applied to pairwise Markov graphs (MGs). Gaussian message passing applied to pairwise MGs is often labeled Gaussian belief propagation (GaBP) and can be used to approximate the marginal of each variable contained in the pairwise MG. We propose a multivariate extension of GaBP (we label this GaBP-m) that can be used to estimate higher-dimensional marginals. Beyond the ability to estimate higher-dimensional marginals, GaBP-m exhibits better convergence behavior than GaBP, and can also provide more accurate univariate marginals. The theoretical results of this paper are based on an extension of the computation tree analysis conducted on univariate nodes to the multivariate case. The main contribution of this paper is the development of a convergence condition for GaBP-m that moves beyond the walk-summability of the precision matrix. Based on this convergence condition, we derived an upper bound for the number of iterations required for convergence of the GaBP-m algorithm. An upper bound on the dissimilarity between the approximate and exact marginal covariance matrices was established. We argue that GaBP-m is robust towards a certain change in variables, a property not shared by iterative solvers of linear systems, such as the conjugate gradient (CG) and preconditioned conjugate gradient (PCG) methods. The advantages of using GaBP-m over GaBP are also illustrated empirically.

Keywords: belief propagation, Gaussian distributions, higher-dimensional marginals, preconditioning, inference

1. Introduction

Let $\mathbf{X}_i : d_i \times 1$ for $i = 1, 2, \ldots, p$ be mutually exclusive and exhaustive subvectors of the random vector $\mathbf{X} : k \times 1$. We are concerned with the approximation of the marginal distributions of $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_p$ using belief propagation (BP), where it is assumed that $\mathbf{X}$ follows a Gaussian distribution with precision matrix $\mathbf{S}$ and potential vector $\boldsymbol{b}$.

BP is an approximate marginalization algorithm that operates on a specified graph struc-

ture. The origin of BP can be traced back to the sum-product algorithm as a decoding algorithm for LDPC codes (Gallager, 1963). When the underlying graph structure forms a tree, BP will provide the correct marginals at convergence. This is in contrast to loopy graphs, where convergence is not guaranteed and, even if convergence occurs, the marginals supplied are not necessarily correct (Pearl, 1988; Weiss and Freeman, 2001). Although this seems to be a damning property, BP has become a popular tool in inference problems in situations in which inference does not necessarily require exact marginals (see, for instance, the near Shannon-limit performance of BP in turbo codes) and because of ease of implementation in distributive settings.

GaBP is BP applied to a pairwise MG constructed from a multivariate Gaussian distribution in canonical parameterization. GaBP can be used to approximate the univariate marginals of the variables represented by the MG. GaBP is guaranteed to converge if the pairwise MG forms a tree structure, but does not necessarily converge for loopy pairwise MGs. The convergence behavior of GaBP has been the subject of research in the literature (Weiss and Freeman, 2001; Malioutov et al., 2006; Su and Wu, 2015; Sui et al., 2015). Weiss and Freeman (2001) interpret the computations done by GaBP as inference on a tree-structured graph. This interpretation is used to show that univariate GaBP will converge if the precision matrix is diagonally dominant and that the marginal means supplied by GaBP, assuming convergence, are exact. We now define two important concepts related to the convergence of GaBP:

Definition 1 (Walk-summability) *Consider a precision matrix* $\mathbf{S} : k \times k = [s_{ij}]$*, and suppose* $\mathbf{D} = diag(\frac{1}{\sqrt{s_{11}}}, \frac{1}{\sqrt{s_{22}}}, \ldots, \frac{1}{\sqrt{s_{kk}}})$*. The matrix* $\mathbf{S}$ *is considered to be walk-summable if the spectral radius of* $|\mathbf{I}_k - \mathbf{DSD}|$ *is less than one.*

Definition 2 (Scaled diagonal dominance) *A precision matrix* $\mathbf{S}$ *is called scaled diagonally dominant if there exists a diagonal matrix* $\mathbf{D}$*, with only positive diagonal entries, such that* $\mathbf{DSD}$ *is strictly diagonally dominant.*

If $\mathbf{A} = [a_{ij}]$, then by $|\mathbf{A}|$ we mean $[|a_{ij}|]$. Malioutov et al. (2006) show that GaBP will converge if the precision matrix is walk-summable, while Moallemi and Van Roy (2009, 2010) show the same for scaled diagonal dominance. In fact, these two conditions are known to be equivalent (Malioutov, 2008; Ruozzi et al., 2009). Su and Wu (2015) derive necessary and sufficient conditions for the convergence of univariate GaBP under a specified initialization set. They also derive necessary and sufficient conditions for damped univariate GaBP, with an allowable interval for the damping factor.

Although technically a marginalization algorithm, GaBP has been applied to solve large and sparse systems of linear equations (Bickson, 2008; Shental et al., 2008; El-Kurdi et al., 2012b). GaBP has also been applied in areas such as channel estimation in communication systems (Montanari et al., 2006; Guo and Ping, 2008; Guo and Huang, 2011), sparse Bayesian learning in large-scale compressed sensing problems (Seeger and Wipf, 2010), estimation on Gaussian graphical models (Chandrasekaran et al., 2008; Liu et al., 2012) and the detection of F-formations in free-standing conversational groups (Kamper, 2017).

Several variants of GaBP have been proposed in the literature to combat the convergence issues of the basic algorithm in loopy graphs (Johnson et al., 2009; Liu, 2010; El-Kurdi et al., 2012a; Ruozzi and Tatikonda, 2013; Liu et al., 2012; Kamper et al., 2018). Johnson et al. (2009) employ diagonal loadings on the precision matrix and iteratively apply GaBP to compute the correct marginal means. A feedback node refers to a node which, when removed, results in a cycle-free graph. The role of (pseudo) feedback nodes in improving the convergence of basic GaBP has been studied by Liu (2010) and Liu et al. (2012). El-Kurdi et al. (2012a) use a relaxation factor to accelerate GaBP in cases where the precision components of the basic algorithm converge, while the use of a reweighted min-sum algorithm has been studied by Ruozzi and Tatikonda (2013). Kamper et al. (2018) specify a regularization scheme on GaBP that guarantees convergence to the exact marginal means, given sufficient regularization. They also show empirically that this regularization scheme can improve the accuracy of the precisions supplied by GaBP as approximations for the exact marginal precisions.

In this paper we consider a multivariate extension of GaBP that can be used to approximate the marginals of $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_p$. For this purpose, we consider a higher-dimensional generalization of a pairwise MG, where each node is allowed to receive more than one random variable. We are not aware of any research that explicitly studies the use of a higher-dimensional MG to approximate higher-dimensional marginals, although related work, on distributed state estimation using Gaussian message-passing, can be found in the literature (Sui et al., 2015). The main contribution of this paper is the development of a sufficient condition for the convergence of GaBP-m. This condition, labeled preconditioned walk-summability, is a multivariate extension of the walk-summability condition for the convergence of GaBP. Under the assumption of preconditioned walk-summability, we derive a bound for the number of iterations required for convergence by GaBP-m. We also establish a bound on the dissimilarity between the approximate and exact marginal covariance matrices. Another important advantage of GaBP-m is its robustness with regard to a certain scaling of the precision matrix and potential vector. This robustness is special, since it can have a substantial effect on the performance of iterative solvers of linear systems such as the CG solver.

The main advantage of GaBP-m over GaBP is in a marginalization context, since GaBP-m can be used to approximate the precision matrix of higher-dimensional marginals. Other advantages include:

1. GaBP-m can converge in cases where GaBP does not.

2. GaBP-m can converge in fewer iterations than GaBP.

3. GaBP-m can improve the accuracy of the approximate univariate marginals. This can be done by directly computing the univariate marginals from the approximated higher-dimensional marginals.

The advantage of using GaBP over GaBP-m is that the message updates associated with each iteration are computationally less expensive.

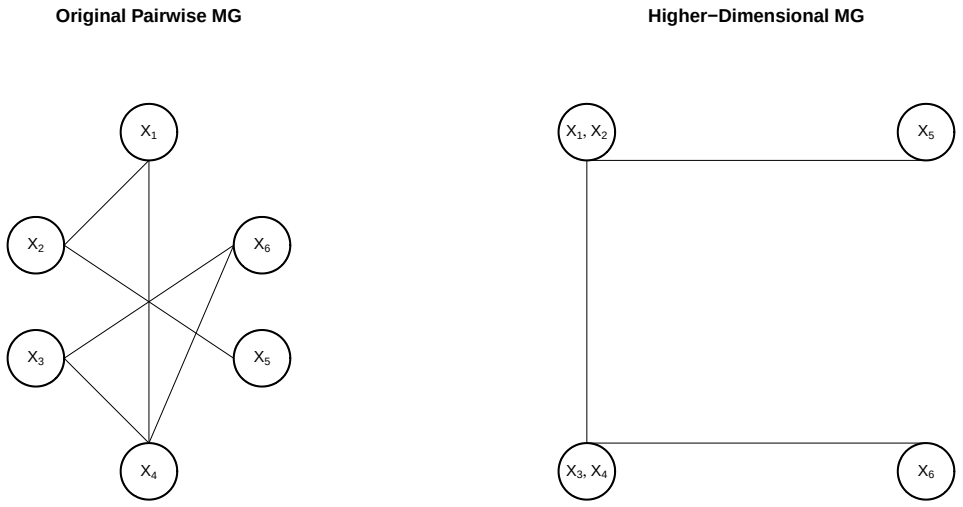

Figure 1: Example of the conversion of a pairwise MG to a higher-dimensional MG. We have six variables $(X_1, X_2, \ldots, X_6)$ clustered into $\mathcal{C}_1 = \{1, 2\}$, $\mathcal{C}_2 = \{3, 4\}$, $\mathcal{C}_3 = \{5\}$ and $\mathcal{C}_4 = \{6\}$. The original pairwise MG is given to the left, while the higher-dimensional extension is given to the right.

2. Preliminaries

In this section we discuss the mathematical machinery that is used in this paper.

2.1. Higher-dimensional MG

We assume, without loss of generality, that $\mathbf{X} = (\mathbf{X}_1', \mathbf{X}_2', \ldots, \mathbf{X}_p')'$. Let $\mathcal{C}_i$ be the set containing all the variables in $\mathbf{X}_i$. We sometimes refer to $\mathcal{C}_i$ as cluster i. Define $\tilde{\mathcal{G}} = (\tilde{\mathcal{V}}, \tilde{\mathcal{E}})$ to be the pairwise MG for $\mathbf{X}$, where $\tilde{\mathcal{V}}$ and $\tilde{\mathcal{E}}$ denote the set of nodes and the set of edges respectively. Note that, when $(s, t) \in \tilde{\mathcal{E}}$, we place the restriction that $s < t$. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the higher-dimensional MG obtained from the clusters $\mathcal{C}_i : i = 1, 2, \ldots, p$. Note that $\mathcal{V}$ will contain a node for each $\mathcal{C}_i : i = 1, 2, \ldots, p$. The pair $(i, j) \in \mathcal{E}$, $i < j$, if and only if there is an $s \in \mathcal{C}_i$ and a $t \in \mathcal{C}_j$ such that either (s, t) or (t, s) is in $\tilde{\mathcal{E}}$.

We illustrate this procedure in Figure 1. We consider a pairwise MG drawn for $\mathbf{X} = (X_1, X_2, X_3, X_4, X_5, X_6)'$ where we perform the clustering $\mathcal{C}_1 = \{1, 2\}$, $\mathcal{C}_2 = \{3, 4\}$, $\mathcal{C}_3 = \{5\}$ and $\mathcal{C}_4 = \{6\}$. From this point onwards, we refer to the higher-dimensional extension of the pairwise MG just as MG.

Algorithm 1 Synchronous GaBP-m

1. Provide a precision matrix $\mathbf{S} : k \times k$, a potential vector $\boldsymbol{b} : k \times 1$, and clusters $\mathcal{C}_i : i = 1, 2, \ldots, p$ as inputs.
2. Specify a tolerance ϵ and a maximum number of iterations m.
3. Initialize $\mathbf{Q}_{ij}^{(0)} = \mathbf{0} : d_j \times d_j$ and $\mathbf{v}_{ij}^{(0)} = \mathbf{0} : d_j \times 1$ for all i and all $j \in \mathcal{N}_i$.
4. Set Err = Inf and $n = 0$.
5. While Err $> \epsilon$
   (a) Compute $\mathbf{P}_i^{(n)} = \mathbf{S}_{ii} + \sum_{j \in \mathcal{N}_i} \mathbf{Q}_{ji}^{(n)}$ and $\mathbf{z}_i^{(n)} = \boldsymbol{b}_i + \sum_{j \in \mathcal{N}_i} \mathbf{v}_{ji}^{(n)}$ for $i = 1, 2, \ldots, p$.
   (b) Set $\boldsymbol{\mu}_i^{(n)} = [\mathbf{P}_i^{(n)}]^{-1} \mathbf{z}_i^{(n)}$, $\boldsymbol{e}_i^{(n)} = \sum_j \mathbf{S}_{ij} \boldsymbol{\mu}_j^{(n)} - \boldsymbol{b}_i$ and Err $= \max_i \{||\boldsymbol{e}_i^{(n)}||_\infty\}$.
   (c) If Err $> \epsilon$, do for all $i \in \{1, 2, \ldots, p\}$ and all $j \in \mathcal{N}_i$:
   $\mathbf{Q}_{ij}^{(n+1)} = -\mathbf{S}_{ji} [\mathbf{P}_i^{(n)} - \mathbf{Q}_{ji}^{(n)}]^{-1} \mathbf{S}_{ij}$ and
   $\mathbf{v}_{ij}^{(n+1)} = -\mathbf{S}_{ji} [\mathbf{P}_i^{(n)} - \mathbf{Q}_{ji}^{(n)}]^{-1} [\mathbf{z}_i^{(n)} - \mathbf{v}_{ji}^{(n)}]$.
   (d) Increment n.
   (e) If $n = m$, break.
6. End.

2.2. Factorization of the Gaussian Density

We assume that $\mathbf{X}$ follows a multivariate Gaussian distribution with precision matrix $\mathbf{S}$ and potential vector $\boldsymbol{b}$. By $\mathbf{S}_{ij}$ we refer to the submatrix of $\mathbf{S}$ corresponding to the variables in $\mathcal{C}_i$ and $\mathcal{C}_j$ for the rows and columns respectively. The vector $\boldsymbol{b}_i$ refers to the subvector of $\boldsymbol{b}$ corresponding to the variables in $\mathcal{C}_i$.

Note that we can factor the Gaussian density according to the MG:

$$f(\mathbf{x}) = \frac{1}{Z} \prod_{i=1}^{p} \phi_i(\mathbf{x}_i) \prod_{(i,j) \in \mathcal{E}} \psi_{ij}(\mathbf{x}_i, \mathbf{x}_j), \tag{1}$$

where $\phi_i(\mathbf{x}_i) = \exp\left[-\frac{1}{2} \mathbf{x}_i' \mathbf{S}_{ii} \mathbf{x}_i + \boldsymbol{x}_i' \boldsymbol{b}_i \right]$, $\psi_{ij}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left[-\mathbf{x}_i' \mathbf{S}_{ij} \mathbf{x}_j \right]$, Z is a normalization constant, $\mathbf{x} = (\mathbf{x}_1', \mathbf{x}_2', \ldots, \mathbf{x}_p')'$ follows the decomposition $\mathbf{X} = (\mathbf{X}_1', \mathbf{X}_2', \ldots, \mathbf{X}_p')'$, and $\mathcal{E} = \{(i,j) : i < j \text{ and } \mathbf{S}_{ij} \neq \mathbf{0} : d_i \times d_j\}$.

2.3. Derivation of GaBP-m

We restrict our focus to a synchronous sum-product derivation of GaBP-m. Define $\mathcal{N}_i = \{j : (i,j) \text{ or } (j,i) \in \mathcal{E}\}$ to denote the neighborhood of cluster i. In the Gaussian case, we have $\mathcal{N}_i = \{j \neq i : \mathbf{S}_{ij} \neq \mathbf{0} : d_i \times d_j\}$. Suppose that $j \in \mathcal{N}_i$, then by $\mathcal{N}_i \setminus j$ we mean the set $\mathcal{N}_i$ with cluster j removed. Based on Equation (1), we formulate the message updates

of GaBP-m analogous to the message updates of BP applied to a pairwise MG:

$$m_{ij}^{(n+1)}(\mathbf{x}_j) = \int_{\mathbf{x}_i} \phi_i(\mathbf{x}_i)\psi_{ij}(\mathbf{x}_i, \mathbf{x}_j) \prod_{t \in \mathcal{N}_i \setminus j} m_{ti}^{(n)}(\mathbf{x}_i) d\mathbf{x}_i, \tag{2}$$

for all clusters i and all $j \in \mathcal{N}_i$. Note that when $i > j$ we use $\psi_{ij}(\mathbf{x}_i, \mathbf{x}_j) = \psi_{ji}(\mathbf{x}_j, \mathbf{x}_i)$. For the Gaussian case, Equation (2) reduces to:

$$m_{ij}^{(n+1)}(\mathbf{x}_j) = \int_{\mathbf{x}_i} \exp\Big[-\frac{1}{2}\mathbf{x}_i'\mathbf{S}_{ii}\mathbf{x}_i + \mathbf{x}_i'\boldsymbol{b}_i \Big] \exp\Big[-\mathbf{x}_i'\mathbf{S}_{ij}\mathbf{x}_j \Big] \prod_{t \in \mathcal{N}_i \setminus j} m_{ti}^{(n)}(\mathbf{x}_i) d\mathbf{x}_i. \tag{3}$$

We now make the assumption that $m_{ti}^{(n)}(\mathbf{x}_i) \propto \exp\Big[-\frac{1}{2}\mathbf{x}_i'\mathbf{Q}_{ti}^{(n)}\mathbf{x}_i + \mathbf{x}_i'\mathbf{v}_{ti}^{(n)} \Big]$ for all $t \in \mathcal{N}_i$. Substitution into Equation (3) yields:

$$m_{ij}^{(n+1)}(\mathbf{x}_j) \propto \exp\Big[-\frac{1}{2}\mathbf{x}_j'\mathbf{Q}_{ij}^{(n+1)}\mathbf{x}_j + \mathbf{x}_j'\mathbf{v}_{ij}^{(n+1)} \Big], \tag{4}$$

where

$$\mathbf{Q}_{ij}^{(n+1)} = -\mathbf{S}_{ji}[\mathbf{P}_{ij}^{(n)}]^{-1}\mathbf{S}_{ij} \tag{5}$$

$$\mathbf{v}_{ij}^{(n+1)} = -\mathbf{S}_{ji}[\mathbf{P}_{ij}^{(n)}]^{-1}[\boldsymbol{b}_i + \sum_{t \in \mathcal{N}_i \setminus j} \mathbf{v}_{ti}^{(n)}] \tag{6}$$

$$\mathbf{P}_{ij}^{(n)} = \mathbf{S}_{ii} + \sum_{t \in \mathcal{N}_i \setminus j} \mathbf{Q}_{ti}^{(n)}. \tag{7}$$

Note that we used the following integral in the derivation of Equation (4):

$$\int_{\mathbf{y}} \exp\Big(-\frac{1}{2}\mathbf{y}'\mathbf{A}\mathbf{y} + \mathbf{y}'\boldsymbol{a} \Big) d\mathbf{y} = (2\pi)^{\frac{m}{2}} (\det(\mathbf{A}))^{-\frac{1}{2}} \exp\Big(\frac{1}{2}\boldsymbol{a}'\mathbf{A}^{-1}\boldsymbol{a} \Big),$$

where $\mathbf{A}$ is positive definite. The message updates in Equations (5) and (6) are performed efficiently in Algorithm 1. At each iteration of GaBP-m, we can construct an approximate marginal for cluster i by collecting all incoming messages from its neighborhood:

$$\hat{f}_i^{(n)}(\mathbf{x}_i) = \phi_i(\mathbf{x}_i) \prod_{t \in \mathcal{N}_i} m_{ti}^{(n)}(\mathbf{x}_i) \propto \exp\Big[-\frac{1}{2}\mathbf{x}_i'\mathbf{P}_i^{(n)}\mathbf{x}_i + \mathbf{x}_i'\mathbf{z}_i^{(n)} \Big], \tag{8}$$

where $\mathbf{P}_i^{(n)} = \mathbf{S}_{ii} + \sum_{j \in \mathcal{N}_i} \mathbf{Q}_{ji}^{(n)}$ and $\mathbf{z}_i^{(n)} = \boldsymbol{b}_i + \sum_{j \in \mathcal{N}_i} \mathbf{v}_{ji}^{(n)}$. Clearly, the density given in Equation (8) is a Gaussian density with precision $\mathbf{P}_i^{(n)}$ and mean $\boldsymbol{\mu}_i^{(n)} = [\mathbf{P}_i^{(n)}]^{-1}\mathbf{z}_i^{(n)}$. We refer to $\hat{f}_i^{(n)}(\mathbf{x}_i)$ as the posterior distribution of cluster i at iteration n. The quantities $\mathbf{P}_i^{(n)}$ and $\boldsymbol{\mu}_i^{(n)}$ are labeled the posterior precision and the posterior mean associated with cluster i at iteration n respectively.

2.4. Notes on the Perron-Frobenius Theorem

In this section we discuss results related to the Perron-Frobenius theorem. This theorem plays an important role in the theoretical considerations of this paper.

Theorem 1 (Perron-Frobenius theorem) *Consider a non-negative and irreducible real matrix* $\mathbf{A} : k \times k$*. There exists a* $\mathbf{v} > \mathbf{0}$ *such that* $\mathbf{A}\mathbf{v} = \rho(\mathbf{A})\mathbf{v}$*, where* $\rho(\mathbf{A})$ *denotes the spectral radius of* $\mathbf{A}$*.*

We use the notation $\mathbf{x} > \mathbf{y}$ to indicate that each component of $\mathbf{x}$ is greater than the corresponding component of $\mathbf{y}$. The following important lemma follows from the Collatz-Wielandt formula.

Lemma 1 *Let* $\rho(.)$ *be the spectral radius of a matrix. For any real matrix* $\mathbf{A} : k \times k$ *we have* $\rho(\mathbf{A}) \leq \rho(|\mathbf{A}|)$*.*

For the remainder of this paper we assume, without loss of generality, that GaBP-m is applied to an irreducible MG. If this assumption does not hold, then GaBP-m can be applied separately to irreducible subgraphs of the MG.

2.5. Preconditioned Walk-summability

The main theoretical result of this paper is the development of a sufficient condition for the convergence of GaBP-m. We label this condition preconditioned walk-summability, since it involves a certain preconditioning of the precision matrix based on how the clusters are chosen. We give the following definitions:

Definition 3 (Valid Preconditioner) *We call a matrix* $\mathbf{\Lambda} : k \times k$ *a valid preconditioner with respect to the clusters* $\mathcal{C}_i : i = 1, 2, \ldots, p$ *if* $\mathbf{\Lambda}_{ii}$ *is positive definite and* $\mathbf{\Lambda}_{ij} = \mathbf{0} : d_i \times d_j$*.*

Definition 4 (Preconditioned Walk-summability) *Consider a precision matrix* $\mathbf{S} : k \times k$ *and clusters* $\mathcal{C}_i : i = 1, 2, \ldots, p$*. The precision matrix* $\mathbf{S}$ *is preconditioned walk-summable if there exists a valid preconditioner* $\mathbf{\Lambda}$ *such that* $\mathbf{\Lambda}\mathbf{S}\mathbf{\Lambda}$ *is walk-summable.*

We note that preconditioned walk-summability is novel with respect to walk-summability and scaled diagonal dominance in the sense that the preconditioner does not need to be diagonal. In fact, the larger the chosen clusters, the wider the range of valid preconditioners. In the empirical section we give an explicit example where GaBP fails to converge on a non-(but preconditioned) walk-summable precision matrix.

2.6. Computation Trees

A computation tree is a representation of loopy belief propagation (LBP) in terms of inference on a tree-structured graph. Introduced by Weiss and Freeman (2001), it has been widely used in the study of the convergence behavior of LBP (Malioutov et al., 2006; Moallemi and Van Roy, 2009, 2010; Ruozzi and Tatikonda, 2013; Sui et al., 2015).

To generate a computation tree for GaBP-m, we start by clustering the k variables into p clusters. We then construct the higher-dimensional MG, which is used to construct the

topology of the computation tree exactly as we would have in the univariate case.

Our discussion of the process to generate a computation tree from an MG closely follows the discussion of Malioutov et al. (2006). We restrict our focus to computation trees for synchronous message-passing. Each message $m_{ij}^{(n-1)}(.)$ receives a computation tree $\mathcal{T}_{ij}^{(n)}$ (the superscript of the tree represents the depth of the tree rather than the iteration number). To construct $\mathcal{T}_{ij}^{(n)}$, we join all the trees $\mathcal{T}_{ki}^{(n-1)}$, for $k \in \mathcal{N}_i \setminus j$, at their common root node (this node refers to cluster i). We then add a new root node (with a reference to cluster j) and an edge to the previous root node. The computation tree for cluster i, $\mathcal{T}_i^{(n)}$, is constructed by joining all trees $\mathcal{T}_{ki}^{(n-1)}$, for $k \in \mathcal{N}_i$, at their common root node. The trees $\mathcal{T}_{ij}^{(1)}$ consist of a single node with reference to cluster j. An example of this process is given in Figure 2. Note that each node in the computation tree has a reference to a specific cluster.

The above process shows that a (synchronous) computation tree has a natural division into different layers. This allows a further conversion of the graph structure. All nodes in a given layer of the computation tree are collected into a single node. This leads to a graph in the form of a line topology (right-hand side of Figure 2). The line topology of $\mathcal{T}_i^{(n)}$ is labeled $\mathcal{L}_i^{(n)}$. This conversion will be used in the derivation of preconditioned walk-summability as a sufficient condition for convergence.

Each computation tree has an associated precision matrix and potential vector. The sparsity structure of the precision matrix follows the topology of the computation tree. A node in the computation tree with a reference to cluster i receives $\mathbf{S}_{ii}$ and $\boldsymbol{b}_i$ as a precision matrix and potential vector respectively. An edge between two nodes in the computation tree receives $\mathbf{S}_{ij}$ as its precision matrix, assuming the references of the nodes are i and j respectively.

The next section offers a proof that the posterior precision and posterior mean for node i of synchronous GaBP-m at iteration $n-1$ can be obtained by performing inference on the tree $\mathcal{T}_i^{(n)}$.

3. GaBP-m as Inference on a Computation Tree

In this section we discuss the interpretation of GaBP-m as inference on a computation tree and the effect of a certain change in the input variables on this algorithm. The main results of this section are:

1. A proof that the posterior mean and posterior precision of a cluster can be obtained by marginalizing a computation tree.
2. That there is a simple way of moving between the output of GaBP-m before and after a certain change in variables is introduced. This suggests that the performance of GaBP-m is robust towards this change in variables.

We focus, without loss of generality, on the computation tree associated with node 1. We denote the computation tree and line topology for node 1 by $\mathcal{T}_n$ and $\mathcal{L}_n$ respectively. Associate

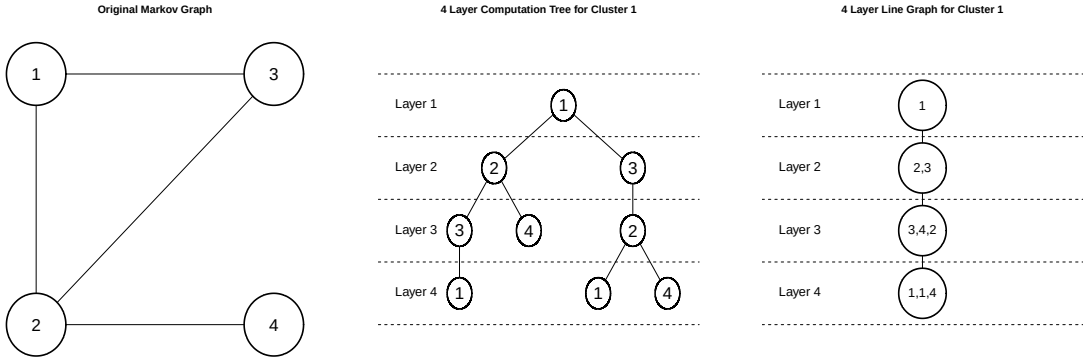

Figure 2: Different graph representations of an MG. The original MG is displayed to the left, the computation tree for cluster 1 is displayed in the middle, while the line graph representation of the same cluster is displayed to the right. Each node (in all three graphs) contains references to certain clusters.

with $\mathcal{T}_n$ the precision matrix $\mathbf{T}_n : m_n \times m_n$ and potential vector $\mathbf{t}_n : m_n \times 1$. Marginalizing $\mathcal{T}_n$ involves computing $\mathbf{T}_n^{-1}$ and $\mathbf{T}_n^{-1}\mathbf{t}_n$. We want to show that $\mathbf{P}_1^{(n-1)}$ and $\boldsymbol{\mu}_1^{(n-1)}$ can be obtained by extracting the diagonal block and subvector corresponding to cluster 1 of $\mathbf{T}_n^{-1}$ and $\mathbf{T}_n^{-1}\mathbf{t}_n$ respectively. We start by defining a few matrices for convenience. Refer to Appendix C for a running example of some of the considerations in this section.

3.1. Matrix Definitions

A movement along a computation tree is defined to be first by layer, starting at layer 1, and within each layer from left to right. Unless explicitly stated otherwise, we assume that the order of the nodes in the precision matrix and potential vector associated with $\mathcal{T}_n$ is according to the movement along $\mathcal{T}_n$.

Definition 5 (Row-extractor matrix of cluster t) *Define*

$$\mathbf{F}_t : d_t \times k = \begin{bmatrix} \mathbf{0} : d_t \times d_1 & \dots & \mathbf{0} : d_t \times d_{t-1} & \mathbf{I}_{d_t} & \mathbf{0} : d_t \times d_{t+1} & \dots & \mathbf{0} : d_t \times d_p \end{bmatrix}.$$

Recalling the ordering of our variables, we see that $\mathbf{F}_t\mathbf{S}$ extracts the rows of $\mathbf{S}$ corresponding to the variables in $\mathcal{C}_t$. Also note that:

$$\mathbf{S}_{jt}\mathbf{F}_t = \begin{bmatrix} \mathbf{0} : d_j \times d_1 & \dots & \mathbf{0} : d_j \times d_{t-1} & \mathbf{S}_{jt} & \mathbf{0} : d_j \times d_{t+1} & \dots & \mathbf{0} : d_j \times d_p \end{bmatrix}.$$

Definition 6 (Row-extractor matrix) *The row-extractor matrix $\mathbf{E}_n : m_n \times p$ is obtained by moving along the computation tree, noting the cluster reference of each node and stacking the row-extractor matrix of this cluster.*

Since $\mathbf{F}_t\boldsymbol{b} = \boldsymbol{b}_t$, we see that:

$$\mathbf{t}_n = \mathbf{E}_n\boldsymbol{b}.$$

Definition 7 (Node-extractor matrix) *Consider node j of the computation tree (read from left to right along the layers). Let $\mathbf{G}_n^{(j)}$ be such that $[\mathbf{G}_n^{(j)}]'\mathbf{T}_n$ extracts the rows of $\mathbf{T}_n$ corresponding to node j. Define the root extractor matrix as $\mathbf{G}_n = \mathbf{G}_n^{(1)}$.*

Definition 8 (Sparse replicate of cluster t) *A sparse replicate of cluster t is obtained by taking $\mathbf{F}_t\mathbf{S} = \begin{bmatrix}\mathbf{S}_{t1} & \dots & \mathbf{S}_{tt} & \dots & \mathbf{S}_{tp}\end{bmatrix}$ and setting $\mathbf{S}_{tj} \leftarrow \mathbf{0} : d_t \times d_j$ for all j in a non-empty subset of $\mathcal{N}_t$.*

Note that the matrix $\mathbf{T}_n\mathbf{E}_n$ has a row-block decomposition according to the computation tree. This means that the jth row block of $\mathbf{T}_n\mathbf{E}_n$ corresponds to the jth node of the computation tree. We now consider the following lemma.

Lemma 2 *We consider the following two scenarios for a node in the computation tree with reference to cluster t:*

1. *If this node is not in the final layer, then the corresponding row block of $\mathbf{T}_n\mathbf{E}_n$ is $\mathbf{F}_t\mathbf{S}$.*
2. *If this node is in the final layer, then the corresponding row block of $\mathbf{T}_n\mathbf{E}_n$ is a sparse replicate of cluster t.*

Proof Note that $\mathbf{T}_n\mathbf{E}_n$ has a row-block decomposition according to its computation tree, i.e. the jth row-block corresponds to the jth node in the computation tree. Consider node j of the computation tree. Note that we can extract the jth row block of $\mathbf{T}_n\mathbf{E}_n$ by computing $[\mathbf{G}_n^{(j)}]'\mathbf{T}_n\mathbf{E}_n$. We use the notation $I(j)$ to denote the cluster to which node j has a reference. Suppose that $I(j) = t$. Let O_j denote the neighborhood of node j in the computation tree, and let $J(O_j) = \{I(s) : s \in O_j\}$. We have that:

$$[\mathbf{G}_n^{(j)}]'\mathbf{T}_n\mathbf{E}_n = \mathbf{S}_{tt}\mathbf{F}_t + \sum_{k \in J(O_j)} \mathbf{S}_{tk}\mathbf{F}_k.$$

The proof follows directly from the fact that $J(O_j) = \mathcal{N}_t$, if j is not in the terminal layer, and $J(O_j) \subset \mathcal{N}_t$ otherwise. ■

A consequence of Lemma 2 is given in the following corollary.

Corollary 1 *We have:*

$$\mathbf{T}_n\mathbf{E}_n = \mathbf{E}_n\mathbf{S} + \mathbf{L}_n, \tag{9}$$

where the first m_{n-1} rows of $\mathbf{L}_n$ contain only zero entries.

We note that Lemma 2 and Corollary 1 do not only apply for computation trees of symmetric matrices, but also to computation trees of general non-symmetric matrices. We use this fact in some of the proofs contained in this paper.

The remainder of this section is dedicated to showing the validity of the following formulas:

$$\mathbf{P}_1^{(n-1)} = [\mathbf{G}_n' \mathbf{T}_n^{-1} \mathbf{G}_n]^{-1} \tag{10}$$

$$\boldsymbol{\mu}_1^{(n-1)} = \mathbf{G}_n' \mathbf{T}_n^{-1} \mathbf{E}_n \boldsymbol{b}. \tag{11}$$

We first discuss a tree-pruning procedure that is aimed at removing terminal nodes of the computation tree in such a way that inference at the root node remains unchanged. This is followed by a discussion of how the tree-pruning procedure can be used to prove formulas (10) and (11).

3.2. Tree-pruning Procedure

Consider a general tree $\mathcal{T}$ with precision matrix $\mathbf{T}$ and potential vector $\mathbf{t}$. Suppose that $r_\mu(\mathcal{T})$ and $r_P(\mathcal{T})$ give the marginal mean and marginal precision respectively of the root node of $\mathcal{T}$. We can obtain $r_P(\mathcal{T})$ by inverting the diagonal block of $\mathbf{T}^{-1}$ corresponding to the root node. In a similar way, $r_\mu(\mathcal{T})$ is obtained by extracting the first subvector (corresponding to the root node) of $\mathbf{T}^{-1}\mathbf{t}$.

The tree-pruning procedure iteratively prunes terminal nodes such that the marginal precision matrix and marginal mean vector of the root node of the remaining (trimmed) tree remain unchanged. Let $\tilde{\mathcal{T}}$ be the trimmed tree obtained after eliminating certain terminal nodes from $\mathcal{T}$, then we must have that $r_\mu(\mathcal{T}) = r_\mu(\tilde{\mathcal{T}})$ and $r_P(\mathcal{T}) = r_P(\tilde{\mathcal{T}})$.

At each step of the pruning process, we select one of the nodes in the penultimate layer (we call this node the bereaved parent), the objective being to prune its children in the terminal layer. This must be done in such a way that inference at the root node remains unchanged. In order to achieve this, we need to change certain elements of the precision and potential associated with the nodes in the trimmed tree from their corresponding values in the untrimmed tree. In the remainder of this section, we show that it is only necessary to change the precision matrix and potential vector associated with the bereaved parent. Moreover, these adjustments can be obtained by marginalizing the tree with the bereaved parent as the root node.

Let $\mathbf{T}$ and $\mathbf{t}$ be the precision matrix and potential vector respectively of $\mathcal{T}$. We can write, without loss of generality:

$$\mathbf{T} = \begin{bmatrix} \mathbf{T}_{11} & \mathbf{T}_{12} & \mathbf{0} \\ \mathbf{T}_{21} & \mathbf{T}_{22} & \mathbf{T}_{23} \\ \mathbf{0} & \mathbf{T}_{32} & \mathbf{T}_{33} \end{bmatrix}$$

$$\mathbf{t} = \begin{bmatrix} \mathbf{t}_1 \\ \mathbf{t}_2 \\ \mathbf{t}_3 \end{bmatrix}.$$

In general, we associate $\mathbf{t}_3$ with the nodes to be pruned, $\mathbf{t}_2$ with the bereaved parent and $\mathbf{t}_1$ with the remaining nodes. A similar interpretation holds for the precision matrix. Consider

$$\mathbf{T} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}$$
$$\mathbf{t} = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \end{bmatrix},$$

where $\mathbf{A}_{11} = \begin{bmatrix} \mathbf{T}_{11} & \mathbf{T}_{12} \\ \mathbf{T}_{21} & \mathbf{T}_{22} \end{bmatrix}$, $\mathbf{A}_{12} = \begin{bmatrix} \mathbf{0} \\ \mathbf{T}_{23} \end{bmatrix}$, $\mathbf{A}_{21} = \begin{bmatrix} \mathbf{0} & \mathbf{T}_{32} \end{bmatrix}$, $\mathbf{A}_{22} = \mathbf{T}_{33}$, $\mathbf{a}_1 = \begin{bmatrix} \mathbf{t}_1 \\ \mathbf{t}_2 \end{bmatrix}$ and $\mathbf{a}_2 = \mathbf{t}_3$. Block-wise matrix inversion yields

$$\mathbf{T}^{-1} = \begin{bmatrix} \mathbf{A}_{11.2}^{-1} & -\mathbf{A}_{11.2}^{-1}\mathbf{A}_{12}\mathbf{A}_{22}^{-1} \\ \dots & \dots \end{bmatrix},$$

where $\mathbf{A}_{11.2} = \mathbf{A}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{21}$ and the $\dots$ represents an irrelevant part of the matrix (nodes to be pruned). Let $\tilde{\mathcal{T}}$ denote the trimmed tree. In order to preserve root inference, the precision matrix and potential vector associated with $\tilde{\mathcal{T}}$ must be $\tilde{\mathbf{T}} = \mathbf{A}_{11.2}$ and $\tilde{\mathbf{t}} = \mathbf{a}_1 - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{a}_2$ respectively. It can be shown that:

$$\mathbf{A}_{11.2} = \begin{bmatrix} \mathbf{T}_{11} & \mathbf{T}_{12} \\ \mathbf{T}_{21} & \mathbf{T}_{22} - \mathbf{T}_{23}\mathbf{T}_{33}^{-1}\mathbf{T}_{32} \end{bmatrix} \tag{12}$$

$$\mathbf{a}_1 - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{a}_2 = \begin{bmatrix} \mathbf{t}_1 \\ \mathbf{t}_2 - \mathbf{T}_{23}\mathbf{T}_{33}^{-1}\mathbf{t}_3 \end{bmatrix}. \tag{13}$$

Equations (12) and (13) show that it is only necessary to adjust the precision and potential associated with the bereaved parent. Thereafter, we can prune the terminal nodes.

Let $\mathcal{M}$ be the tree containing the nodes to be pruned, with the bereaved parent as the root node. The precision and potential associated with $\mathcal{M}$ are $\mathbf{M} = \begin{bmatrix} \mathbf{T}_{22} & \mathbf{T}_{23} \\ \mathbf{T}_{32} & \mathbf{T}_{33} \end{bmatrix}$ and $\mathbf{m} = \begin{bmatrix} \mathbf{t}_2 \\ \mathbf{t}_3 \end{bmatrix}$ respectively. Setting $\mathbf{T}_{22.3} = \mathbf{T}_{22} - \mathbf{T}_{23}\mathbf{T}_{33}^{-1}\mathbf{T}_{32}$, we see, using block-wise matrix inversion, that:

$$r_P(\mathcal{M}) = \mathbf{T}_{22.3}$$
$$r_\mu(\mathcal{M}) = \mathbf{T}_{22.3}^{-1}[\mathbf{t}_2 - \mathbf{T}_{23}\mathbf{T}_{33}^{-1}\mathbf{t}_3].$$

In summary, to prune terminal nodes while preserving the marginal quantities of the root node, we need to do the following:

1. Compute $r_\mu(\mathcal{M})$ and $r_P(\mathcal{M})$.
2. Obtain the trimmed tree by eliminating the terminal nodes from the graph and only adjusting the potential and the precision of the bereaved parent to $r_P(\mathcal{M})r_\mu(\mathcal{M})$ and $r_P(\mathcal{M})$ respectively.

We now prove computation tree correctness by showing that the computations done by GaBP-m can be represented by the pruning procedure described in this section.

3.3. Computation Tree Correctness

By computation tree correctness we mean the validity of the formulas given in (10) and (11). In order to establish this, recall that $\mathbf{P}_{ij}^{(l)} = \mathbf{S}_{ii} + \sum_{t \in \mathcal{N}_i \setminus j} \mathbf{Q}_{ti}^{(l)}$ and define

$$\mathbf{z}_{ij}^{(l)} = \boldsymbol{b}_i + \sum_{t \in \mathcal{N}_i \setminus j} \mathbf{v}_{ti}^{(l)}$$
$$\boldsymbol{\mu}_{ij}^{(l)} = [\mathbf{P}_{ij}^{(l)}]^{-1} \mathbf{z}_{ij}^{(l)}.$$

Consider the following recursive expansions (recall Equations (5) to (7)):

$$\mathbf{P}_{ij}^{(l)} = \mathbf{S}_{ii} - \sum_{t \in \mathcal{N}_i \setminus j} \mathbf{S}_{it} [\mathbf{P}_{ti}^{(l-1)}]^{-1} \mathbf{S}_{it}$$
$$\mathbf{z}_{ij}^{(l)} = \boldsymbol{b}_i - \sum_{t \in \mathcal{N}_i \setminus j} \mathbf{S}_{it} [\mathbf{P}_{ti}^{(l-1)}]^{-1} \mathbf{z}_{ti}^{(l-1)}.$$

Suppose we have pruned $\mathcal{T}_n$ to have only l layers, where $3 \leq l \leq n$. Consider a bereaved parent, in layer $l-1$. We assume that the bereaved parent has a reference to cluster i while its parent (in layer $l-2$) has a reference to cluster j. Due to the construction of $\mathcal{T}_n$, we know that the bereaved parent will have mutually unconnected terminal nodes as children, each with a reference to a different cluster in $\mathcal{N}_i \setminus j$. Let $|\mathcal{N}_i \setminus j| = s_{ij}$ and $\mathcal{N}_i \setminus j = \{i_1, i_2, \ldots, i_{s_{ij}}\}$, and suppose that $\mathcal{M}$ is the tree with the selected bereaved parent as root, connected to its children in the terminal layer. We assume that the precision and potential associated with one of these terminal nodes, with a reference to cluster i_t, are $\mathbf{P}_{i_t i}^{(n-l)}$ and $\mathbf{z}_{i_t i}^{(n-l)}$ respectively. If $\mathbf{M}$ and $\mathbf{m}$ denote the precision and potential of $\mathcal{M}$ respectively, we have:

$$\mathbf{M} = \begin{bmatrix} \mathbf{S}_{ii} & \mathbf{S}_{ii_1} & \mathbf{S}_{ii_2} & \ldots & \mathbf{S}_{ii_{s_{ij}}} \\ \mathbf{S}_{i_1 i} & \mathbf{P}_{i_1 i}^{(n-l)} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{S}_{i_2 i} & \mathbf{0} & \mathbf{P}_{i_2 i}^{(n-l)} & \ldots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{S}_{i_{s_{ij}} i} & \mathbf{0} & \ldots & \mathbf{0} & \mathbf{P}_{i_{s_{ij}} i}^{(n-l)} \end{bmatrix} \tag{14}$$

$$\mathbf{m} = \begin{bmatrix} \boldsymbol{b}_i \\ \mathbf{z}_{i_1 i}^{(n-l)} \\ \mathbf{z}_{i_2 i}^{(n-l)} \\ \vdots \\ \mathbf{z}_{i_{s_{ij}} i}^{(n-l)} \end{bmatrix}. \tag{15}$$

Using block-wise matrix inversion, it is easy to see that:

$$r_P(\mathcal{M}) = \mathbf{S}_{ii} - \sum_{t=1}^{s_{ij}} \mathbf{S}_{ii_t} [\mathbf{P}_{i_t i}^{(n-l)}]^{-1} \mathbf{S}_{i_t i} = \mathbf{P}_{ij}^{(n-l+1)} \tag{16}$$

$$r_\mu(\mathcal{M}) = [\mathbf{P}_{ij}^{(n-l+1)}]^{-1} [\boldsymbol{b}_i - \sum_{t=1}^{s_{ij}} \mathbf{S}_{ii_t} [\mathbf{P}_{i_t i}^{(n-l)}]^{-1} \mathbf{z}_{i_t i}^{(n-l)}] = [\mathbf{P}_{ij}^{(n-l+1)}]^{-1} \mathbf{z}_{ij}^{(n-l+1)}. \tag{17}$$

To prune these terminal nodes, the tree-pruning procedure requires adjusting the precision and potential of the bereaved parent to $\mathbf{P}_{ij}^{(n-l+1)}$ and $\mathbf{z}_{ij}^{(n-l+1)}$ respectively. Once we have pruned all the terminal nodes, we see that we can apply a similar process to the new terminal layer, $l-1$.

Let us consider the case $l = n$, where a terminal node with reference to cluster i_1 and its parent with reference to cluster i are considered. At this stage, the precision and potential associated with this terminal node are $\mathbf{S}_{i_1 i_1}$ and $\boldsymbol{b}_{i_1}$ respectively. Since,

$$\begin{aligned}\mathbf{Q}_{i_1 i}^{(1)} &= -\mathbf{S}_{i_1 i}\mathbf{S}_{i_1 i_1}^{-1}\mathbf{S}_{i_1 i}\\ \mathbf{v}_{i_1 i}^{(1)} &= -\mathbf{S}_{i_1 i}\mathbf{S}_{i_1 i_1}^{-1}\boldsymbol{b}_{i_1},\end{aligned}$$

we see that $\mathbf{P}_{i_1 i}^{(0)} = \mathbf{S}_{i_1 i_1}$ and $\mathbf{z}_{i_1 i}^{(0)} = \boldsymbol{b}_{i_1}$. Hence, the type of pruning described in Equations (14) and (15) apply at all stages of pruning by induction.

Suppose we have completed the pruning such that only two layers remain. The trimmed tree will have a root node corresponding to cluster 1, and this node will be connected to terminal nodes, each of which has a reference to a different cluster in $\mathcal{N}_1$. Let $\mathcal{N}_1 = \{i_1, i_2, \ldots, i_{s_1}\}$. Since we have followed the pruning procedure, the precision matrix and potential vector associated with a terminal node, with reference to cluster i_t, are $\mathbf{z}_{i_t 1}^{(n-2)}$ and $\mathbf{P}_{i_t 1}^{(n-2)}$ respectively. The final step is to find the root marginal of the following precision matrix and potential vector:

$$\mathbf{M} = \begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{1i_1} & \mathbf{S}_{1i_2} & \ldots & \mathbf{S}_{1i_{n_1}} \\ \mathbf{S}_{i_1 1} & \mathbf{P}_{i_1 1}^{(n-2)} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{S}_{i_2 1} & \mathbf{0} & \mathbf{P}_{i_2 1}^{(n-2)} & \ldots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{S}_{i_{s_1} 1} & \mathbf{0} & \ldots & \mathbf{0} & \mathbf{P}_{i_{n_1} 1}^{(n-2)} \end{bmatrix}$$

$$\mathbf{m} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{z}_{i_1 1}^{(n-2)} \\ \mathbf{z}_{i_2 1}^{(n-2)} \\ \vdots \\ \mathbf{z}_{i_{s_1} 1}^{(n-2)} \end{bmatrix}.$$

One more application of block-wise matrix inversion reveals that:

$$r_P(\mathcal{T}_n) = \mathbf{S}_{11} - \sum_{t=1}^{s_1} \mathbf{S}_{1i_t}[\mathbf{P}_{i_t 1}^{(n-2)}]^{-1}\mathbf{S}_{i_t 1} = \mathbf{S}_{11} + \sum_{t=1}^{s_1} \mathbf{Q}_{i_t 1}^{(n-1)} = \mathbf{P}_1^{(n-1)} \tag{18}$$

$$\begin{aligned} r_\mu(\mathcal{T}_n) &= [\mathbf{P}_1^{(n-1)}]^{-1}[\boldsymbol{b}_1 - \sum_{t=1}^{s_1} \mathbf{S}_{1i_t}[\mathbf{P}_{i_t 1}^{(n-2)}]^{-1}\mathbf{z}_{i_t 1}^{(n-2)}] \\ &= [\mathbf{P}_1^{(n-1)}]^{-1}[\boldsymbol{b}_1 + \sum_{t=1}^{s_1} \mathbf{v}_{i_t 1}^{(n-1)}] = \boldsymbol{\mu}_1^{(n-1)}. \end{aligned} \tag{19}$$

Equations (18) and (19) validate the formulas given in (10) and (11). Note that similar formulas can be derived for all clusters i.

We note that GaBP-m applied to a tree-structured MG converges and yields the exact marginals at convergence. The key property is that the computation tree for a cluster in a tree-structured MG eventually replicates the tree itself. Since GaBP-m marginalizes the computation tree exactly, it will also marginalize the tree-structured MG exactly.

3.4. Recovery after Change in Variables

We discuss the relationship between GaBP-m applied to the precision matrix $\mathbf{S}$ and potential vector $\boldsymbol{b}$ before and after a certain change in variables is introduced. We show that the output of GaBP-m applied to the the original inputs can be recovered from the application of GaBP-m to the changed inputs through a simple transformation. This is a key relationship for proving that preconditioned walk-summability is a sufficient condition for convergence.

Consider a valid preconditioner $\mathbf{\Lambda}$ relative to the clusters $\mathcal{C}_i : i = 1, 2, \ldots, p$. We consider the relationship between GaBP-m applied to the inputs $\{\mathbf{S}, \boldsymbol{b}\}$ and $\{\tilde{\mathbf{S}}, \tilde{\boldsymbol{b}}\}$ where $\tilde{\mathbf{S}} = \mathbf{\Lambda}\mathbf{S}\mathbf{\Lambda}$ and $\tilde{\boldsymbol{b}} = \mathbf{\Lambda}\boldsymbol{b}$. Consider applying GaBP-m to the inputs $\{\tilde{\mathbf{S}}, \tilde{\boldsymbol{b}}\}$. For cluster i, we denote the posterior mean vector and posterior precision matrix (at iteration $n-1$) as $\tilde{\boldsymbol{\mu}}_i^{(n-1)}$ and $\tilde{\mathbf{P}}_i^{(n-1)}$ respectively. Consider the following lemma:

Lemma 3 *We have the following relationship for all clusters i:*

$$\begin{aligned} \boldsymbol{\mu}_i^{(n-1)} &= \mathbf{\Lambda}_{ii}\tilde{\boldsymbol{\mu}}_i^{(n-1)} \\ [\mathbf{P}_i^{(n-1)}]^{-1} &= \mathbf{\Lambda}_{ii}[\tilde{\mathbf{P}}_i^{(n-1)}]^{-1}\mathbf{\Lambda}_{ii}. \end{aligned}$$

Proof We consider, without loss of generality, the proof for cluster 1. Consider the block-diagonal matrix $\mathbf{B}_n : m_n \times m_n$ with a decomposition according to the computation tree. This matrix is constructed by moving along the computation tree, noting the reference of each node (say i) and adding a symmetric and positive definite matrix $\mathbf{\Lambda}_{ii}$ as diagonal blocks of $\mathbf{B}_n$ (all other entries are zero). It can be shown that $\tilde{\mathbf{T}}_n = \mathbf{B}_n\mathbf{T}_n\mathbf{B}_n$ is the precision matrix corresponding to the computation tree (for cluster 1) constructed from $\tilde{\mathbf{S}}$. The following holds:

$$\begin{aligned} \mathbf{P}_1^{(n-1)} &= [\mathbf{G}_n'\mathbf{T}_n^{-1}\mathbf{G}_n]^{-1} \\ &= [\mathbf{G}_n'\mathbf{B}_n\tilde{\mathbf{T}}_n^{-1}\mathbf{B}_n\mathbf{G}_n]^{-1}. \end{aligned} \tag{20}$$

Note that $\mathbf{G}_n'\mathbf{B}_n$ equals the first d_1 rows of $\mathbf{B}_n$; therefore, since $\mathbf{B}_n$ is block-diagonal, $\mathbf{G}_n'\mathbf{B}_n = \mathbf{\Lambda}_{11}\mathbf{G}_n'$. Equation (20) becomes

$$\mathbf{P}_1^{(n-1)} = \mathbf{\Lambda}_{11}^{-1}\tilde{\mathbf{P}}_1^{(n-1)}\mathbf{\Lambda}_{11}^{-1}, \tag{21}$$

where $\tilde{\mathbf{P}}_1^{(n-1)} = [\mathbf{G}_n' \tilde{\mathbf{T}}_n^{-1} \mathbf{G}_n]^{-1}$. Consider

$$\begin{aligned}\boldsymbol{\mu}_1^{(n-1)} &= \mathbf{G}_n' \mathbf{T}_n^{-1} \mathbf{E}_n \boldsymbol{b} \\ &= \mathbf{G}_n' \mathbf{B}_n \tilde{\mathbf{T}}_n^{-1} \mathbf{B}_n \mathbf{E}_n \boldsymbol{b} \\ &= \boldsymbol{\Lambda}_{11} \mathbf{G}_n' \tilde{\mathbf{T}}_n^{-1} \mathbf{B}_n \mathbf{E}_n \boldsymbol{b}.\end{aligned}$$

Now, since $\mathbf{B}_n$ is block-diagonal, we must have $\mathbf{B}_n \mathbf{E}_n = \mathbf{E}_n \boldsymbol{\Lambda}$, and therefore

$$\boldsymbol{\mu}_1^{(n-1)} = \boldsymbol{\Lambda}_{11} \mathbf{G}_n' \tilde{\mathbf{T}}_n^{-1} \mathbf{E}_n \boldsymbol{\Lambda} \boldsymbol{b} = \boldsymbol{\Lambda}_{11} \tilde{\boldsymbol{\mu}}_1^{(n-1)}, \tag{22}$$

where $\tilde{\boldsymbol{\mu}}_1^{(n-1)} = \mathbf{G}_n' \tilde{\mathbf{T}}_n^{-1} \mathbf{E}_n \boldsymbol{\Lambda} \boldsymbol{b}$. ■

Equations (21) and (22) show that there is an easy way of moving between the computations done by GaBP-m on $\{\mathbf{S}, \boldsymbol{b}\}$ and $\{\tilde{\mathbf{S}}, \tilde{\boldsymbol{b}}\}$ that does not depend on the iteration number. These considerations suggest that GaBP-m should be robust towards the change in variables as discussed above.

The way in which $\tilde{\mathbf{S}}$ and $\tilde{\boldsymbol{b}}$ are obtained from $\mathbf{S}$ and $\boldsymbol{b}$ is typically used to precondition inputs to iterative solvers of linear systems, such as the conjugate gradient (CG) method. In the case of the CG method, the difference in the convergence behavior of application to the different sets of inputs can be substantial, in the sense that the preconditioned variant converges much faster (Shewchuk, 1994). Equation (22) shows that the rate of convergence of $\boldsymbol{\mu}_1^{(n)}$ and $\tilde{\boldsymbol{\mu}}_1^{(n)}$ is identical, and that GaBP-m automatically benefits from the preconditioning without having to do this explicitly (convergence is guaranteed under preconditioned walk-summability). For other solvers of linear systems, we need to incorporate the preconditioning explicitly in order to obtain the benefit of a faster convergence rate. These considerations provide theoretical backing for some of the observations made in the empirical section.

4. Convergence of GaBP-m

In this section we prove that preconditioned walk-summability is a sufficient condition for the convergence of GaBP-m. We start by assuming that $\mathbf{S}$ is preconditioned walk-summable, and hence assume the existence of a valid preconditioner $\boldsymbol{\Lambda}$ such that $\tilde{\mathbf{S}} = \boldsymbol{\Lambda}\mathbf{S}\boldsymbol{\Lambda}$ is walk-summable. We then show that GaBP-m applied to $\tilde{\mathbf{S}}$ and $\tilde{\boldsymbol{b}} = \boldsymbol{\Lambda}\boldsymbol{b}$ is guaranteed to converge. The convergence of GaBP-m applied to the original inputs is proved using the results of Section 3.4.

4.1. Convergence of GaBP-m on Preconditioned Inputs

We can assume, without loss of generality, that the diagonal entries of $\tilde{\mathbf{S}}$ are all equal to one (if not, we can incorporate it in the preconditioning). Setting $\tilde{\mathbf{S}} = \mathbf{I}_k - \tilde{\mathbf{R}}$, we see that $\rho(|\tilde{\mathbf{R}}|) < 1$ by assumption. The precision matrix of the computation tree can be expressed as:

$$\tilde{\mathbf{T}}_n = \mathbf{I}_{m_n} - \tilde{\mathbf{R}}_n,$$

where $\tilde{\mathbf{R}}_n$ is a block tri-diagonal matrix corresponding to the line topology of the computation tree. Hence, we can express,

$$\tilde{\mathbf{R}}_n = \begin{bmatrix} \mathbf{C}_{11} & \mathbf{C}_{12} & & & & & & & & \\ \mathbf{C}_{21} & \mathbf{C}_{22} & \mathbf{C}_{23} & & & & & & & \\ & \mathbf{C}_{32} & \mathbf{C}_{33} & \mathbf{C}_{34} & & & & & & \\ & & & \cdots & \cdots & \cdots & & & & \\ & & & & \cdots & \cdots & \cdots & & & \\ & & & & & \cdots & \cdots & \cdots & & \\ & & & & & & & \mathbf{C}_{n-1;n-2} & \mathbf{C}_{n-1;n-1} & \mathbf{C}_{n-1;n} \\ & & & & & & & & \mathbf{C}_{n;n-1} & \mathbf{C}_{nn} \end{bmatrix},$$

with the understanding that the blank spaces are all zero. The $\mathbf{C}_{ii}$ blocks are zero-diagonal square matrices with numbers of rows equal to the sums of the sizes of all nodes in the ith layer of the computation tree. We now give the following lemma (the proof is given in Appendix A.1).

Lemma 4 *Let $\tilde{\mathbf{S}} = \mathbf{I}_k - \tilde{\mathbf{R}}$ with $\rho(|\tilde{\mathbf{R}}|) < 1$, then:*

1. *$\rho(|\tilde{\mathbf{R}}_n|) \leq \rho(|\tilde{\mathbf{R}}|)$ for all n.*
2. *The quantity $||\tilde{\mathbf{R}}_n^k||_\infty \leq \kappa_1[\rho(|\tilde{\mathbf{R}}|)]^k$ for a constant κ_1 that does not depend on n.*
3. *$||\tilde{\mathbf{T}}_n^{-1}||_\infty \leq \kappa_2$ for a constant κ_2 that does not depend on n.*

The constants referenced are:

$$\kappa_1 = \frac{max_i\{v_i\}}{min_j\{v_j\}} \tag{23}$$

$$\kappa_2 = \frac{\kappa_1}{1 - \rho(|\tilde{\mathbf{R}}|)}, \tag{24}$$

where $\mathbf{v} = (v_1, v_2, \ldots, v_k)' > \mathbf{0}$ such that $|\tilde{\mathbf{R}}|\mathbf{v} = \rho(|\tilde{\mathbf{R}}|)\mathbf{v}$.

Although Lemma 4 uses the computation tree for cluster 1 as an example, we see that this cluster plays no role in the computation of κ_1 and κ_2. Lemma 4 also applies to computation trees constructed for other clusters, and the computation of κ_1 and κ_2 remains as in (23) and (24) respectively (they do not depend on the cluster chosen to represent the root node).

We restrict our focus to proving the convergence of the posterior mean and posterior variances of GaBP-m applied to the preconditioned inputs. Let $\tilde{\boldsymbol{\mu}}$ be such that $\tilde{\mathbf{S}}\tilde{\boldsymbol{\mu}} = \boldsymbol{b}$ and suppose $\lfloor\tilde{\mathbf{S}}^{-1}\rfloor_{ii}$ represents the submatrix of $\tilde{\mathbf{S}}^{-1}$ corresponding to the variables in cluster i. The following theorem summarizes our convergence results (for a proof see Appendix B).

Theorem 2 *Consider GaBP-m applied to the walk-summable preconditioned inputs* $\tilde{\mathbf{S}}$ *and* $\tilde{\boldsymbol{b}}$ *with clusters* $\mathcal{C}_i : i = 1, 2, \ldots p$. *The following results apply for all clusters* i*:*

1. $||\tilde{\boldsymbol{\mu}}_i^{(n-1)} - \tilde{\boldsymbol{\mu}}_i||_\infty \leq \kappa_4[\rho(|\tilde{\mathbf{R}}|)]^{n-1}$.
2. $[\tilde{\mathbf{P}}_i^{(n-1)}]^{-1} \sim \lfloor \tilde{\mathbf{S}}^{-1} \rfloor_{ii} - \tilde{\mathbf{Z}}_n^{(i)}$.
3. $||\tilde{\mathbf{Z}}_n^{(i)}||_\infty \leq \kappa_1\kappa_2[\rho(|\tilde{\mathbf{R}}|)]^{t-1}$ *for* $n > t - 1$.

Here, $\kappa_4 = \kappa_1\kappa_2\kappa_3$, *with* $\kappa_3 = ||\tilde{\mathbf{S}}||_\infty||\tilde{\boldsymbol{\mu}}||_\infty$, $\tilde{\mathbf{Z}}_n^{(i)}$ *forms a Cauchy sequence and* t *is the first layer after the root node where cluster* i *is referenced in its computation tree.*

Since $\rho(|\tilde{\mathbf{R}}|) < 1$, Theorem 2 clearly implies convergence of the posterior means and covariance matrices for all clusters. Moreover, the converged posterior means represent the exact marginal means, while the converged posterior covariance matrices can be regarded as approximations to the true marginal covariance matrices. The approximation error of the converged posterior covariance matrices depends on $\lim_{n\to\infty} \tilde{\mathbf{Z}}_n^{(i)} = \tilde{\mathbf{Z}}_\infty^{(i)}$. In Theorem 2, we always have that $t \geq 3$, but t may be larger depending on the topology of the MG. Note that, if the computation tree never references cluster i again (as in the case of a tree-structured $\mathbf{S}$), then the converged posterior covariance matrix will be equal to the exact marginal covariance matrix. We see that Theorem 2 not only proves convergence, but also gives the rate of convergence of the posterior means and a framework for analyzing the accuracy of the approximate marginal covariance matrices.

4.2. Convergence of GaBP-m on Original Inputs

Define $\boldsymbol{\mu}$, $\boldsymbol{\mu}_i$ and $\lfloor \mathbf{S}^{-1} \rfloor_{ii}$ analogous to $\tilde{\boldsymbol{\mu}}$, $\tilde{\boldsymbol{\mu}}_i$ and $\lfloor \tilde{\mathbf{S}}^{-1} \rfloor_{ii}$ respectively. We give the following corollary to Theorem 2.

Corollary 2 *Consider GaBP-m applied to the original (preconditioned walk-summable) inputs* $\mathbf{S}$ *and* $\boldsymbol{b}$ *with clusters* $\mathcal{C}_i : i = 1, 2, \ldots p$. *The following results apply for all clusters* i*:*

1. $||\boldsymbol{\mu}_i^{(n-1)} - \boldsymbol{\mu}_i||_\infty \leq \kappa_4||\boldsymbol{\Lambda}_{ii}||_\infty[\rho(|\tilde{\mathbf{R}}|)]^{n-1}$.
2. $[\mathbf{P}_i^{(n-1)}]^{-1} \sim \lfloor \mathbf{S}^{-1} \rfloor_{ii} - \mathbf{Z}_n^{(i)}$.
3. *For* $n > t - 1$ *we have* $||[\mathbf{P}_i^{(n)}]^{-1} - \lfloor \mathbf{S}^{-1} \rfloor_{ii}||_\infty \leq \kappa_1\kappa_2||\boldsymbol{\Lambda}_{ii}||_\infty^2[\rho(|\tilde{\mathbf{R}}|)]^{t-1}$.

Here, $\mathbf{Z}_n^{(i)} = \boldsymbol{\Lambda}_{ii}\tilde{\mathbf{Z}}_n^{(i)}\boldsymbol{\Lambda}_{ii}$ *forms a Cauchy sequence,* $\kappa_i : i = 1, 2, 3, 4$ *are defined as in Theorem 2, and* t *is the first layer after the root node where cluster* i *is referenced in its computation tree.*

Proof The proof follows directly from Theorem 2, Lemma 3 and:

$$\boldsymbol{\mu}_i = \boldsymbol{\Lambda}_{ii}\tilde{\boldsymbol{\mu}}_i$$
$$\lfloor \mathbf{S}^{-1} \rfloor_{ii} = \boldsymbol{\Lambda}_{ii}\lfloor \tilde{\mathbf{S}}^{-1} \rfloor_{ii}\boldsymbol{\Lambda}_{ii},$$

which is true for all clusters i. ■

Clearly, Corollary 2 implies that preconditioned walk-summability is a sufficient condition for convergence of GaBP-m applied to the original inputs.

5. Empirical Results

In this next section we provide three empirical studies of the GaBP-m algorithm. The first study shows the novelty of preconditioned walk-summability with respect to walk-summability and scaled diagonal dominance using a specific set of inputs. The second study illustrates the advantages of using GaBP-m over the CG method within the context of preconditioning. We conclude by promoting GaBP-m over GaBP in an inference context.

5.1. Preconditioned Walk-summability

Consider applying GaBP-m to a precision matrix $\mathbf{S} : k \times k$ and potential vector $\boldsymbol{b} : k \times 1$, where the variables are assigned to nodes based on the clusters $\mathcal{C}_i : i = 1, 2, \ldots, p$. We have not discussed the selection of $\mathbf{\Lambda}$ in practice. One type of preconditioner we found to be effective in the prediction of the convergence of GaBP is the selection:

$$\mathbf{\Lambda}_{ii} = \mathbf{S}_{ii}^{-0.5} \text{ for } i = 1, 2, \ldots, p. \tag{25}$$

We now provide a specific example illustrating the novelty of preconditioned walk-summability. Consider the following precision matrix and potential vector: [1]

$$\mathbf{S} = \begin{bmatrix} 1.0000000 & 0.17373710 & 0.1850847 & 0.3354267 & 0.26006082 & 0.2192431 \\ 0.1737371 & 1.00000000 & 0.0881614 & 0.2410132 & 0.03527682 & 0.1426373 \\ 0.1850847 & 0.08816140 & 1.0000000 & 0.3954153 & 0.24977742 & 0.2611699 \\ 0.3354267 & 0.24101317 & 0.3954153 & 1.0000000 & 0.24246971 & 0.1855578 \\ 0.2600608 & 0.03527682 & 0.2497774 & 0.2424697 & 1.00000000 & 0.8966630 \\ 0.2192431 & 0.14263726 & 0.2611699 & 0.1855578 & 0.89666296 & 1.0000000 \end{bmatrix}$$

$$\boldsymbol{b} = \begin{bmatrix} 0.1878888 \\ 0.0430620 \\ 0.5864501 \\ 0.4414838 \\ 0.2120225 \\ 0.1740536 \end{bmatrix}.$$

The spectral radius of $|\mathbf{I}_k - \mathbf{S}|$ is $1.4069 > 1$, which does not give a decisive answer on whether GaBP will converge; in fact, we see that the application of GaBP to these inputs diverges. We now consider GaBP-m applied to these inputs, with the clusters $\mathcal{C}_1 = \{1, 2\}$, $\mathcal{C}_2 = \{3, 4\}$ and $\mathcal{C}_3 = \{5, 6\}$. Using the preconditioning defined in (25), we see that the spectral radius of $|\mathbf{I}_k - \tilde{\mathbf{S}}|$ is $0.6689 < 1$, and hence $\mathbf{S}$ is preconditioned walk-summable. The application of GaBP-m yields convergence after 19 iterations (using $\epsilon = 10^{-10}$). This convergence is explained by preconditioned walk-summability. We also see that GaBP-m can converge in cases where GaBP does not.

We conclude this example by investigating the bounds given in Corollary 2. Suppose we

1. These inputs represent an extract from the correlation matrix and correlation vector of the diabetes data used by Efron et al. (2004) to illustrate the least angle regression algorithm.

want n to be sufficiently large such that $\max_i\{||\boldsymbol{\mu}_i^{(n)} - \boldsymbol{\mu}_i||_\infty\} \leq \epsilon$, for some specified ϵ. Under the assumption of preconditioned walk-summability, we have:

$$n \geq \frac{\log(\frac{\epsilon}{\kappa_4 \max_i\{||\boldsymbol{\Lambda}_{ii}||_\infty\}})}{\log(\rho(|\tilde{\mathbf{R}}|))}. \tag{26}$$

Evaluating (26), we see that convergence will occur after at most 66 iterations (almost 3.5 times the actual number). The bound for the dissimilarity between the posterior and marginal variances in Corollary 2, where $t = 3$, can be evaluated as 5.191402 for cluster 1, which is poor compared to the actual value of 0.0929461. This bound will likely be better for sparse graphs (our example represents a dense graph).

5.2. Robustness Towards Change in Variables

In this empirical study, we compare GaBP and GaBP-m to the CG and PCG algorithms. A description of the CG algorithm can be found in Shewchuk (1994). Our goal with this empirical comparison is not to promote GaBP and GaBP-m as solvers of linear systems, but rather to illustrate the stability of these algorithms with regard to the change in variables discussed in Section 3.4. The idea is to compare the performance of these algorithms by considering the application of the basic methods to an ill-conditioned $\mathbf{S}$. The performances on a well-conditioned matrix, $\tilde{\mathbf{S}} = \boldsymbol{\Lambda}\mathbf{S}\boldsymbol{\Lambda}$, are also compared.

In Section 3.4 we found that the rate of convergence of the means in GaBP-m applied to $\mathbf{S}$ will be identical to that of GaBP-m applied to $\tilde{\mathbf{S}}$, assuming that the clusters are chosen appropriately. In this empirical study, we will argue that the CG algorithm cannot benefit from the improved conditioning of $\tilde{\mathbf{S}}$ without explicitly incorporating the preconditioning. The main advantage of GaBP-m over the CG and PCG methods is that we do not need to know the preconditioner in advance to obtain the improved convergence rate. In this sense, GaBP-m automatically benefits from the convergence rate of the best valid preconditioned inputs.

Kamper et al. (2018) specify a method for generating precision matrices with arbitrary zero-diagonal spectral radius. The zero-diagonal spectral radius of $\mathbf{S}$ is obtained as the spectral radius of $\mathbf{R}$ in $\mathbf{S} = \mathbf{I}_k - \mathbf{R}$, assuming that $\mathbf{S}$ has been scaled to have only ones along its diagonal. We use this method to generate our inputs to the different algorithms.

This empirical study involves two illustrations. In the first illustration we compare GaBP to the CG and PCG algorithms. We note that the considerations of Section 3.4 also apply to GaBP in the sense that we can choose k clusters each of size 1. This constrains $\boldsymbol{\Lambda}$ to be a diagonal matrix, and we will show that GaBP is robust towards changes in variables induced by a diagonal preconditioner. To generate a set of inputs, we do the following:

1. Select a zero-diagonal spectral radius ($\tilde{\rho}$) uniformly from the interval $[0.5; 0.9]$.
2. Use the method of Kamper et al. (2018) to generate a $1\,000 \times 1\,000$ precision matrix $\tilde{\mathbf{S}}$ (with zero-diagonal spectral radius equal to $\tilde{\rho}$) and potential vector $\tilde{\boldsymbol{b}} : 1\,000 \times 1$. These will act as our preconditioned inputs.

3. Generate $\mathbf{u} : 1\ 000 \times 1$, where each element is selected independently from a uniform random distribution over the interval $[0.01; 1]$. Set $\mathbf{D_u} = \text{diag}(\mathbf{u})$.

4. Our original (ill-conditioned) inputs are $\mathbf{S} = \mathbf{D_u}\tilde{\mathbf{S}}\mathbf{D_u}$ and $\boldsymbol{b} = \mathbf{D_u}\tilde{\boldsymbol{b}}$.

We generate 1 000 of these inputs randomly and compare four methods. The methods are GaBP on $\{\mathbf{S}, \boldsymbol{b}\}$, GaBP on $\{\tilde{\mathbf{S}}, \tilde{\boldsymbol{b}}\}$, CG on $\{\mathbf{S}, \boldsymbol{b}\}$ and CG on $\{\tilde{\mathbf{S}}, \tilde{\boldsymbol{b}}\}$. The labels for these methods are gabp.org, gabp.pre, cg.org and cg.pre respectively. The "org" denotes application to the original ill-conditioned inputs, while the "pre" denotes the preconditioned inputs. We compare these methods by recording the number of iterations required by each method for convergence. Convergence is defined to occur when $||\mathbf{S}\boldsymbol{\mu}^{(n)} - \boldsymbol{b}||_\infty \leq 10^{-6}$. The results of our simulations are given in the top graph of Figure 3. We see that the performance of GaBP on the different sets of inputs is nearly identical. This is in contrast to the CG method, where the difference in performance is substantial and the robustness of GaBP to the preconditioning is well illustrated. The CG method on the original inputs performs the worst by far out of all the methods; however, the preconditioned version outperforms the GaBP variants. It is important to note that the diagonal PCG method requires the specification of a preconditioner and does not keep the computations on the scale of the original inputs.

In our second illustration, we consider the automatic preconditioning done by GaBP-m, with the understanding that the nodes selected are higher-dimensional. This allows our preconditioner to be block-diagonal (in contrast to diagonal as for GaBP). We consider $1\ 000 \times 1\ 000$ precision matrices where we have 31 clusters each of size 31, and one additional cluster of size 39. Consider the following simulation procedure:

1. Select a zero-diagonal spectral radius ($\tilde{\rho}$) uniformly from the interval $[0.5; 0.9]$.

2. Generate a precision matrix $\tilde{\mathbf{S}} : 1\ 000 \times 1\ 000$ (with zero-diagonal spectral radius equal to $\tilde{\rho}$) and potential vector $\tilde{\boldsymbol{b}} : 1\ 000 \times 1$. These will act as our preconditioned inputs.

3. We define a $1\ 000 \times 1\ 000$ matrix $\tilde{\boldsymbol{\Lambda}}$. For each cluster i of size d_i, we generate a $d_i \times d_i$ precision matrix with zero-diagonal spectral radius equal to 1.5, and set $\tilde{\boldsymbol{\Lambda}}_{ii}$ equal to this matrix. When $i \neq j$, $\tilde{\boldsymbol{\Lambda}}_{ij}$ is a matrix of zeros.

4. Our original (ill-conditioned) inputs are $\mathbf{S} = \tilde{\boldsymbol{\Lambda}}\tilde{\mathbf{S}}\tilde{\boldsymbol{\Lambda}}$ and $\boldsymbol{b} = \tilde{\boldsymbol{\Lambda}}\tilde{\boldsymbol{b}}$.

Again, we generate 1 000 of these structures and compare four different methods. We apply GaBP-m to $\{\mathbf{S}, \boldsymbol{b}\}$ with clusters as used in the simulation, and GaBP-m to $\{\tilde{\mathbf{S}}, \tilde{\boldsymbol{b}}\}$ with clusters as used in the simulation. These methods are labeled gabp.org and gabp.pre respectively. Let $\mathbf{D}_{\tilde{\boldsymbol{\Lambda}}}$ be the diagonal matrix, with the diagonal obtained from the diagonal of $\tilde{\boldsymbol{\Lambda}}$. We define a diagonal preconditioner $\mathbf{D}_{\tilde{\boldsymbol{\Lambda}}}^{-1}$ (this was used in the previous simulations) and apply CG to $\{\mathbf{D}_{\tilde{\boldsymbol{\Lambda}}}^{-1}\mathbf{S}\mathbf{D}_{\tilde{\boldsymbol{\Lambda}}}^{-1}, \mathbf{D}_{\tilde{\boldsymbol{\Lambda}}}^{-1}\boldsymbol{b}\}$. This method is labeled cg.diag. The last method is CG applied to $\{\tilde{\mathbf{S}}, \tilde{\boldsymbol{b}}\}$, which we label cg.pre. The results are illustrated in the bottom graph in Figure 3. We see a similar pattern as in the diagonal preconditioning case. The CG-based variants show the most volatility towards the preconditioning. In contrast to

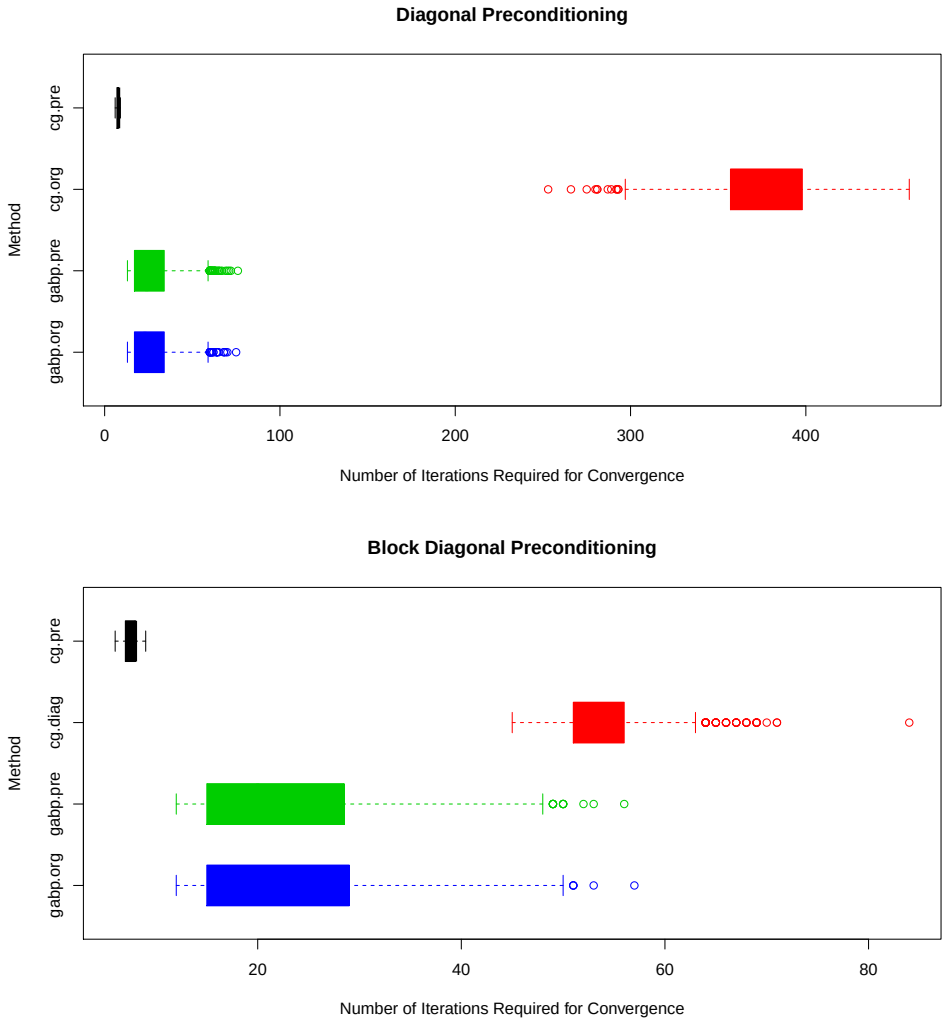

Figure 3: Visualization of the results of our simulations regarding preconditioning. The top and bottom graphs represent diagonal and block-diagonal preconditioning respectively. In both cases, we see that the performance of GaBP(-m) on the preconditioned and original inputs is nearly identical. This is in contrast to the performance of the CG method on both sets of inputs, where the difference is substantial.

the diagonal case, we see that the GaBP-m variants now tend to converge with a smaller number of iterations when compared to the diagonally preconditioned CG variant. The full preconditioned CG variant performs the best of all the methods.

5.3. GaBP-m vs GaBP

In this section, we discuss the utility of GaBP-m compared to GaBP. To facilitate this comparison, we used the following simulation procedure:

1. Select a zero-diagonal spectral radius uniformly from $[0.8; 1]$.
2. Generate a 100×100 precision matrix and a corresponding 100×1 potential vector using the simulation scheme from Kamper et al. (2018). The precision matrix is scaled to have a zero-diagonal spectral radius, as determined in (1).
3. Randomly generate 10 clusters each of size 10.

The above procedure was repeated 1 000 times to generate 1 000 different inputs. GaBP and GaBP-m (using the randomly selected clusters) were then applied to each of these data structures. The goal is to compare GaBP with GaBP-m based on the following considerations:

1. The number of iterations required for convergence.
2. The univariate inference quality of each method. We note that both GaBP and GaBP-m supply the exact marginal means at convergence. GaBP-m can be used for univariate marginal approximation by first approximating the higher-dimensional marginal, and then computing the univariate marginals from this approximation using a direct method. For instance, using GaBP-m, we can approximate the marginal precision associated with a specific cluster and then apply direct matrix inversion to approximate the univariate marginal precisions.

For the univariate inference quality we consider the KL divergence of the exact marginal distribution to the approximate marginal distribution. If f_i and $\hat{f}_i$ are the exact and approximate marginal associated with variable i, we are computing:

$$D_{KL}(f_i||\hat{f}_i) = \int_{-\infty}^{\infty} f_i(y) \log\left(\frac{f_i(y)}{\hat{f}_i(y)}\right) dy.$$

Note that, for a specific method, we get a KL divergence for each variable. The inference quality of a method for a specific input is measured by the mean of these divergences.

The results from our simulation study are summarized in Figure 4. Boxplots are drawn for the number of iterations required for convergence by GaBP and GaBP-m for the different simulated inputs. To illustrate inference quality, we draw boxplots of the mean KL divergences arising from the different inputs. The number of iterations required for convergence by GaBP-m tends to be less than that required by GaBP. We see that the univariate inference quality of GaBP-m tends to be better than that supplied by GaBP. GaBP-m also

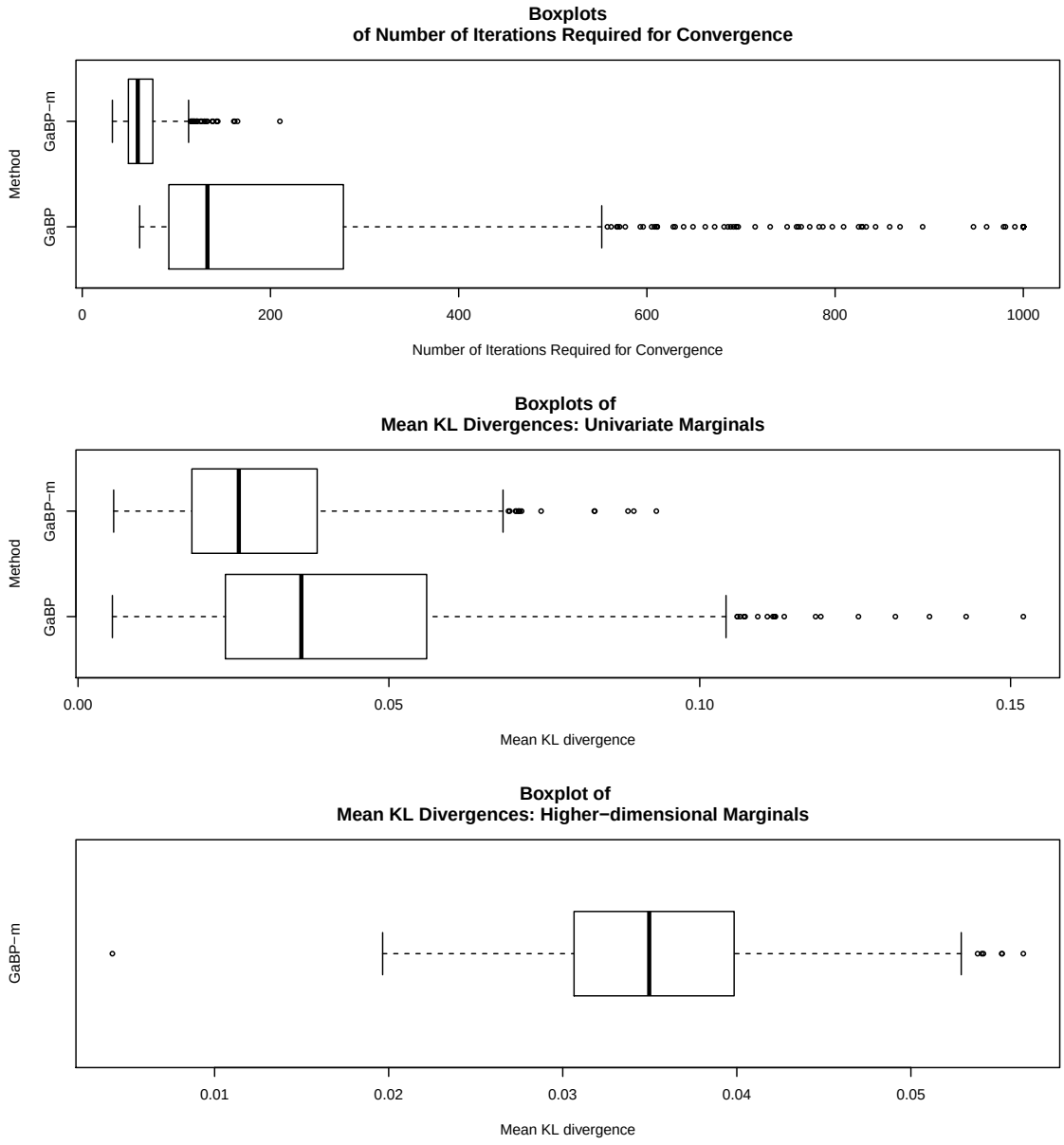

Figure 4: Visualization of the results of our simulations regarding the utility of GaBP-m. The top graph represents the number of iterations required for convergence by GaBP-m and GaBP. The middle graph illustrates the univariate inference quality of both methods, while the bottom graph analyses the higher-dimensional inference quality of GaBP-m. We see that GaBP-m tends to require a smaller number of iterations to converge and tends to provide more accurate univariate marginals. GaBP-m also provides useful approximate higher-dimensional marginals, something that GaBP cannot do.

has the advantage of offering approximations of the marginal precisions of the selected clusters. The inference quality of GaBP-m, with respect to the clusters, is also given in Figure 4. Inference quality was defined analogous to the univariate case. We see that GaBP-m can provide useful approximations for the higher-dimensional marginal distributions.

6. Conclusion

In this paper, we have considered a multivariate extension of belief propagation applied to pairwise Gaussian MGs (we labeled this GaBP-m). The main advantage of GaBP-m over GaBP is that it can be used to approximate the precision matrix of higher-dimensional marginals. In addition, GaBP-m may require fewer iterations than GaBP to converge and can be used to find better approximations for the univariate marginals. A multivariate version of the computation tree analysis, used to analyze univariate GaBP, was given. These computation trees were used to derive a relationship between GaBP-m applied to a set of inputs, before and after a certain change in variables was introduced. It was argued that GaBP-m is robust towards this change in variables, a property not shared by other solvers of linear systems such as the conjugate gradient solver. This was also illustrated empirically. The concept of preconditioned walk-summability was defined and shown to be a sufficient condition for convergence of the GaBP-m algorithm. Bounds were provided for the convergence speed and inference quality of GaBP-m under preconditioned walk-summability. The novelty of preconditioned walk-summability with respect to walk-summability and scaled diagonal dominance was discussed and illustrated empirically.

Unfortunately, GaBP-m is not guaranteed to converge for arbitrary precision matrices. Methods of augmenting this algorithm to converge, without compromising inference quality, should be considered in further research. A drawback of GaBP-m over GaBP is that the computational load per iteration scheme is higher. Methods to reduce this computational load, such as rank-1 updates, should be considered. Asynchronous GaBP-m and the effect of different initializations on the performance of the algorithm also require attention.

Acknowledgments

The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the authors and are not necessarily to be attributed to the NRF.

Appendix A. Proofs Leading to Lemma 4

We consider two additional lemmas before proving Lemma 4. Note that we focus, without loss of generality, on cluster 1. Consider clustering the real matrix $\mathbf{A} : k \times k$ according to $\mathcal{C}_i : i = 1, 2, \ldots, p$. We can construct a computation tree for $\mathbf{A}$ regardless of whether or not it is symmetrical. We denote the precision matrix of the computation tree of $\mathbf{A}$, for cluster 1, by $\mathbf{T}_n(\mathbf{A})$. In the appendices we still adopt the convention that $\mathbf{T}_n = \mathbf{T}_n(\mathbf{S})$ and $\tilde{\mathbf{T}}_n = \mathbf{T}_n(\tilde{\mathbf{S}})$. We now present and prove the two additional lemmas.

Lemma 5 *For all n, we have that*

$$||\mathbf{T}_n(\mathbf{A})||_\infty \leq ||\mathbf{A}||_\infty$$

with equality if the computation tree is of sufficient depth, such that all clusters are referenced before the terminal layer.

Proof Consider a node of the computation tree and suppose that this node has a reference to cluster s:

1. If this node is before the terminal layer, then its neighborhood in the computation tree will reference each cluster in $\mathcal{N}_s$ exactly once. Hence, the infinity norm of the row block of $\mathbf{T}_n(\mathbf{A})$ corresponding to this node will be equal to the infinity norm of the row block of $\mathbf{A}$ corresponding to the clusters in $\mathcal{N}_s$.
2. If this node is in the terminal layer, then its neighborhood in the computation tree will reference each cluster in $\mathcal{N}_s$ at most once (certain clusters will not be referenced). Hence, the infinity norm of the row block of $\mathbf{T}_n(\mathbf{A})$ corresponding to this node will be at most equal to the infinity norm of the row block of $\mathbf{A}$ corresponding to the clusters in $\mathcal{N}_s$.

The result follows directly from these points. ■

Lemma 6 *Consider an irreducible matrix $\mathbf{A} : t \times t$ consisting of non-negative elements. There exists an eigenvector $\mathbf{v}$, containing only positive elements, such that $\mathbf{Av} = \rho(\mathbf{A})\mathbf{v}$ (the spectral radius of $\mathbf{A}$ is also an eigenvalue of $\mathbf{A}$). Furthermore, if $\mathbf{D} = diag(\mathbf{v})$, then:*

$$\rho(\mathbf{A}) = ||\mathbf{D}^{-1}\mathbf{AD}||_\infty.$$

Proof First we note that the Perron-Frobenius theorem guarantees the existence of a vector $\mathbf{v}$, consisting only of positive elements, such that $\mathbf{Av} = \rho(\mathbf{A})\mathbf{v}$. Let $\mathbf{A} = [a_{st}]$ and $\mathbf{v} = (v_1, v_2, \ldots, v_t)'$. Note that

$$\mathbf{D}^{-1}\mathbf{AD} = \left[\frac{v_s a_{ts}}{v_t}\right],$$

and all elements of $\mathbf{D}^{-1}\mathbf{AD}$ are non-negative. The consequence is that the sum of the absolute elements of row t of $\mathbf{D}^{-1}\mathbf{AD}$ is

$$\sum_{s=1}^{t} \frac{v_s a_{ts}}{v_t} = \frac{1}{v_t}\sum_{s=1}^{t} v_s a_{ts} = \frac{v_t \rho(\mathbf{A})}{v_t} = \rho(\mathbf{A}),$$

since $\mathbf{v}$ is the eigenvector of $\mathbf{A}$ corresponding to the eigenvalue $\rho(\mathbf{A})$. The row sums of the absolute elements of $\mathbf{D}^{-1}\mathbf{A}\mathbf{D}$ are all equal to $\rho(\mathbf{A})$, and hence $||\mathbf{D}^{-1}\mathbf{A}\mathbf{D}||_\infty = \rho(\mathbf{A})$. ■

A.1. Proof of Lemma 4

Proof The proofs for the components of Lemma 4 are:

1. We assume that the matrix $\tilde{\mathbf{S}}$ is irreducible; if it is not, then GaBP-m can be applied to separate irreducible matrices. Note that the diagonal entries of $\tilde{\mathbf{S}}$ are all equal to one by assumption. Set $\tilde{\mathbf{S}} = \mathbf{I}_k - \tilde{\mathbf{R}}$, and therefore $|\tilde{\mathbf{R}}|$ is irreducible. Let $\mathbf{v}$ be the eigenvector of $|\tilde{\mathbf{R}}|$ corresponding to its spectral radius. Since $|\tilde{\mathbf{R}}|$ is non-negative and irreducible, the Perron-Frobenius theorem guarantees that the elements of $\mathbf{v}$ will be positive. Define $\bar{\mathbf{v}}_n = \mathbf{E}_n\mathbf{v}$, $\mathbf{D}_\mathbf{v} = \text{diag}(\mathbf{v})$ and $\mathbf{D}_{\bar{\mathbf{v}}_n} = \text{diag}(\bar{\mathbf{v}}_n)$. By Lemmas 5 and 6, we see that:

$$\begin{aligned}
\rho(|\tilde{\mathbf{R}}_n|) &= \rho(\mathbf{D}_{\bar{\mathbf{v}}_n}^{-1}|\tilde{\mathbf{R}}_n|\mathbf{D}_{\bar{\mathbf{v}}_n}) \\
&\leq ||\mathbf{D}_{\bar{\mathbf{v}}_n}^{-1}|\tilde{\mathbf{R}}_n|\mathbf{D}_{\bar{\mathbf{v}}_n}||_\infty \\
&= ||\mathbf{T}_n(\mathbf{D}_\mathbf{v}^{-1}|\tilde{\mathbf{R}}|\mathbf{D}_\mathbf{v})||_\infty \\
&\leq ||\mathbf{D}_\mathbf{v}^{-1}|\tilde{\mathbf{R}}|\mathbf{D}_\mathbf{v}||_\infty \\
&= \rho(|\tilde{\mathbf{R}}|).
\end{aligned}$$

2. Consider:

$$\mathbf{D}_{\bar{\mathbf{v}}_n}^{-1}|\tilde{\mathbf{R}}_n|^k\mathbf{D}_{\bar{\mathbf{v}}_n} = \mathbf{D}_{\bar{\mathbf{v}}_n}^{-1}|\tilde{\mathbf{R}}_n|^{k-1}\mathbf{D}_{\bar{\mathbf{v}}_n}\mathbf{D}_{\bar{\mathbf{v}}_n}^{-1}|\mathbf{R}_n|\mathbf{D}_{\bar{\mathbf{v}}_n}. \tag{27}$$

From Equation (27),

$$\begin{aligned}
||\mathbf{D}_{\bar{\mathbf{v}}_n}^{-1}|\tilde{\mathbf{R}}_n|^k\mathbf{D}_{\bar{\mathbf{v}}_n}||_\infty &\leq ||\mathbf{D}_{\bar{\mathbf{v}}_n}^{-1}|\tilde{\mathbf{R}}_n|^{k-1}\mathbf{D}_{\bar{\mathbf{v}}_n}||_\infty||\mathbf{D}_{\bar{\mathbf{v}}_n}^{-1}|\tilde{\mathbf{R}}_n|\mathbf{D}_{\bar{\mathbf{v}}_n}||_\infty \\
&\leq \rho(|\tilde{\mathbf{R}}|)||\mathbf{D}_{\bar{\mathbf{v}}_n}^{-1}|\tilde{\mathbf{R}}_n|^{k-1}\mathbf{D}_{\bar{\mathbf{v}}_n}||_\infty.
\end{aligned} \tag{28}$$

If we apply Inequality (28) recursively, we see that:

$$||\mathbf{D}_{\bar{\mathbf{v}}_n}^{-1}|\tilde{\mathbf{R}}_n|^k\mathbf{D}_{\bar{\mathbf{v}}_n}||_\infty \leq [\rho(|\tilde{\mathbf{R}}|)]^k,$$

and

$$\begin{aligned}
|||\tilde{\mathbf{R}}_n|^k||_\infty &= ||\mathbf{D}_{\bar{\mathbf{v}}_n}\mathbf{D}_{\bar{\mathbf{v}}_n}^{-1}|\tilde{\mathbf{R}}_n|^k\mathbf{D}_{\bar{\mathbf{v}}_n}\mathbf{D}_{\bar{\mathbf{v}}_n}^{-1}||_\infty \\
&\leq ||\mathbf{D}_{\bar{\mathbf{v}}_n}||_\infty||\mathbf{D}_{\bar{\mathbf{v}}_n}^{-1}||_\infty||\mathbf{D}_{\bar{\mathbf{v}}_n}^{-1}|\tilde{\mathbf{R}}_n|^k\mathbf{D}_{\bar{\mathbf{v}}_n}||_\infty \\
&\leq \frac{\max_i\{v_i\}}{\min_j\{v_j\}}[\rho(|\tilde{\mathbf{R}}|)]^k.
\end{aligned}$$

Setting $\kappa_1 = \frac{\max_i\{v_i\}}{\min_j\{v_j\}}$,

$$||\tilde{\mathbf{R}}_n^k||_\infty \leq |||\tilde{\mathbf{R}}_n|^k||_\infty \leq \kappa_1[\rho(|\tilde{\mathbf{R}}|)]^k.$$

3. By assumption, $\rho(|\tilde{\mathbf{R}}|) < 1$, and hence we can write $\tilde{\mathbf{T}}_n^{-1}$ as a Neumann power series:

$$\tilde{\mathbf{T}}_n^{-1} = \sum_{i=0}^{\infty} \tilde{\mathbf{R}}_n^i. \tag{29}$$

The series in Equation (29) implies:

$$\begin{aligned} ||\tilde{\mathbf{T}}_n^{-1}||_\infty &\leq \sum_{i=0}^{\infty} ||\tilde{\mathbf{R}}_n^i||_\infty \\ &\leq \kappa_1 \sum_{i=0}^{\infty} [\rho(|\tilde{\mathbf{R}}|)]^i \\ &= \frac{\kappa_1}{1-\rho(|\tilde{\mathbf{R}}|)} = \kappa_2. \end{aligned}$$

■

Appendix B. Proof of Theorem 2

Proof We focus, without loss of generality, on the computation tree for cluster 1. The proofs for the two components are:

1. Consider $\tilde{\boldsymbol{\mu}}_1^{(n-1)} = \mathbf{G}_n' \tilde{\mathbf{T}}_n^{-1} \mathbf{E}_n \tilde{\boldsymbol{b}}$. Since $\tilde{\mathbf{S}}$ is positive definite, there is one vector $\tilde{\boldsymbol{\mu}}$ in $\Re^k$ with the property $\tilde{\mathbf{S}}\tilde{\boldsymbol{\mu}} = \tilde{\boldsymbol{b}}$. Let us consider solving $\mathbf{z}_n$ in

$$\tilde{\mathbf{T}}_n \mathbf{z}_n = \mathbf{E}_n \tilde{\boldsymbol{b}}.$$

Since $\tilde{\mathbf{S}}\tilde{\boldsymbol{\mu}} = \tilde{\boldsymbol{b}}$, we have $\tilde{\mathbf{T}}_n \mathbf{z}_n = \mathbf{E}_n \tilde{\mathbf{S}} \tilde{\boldsymbol{\mu}}$. Substitution of Equation (9) into this expression yields $\tilde{\mathbf{T}}_n \mathbf{z}_n = (\tilde{\mathbf{T}}_n \mathbf{E}_n - \tilde{\mathbf{L}}_n)\tilde{\boldsymbol{\mu}}$, or

$$\mathbf{z}_n = \mathbf{E}_n \tilde{\boldsymbol{\mu}} - \tilde{\mathbf{T}}_n^{-1} \tilde{\mathbf{L}}_n \tilde{\boldsymbol{\mu}}.$$

Using the fact that $\tilde{\boldsymbol{\mu}}_1^{(n-1)} = \mathbf{G}_n' \mathbf{z}_n$, we have

$$\tilde{\boldsymbol{\mu}}_1^{(n-1)} = \mathbf{G}_n' \mathbf{E}_n \tilde{\boldsymbol{\mu}} - \mathbf{G}_n' \tilde{\mathbf{T}}_n^{-1} \tilde{\mathbf{L}}_n \tilde{\boldsymbol{\mu}} = \tilde{\boldsymbol{\mu}}_1 - \mathbf{G}_n' \tilde{\mathbf{T}}_n^{-1} \tilde{\mathbf{L}}_n \tilde{\boldsymbol{\mu}},$$

where $\tilde{\boldsymbol{\mu}}_1$ is the subvector of $\tilde{\boldsymbol{\mu}}$ corresponding to the variables in cluster 1. Since $\rho(\tilde{\mathbf{R}}_n) \leq \rho(|\tilde{\mathbf{R}}_n|) \leq \rho(|\tilde{\mathbf{R}}|) < 1$, by Lemma 4, we can express $\tilde{\mathbf{T}}_n^{-1}$ as a Neumann power series,

$$\tilde{\mathbf{T}}_n^{-1} = \sum_{i=0}^{\infty} \tilde{\mathbf{R}}_n^i.$$

Consider the vector $\tilde{\mathbf{L}}_n \tilde{\boldsymbol{\mu}}$. Since the rows of $\tilde{\mathbf{L}}_n$ corresponding to the first $n-1$ blocks are all equal to zero (see Lemma 2), we can write $\tilde{\mathbf{L}}_n \tilde{\boldsymbol{\mu}} = (\mathbf{0}_1', \mathbf{0}_2', \ldots, \mathbf{0}_{n-1}', \mathbf{h}_n')'$. The notation $\mathbf{0}_l$ stands for a vector containing a number of zeros equal to the sum of the dimensionalities of the nodes in layer l. The vector $\mathbf{h}_n$ is a specified vector (entries

can be non-zero) equal in size to the sum of the dimensionalities of the nodes in the terminal layer. With this notation, it is easy to see that

$$\mathbf{G}'_n \tilde{\mathbf{R}}_n^i \tilde{\mathbf{L}}_n \tilde{\boldsymbol{\mu}} = \mathbf{0}_1$$

for all $i < n-1$ (note that the first layer contains only one node corresponding to cluster 1, hence $\mathbf{0}_1$ will contain d_1 zeros). We have:

$$\begin{aligned}
\mathbf{G}'_n \tilde{\mathbf{T}}_n^{-1} \tilde{\mathbf{L}}_n \tilde{\boldsymbol{\mu}} &= \mathbf{G}'_n \sum_{i=0}^{\infty} \tilde{\mathbf{R}}_n^i \tilde{\mathbf{L}}_n \tilde{\boldsymbol{\mu}} \\
&= \mathbf{G}'_n \sum_{i=n-1}^{\infty} \tilde{\mathbf{R}}_n^i \tilde{\mathbf{L}}_n \tilde{\boldsymbol{\mu}} \\
&= \mathbf{G}'_n \tilde{\mathbf{R}}_n^{n-1} \sum_{i=0}^{\infty} \tilde{\mathbf{R}}_n^i \tilde{\mathbf{L}}_n \tilde{\boldsymbol{\mu}} \\
&= \mathbf{G}'_n \tilde{\mathbf{R}}_n^{n-1} \sum_{i=0}^{\infty} \tilde{\mathbf{R}}_n^i \tilde{\mathbf{L}}_n \tilde{\boldsymbol{\mu}} \\
&= \mathbf{G}'_n \tilde{\mathbf{R}}_n^{n-1} \tilde{\mathbf{T}}_n^{-1} \tilde{\mathbf{L}}_n \tilde{\boldsymbol{\mu}}. \qquad (30)
\end{aligned}$$

The rows with non-zero entries of $\tilde{\mathbf{L}}_n$ are (sparse) replicates of a row of $\tilde{\mathbf{S}}$ by Lemma 2, and we therefore can say that $||\tilde{\mathbf{L}}_n \tilde{\boldsymbol{\mu}}||_\infty \leq ||\tilde{\mathbf{S}}||_\infty ||\tilde{\boldsymbol{\mu}}||_\infty = \kappa_3$ (does not depend on n). Since $||\mathbf{G}'_n||_\infty = 1$, we have

$$||\tilde{\boldsymbol{\mu}}_1^{(n-1)} - \tilde{\boldsymbol{\mu}}_1||_\infty \leq \kappa_4 [\rho(|\tilde{\mathbf{R}}|)]^{n-1}, \qquad (31)$$

where $\kappa_4 = \kappa_1 \kappa_2 \kappa_3$. Hence $\tilde{\boldsymbol{\mu}}_1^{(n)}$ will converge to $\tilde{\boldsymbol{\mu}}_1$ as $n \to \infty$. We can repeat the above analysis for all clusters i.

2. Note that Inequality (31) also applies when $\tilde{\boldsymbol{b}}$ is a matrix. Due to the order of the variables (that is $\mathbf{X} = (\mathbf{X}'_1, \mathbf{X}'_2, \ldots, \mathbf{X}'_p)'$), we have that cluster 1 will contain the variables $1, 2, \ldots, d_1$ of the original pairwise MG. Let $\mathbf{e}_l : k \times 1$ be a vector of zeros except for entry l, which contains 1. Set $\tilde{\boldsymbol{b}} = \begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \ldots & \mathbf{e}_{d_1} \end{bmatrix}$. Using the convergence implied by (31), we see that

$$\mathbf{G}'_n \tilde{\mathbf{T}}_n^{-1} \mathbf{E}_n \tilde{\boldsymbol{b}} \to \lfloor \tilde{\mathbf{S}}^{-1} \rfloor_{11}$$

as $n \to \infty$, where $\lfloor \tilde{\mathbf{S}}^{-1} \rfloor_{11}$ is the submatrix of $\tilde{\mathbf{S}}^{-1}$ corresponding to the variables in cluster 1. The matrix $\mathbf{E}_n \tilde{\boldsymbol{b}}$ has a row-block decomposition according to the nodes in the computation tree. If a node in the computation tree does not reference cluster 1, then the corresponding block of $\mathbf{E}_n \tilde{\boldsymbol{b}}$ will be zero, otherwise the block is $\mathbf{I}_{d_1}$ (this is because $\mathbf{F}_t \mathbf{e}_l$, for $t \neq 1$ and $l \leq d_1$, extracts a subvector of $\mathbf{e}_l$ containing only zeros). As a consequence, we have $\mathbf{E}_n \tilde{\boldsymbol{b}} = \mathbf{G}_n + \bar{\mathbf{E}}_n$, where $\bar{\mathbf{E}}_n$ is equal to $\mathbf{E}_n \tilde{\boldsymbol{b}}$, except for the first block, which contains only zeros. We can write

$$[\tilde{\mathbf{P}}_1^{(n-1)}]^{-1} = \mathbf{G}'_n \tilde{\mathbf{T}}_n^{-1} \mathbf{G}_n \sim \lfloor \tilde{\mathbf{S}}^{-1} \rfloor_{11} - \mathbf{G}'_n \tilde{\mathbf{T}}_n^{-1} \bar{\mathbf{E}}_n$$

for large n. We now present the following lemma (proof follows in Section B.1).

Lemma 7 *If $\rho(|\tilde{\mathbf{R}}|) < 1$, then the sequence $\tilde{\mathbf{Z}}_n^{(1)} = \mathbf{G}_n' \tilde{\mathbf{T}}_n^{-1} \bar{\mathbf{E}}_n$ is a Cauchy sequence.*

Hence we have:

$$[\tilde{\mathbf{P}}_1^{(n-1)}]^{-1} \sim \lfloor \tilde{\mathbf{S}}^{-1} \rfloor_{11} - \tilde{\mathbf{Z}}_n^{(1)},$$

with $\tilde{\mathbf{Z}}_n^{(1)}$ forming a Cauchy sequence. Note that we can repeat this analysis for any cluster i.

3. Let t be the first layer, excluding layer 1, where cluster 1 is referenced in its computation tree. Note that the first $t-1$ row-blocks of $\bar{\mathbf{E}}_n$, corresponding to the line topology of the computation tree, will all be zero matrices. Hence we can write $\tilde{\mathbf{Z}}_n^{(1)} = \mathbf{G}_n' \mathbf{R}_n^{t-1} \tilde{\mathbf{T}}_n^{-1} \bar{\mathbf{E}}_n$. Since $||\mathbf{G}_n'||_\infty = 1$ and $||\bar{\mathbf{E}}_n||_\infty \leq 1$, then by Lemma 4,

$$\begin{aligned} ||\tilde{\mathbf{Z}}_n^{(1)}||_\infty &= ||\mathbf{G}_n' \mathbf{R}_n^{t-1} \tilde{\mathbf{T}}_n^{-1} \bar{\mathbf{E}}_n||_\infty \\ &\leq ||\mathbf{R}_n^{t-1}||_\infty ||\tilde{\mathbf{T}}_n^{-1}||_\infty \\ &\leq \kappa_1 \kappa_2 [\rho(|\tilde{\mathbf{R}}|)]^{t-1}. \end{aligned}$$

■

B.1. Proof of Lemma 7

Proof For convenience we write $\tilde{\mathbf{Z}}_n = \tilde{\mathbf{Z}}_n^{(1)}$. Consider the following recursive expansions:

$$\tilde{\mathbf{T}}_{n+1} = \begin{bmatrix} \tilde{\mathbf{T}}_n & \tilde{\mathbf{T}}_{1n} \\ \tilde{\mathbf{T}}_{n1} & \mathbf{U}_{nn} \end{bmatrix}$$

$$\bar{\mathbf{E}}_{n+1} = \begin{bmatrix} \bar{\mathbf{E}}_n \\ \mathbf{W}_n \end{bmatrix}$$

for certain matrices $\tilde{\mathbf{T}}_{1n}$, $\mathbf{U}_{nn}$ and $\mathbf{W}_n$. We have

$$\tilde{\mathbf{T}}_{n+1}^{-1} = \begin{bmatrix} \bar{\mathbf{T}}_n^{-1} & -\bar{\mathbf{T}}_n^{-1} \tilde{\mathbf{T}}_{1n} \mathbf{U}_{nn}^{-1} \\ \dots & \dots \end{bmatrix},$$

where the $\dots$ represents an irrelevant part of the matrix, and $\bar{\mathbf{T}}_n = \tilde{\mathbf{T}}_n - \tilde{\mathbf{T}}_{1n} \mathbf{U}_{nn}^{-1} \tilde{\mathbf{T}}_{n1}$. Consider

$$\begin{aligned} \tilde{\mathbf{Z}}_{n+1} &= \mathbf{G}_{n+1}' \tilde{\mathbf{T}}_{n+1}^{-1} \bar{\mathbf{E}}_{n+1} \\ &= \begin{bmatrix} \mathbf{G}_n' & \mathbf{0} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{T}}_n^{-1} & -\bar{\mathbf{T}}_n^{-1} \tilde{\mathbf{T}}_{1n} \mathbf{U}_{nn}^{-1} \\ \dots & \dots \end{bmatrix} \begin{bmatrix} \bar{\mathbf{E}}_n \\ \mathbf{W}_n \end{bmatrix} \\ &= \mathbf{G}_n' \bar{\mathbf{T}}_n^{-1} \bar{\mathbf{E}}_n - \mathbf{G}_n' \bar{\mathbf{T}}_n^{-1} \tilde{\mathbf{T}}_{1n} \mathbf{U}_{nn}^{-1} \mathbf{W}_n. \end{aligned} \tag{32}$$

It can be shown that

$$\begin{aligned} \bar{\mathbf{T}}_n^{-1} &= \tilde{\mathbf{T}}_n^{-1} + \tilde{\mathbf{T}}_n^{-1} \tilde{\mathbf{T}}_{1n} [\mathbf{U}_{nn} - \tilde{\mathbf{T}}_{n1} \tilde{\mathbf{T}}_n^{-1} \tilde{\mathbf{T}}_{1n}]^{-1} \tilde{\mathbf{T}}_{n1} \tilde{\mathbf{T}}_n^{-1} \\ &= \tilde{\mathbf{T}}_n^{-1} + \tilde{\mathbf{T}}_n^{-1} \tilde{\mathbf{T}}_{1n} \mathbf{V}_n \tilde{\mathbf{T}}_n^{-1}, \end{aligned}$$

where $\mathbf{V}_n = [\mathbf{U}_{nn} - \tilde{\mathbf{T}}_{n1}\tilde{\mathbf{T}}_n^{-1}\tilde{\mathbf{T}}_{1n}]^{-1}\tilde{\mathbf{T}}_{n1}$. Furthermore,

$$\begin{aligned}\mathbf{G}'_n\bar{\mathbf{T}}_n^{-1}\bar{\mathbf{E}}_n &= \mathbf{G}'_n\tilde{\mathbf{T}}_n^{-1}\bar{\mathbf{E}}_n + \mathbf{G}'_n\tilde{\mathbf{T}}_n^{-1}\tilde{\mathbf{T}}_{1n}\mathbf{V}_n\tilde{\mathbf{T}}_n^{-1}\bar{\mathbf{E}}_n \\ &= \tilde{\mathbf{Z}}_n + \mathbf{G}'_n\tilde{\mathbf{T}}_n^{-1}\tilde{\mathbf{T}}_{1n}\mathbf{V}_n\tilde{\mathbf{T}}_n^{-1}\bar{\mathbf{E}}_n.\end{aligned}$$

Considering again Equation (32),

$$\begin{aligned}\tilde{\mathbf{Z}}_{n+1} &= \tilde{\mathbf{Z}}_n + \mathbf{G}'_n\tilde{\mathbf{T}}_n^{-1}\tilde{\mathbf{T}}_{1n}\mathbf{V}_n\tilde{\mathbf{T}}_n^{-1}\bar{\mathbf{E}}_n - \mathbf{G}'_n\bar{\mathbf{T}}_n^{-1}\tilde{\mathbf{T}}_{1n}\mathbf{U}_{nn}^{-1}\mathbf{W}_n \\ &= \tilde{\mathbf{Z}}_n + \mathbf{G}'_n\tilde{\mathbf{T}}_n^{-1}\tilde{\mathbf{T}}_{1n}\mathbf{V}_n\tilde{\mathbf{T}}_n^{-1}\bar{\mathbf{E}}_n - \mathbf{G}'_n[\tilde{\mathbf{T}}_n^{-1} + \tilde{\mathbf{T}}_n^{-1}\tilde{\mathbf{T}}_{1n}\mathbf{V}_n\tilde{\mathbf{T}}_n^{-1}]\tilde{\mathbf{T}}_{1n}\mathbf{U}_{nn}^{-1}\mathbf{W}_n \\ &= \tilde{\mathbf{Z}}_n + \mathbf{G}'_n\tilde{\mathbf{T}}_n^{-1}\tilde{\mathbf{T}}_{1n}\mathbf{V}_n\tilde{\mathbf{T}}_n^{-1}\bar{\mathbf{E}}_n - \mathbf{G}'_n\tilde{\mathbf{T}}_n^{-1}\tilde{\mathbf{T}}_{1n}\mathbf{U}_{nn}^{-1}\mathbf{W}_n - \mathbf{G}'_n\tilde{\mathbf{T}}_n^{-1}\tilde{\mathbf{T}}_{1n}\mathbf{V}_n\tilde{\mathbf{T}}_n^{-1}\tilde{\mathbf{T}}_{1n}\mathbf{U}_{nn}^{-1}\mathbf{W}_n \\ &= \tilde{\mathbf{Z}}_n + \mathbf{G}'_n\tilde{\mathbf{T}}_n^{-1}\tilde{\mathbf{T}}_{1n}[\mathbf{V}_n\tilde{\mathbf{T}}_n^{-1}\bar{\mathbf{E}}_n - \mathbf{V}_n\tilde{\mathbf{T}}_n^{-1}\tilde{\mathbf{T}}_{1n}\mathbf{U}_{nn}^{-1}\mathbf{W}_n - \mathbf{U}_{nn}^{-1}\mathbf{W}_n] \\ &= \tilde{\mathbf{Z}}_n + \mathbf{G}'_n\tilde{\mathbf{T}}_n^{-1}\tilde{\mathbf{T}}_{1n}[\mathbf{V}_n\tilde{\mathbf{T}}_n^{-1}(\bar{\mathbf{E}}_n - \tilde{\mathbf{T}}_{1n}\mathbf{U}_{nn}^{-1}\mathbf{W}_n) - \mathbf{U}_{nn}^{-1}\mathbf{W}_n]. \qquad (33)\end{aligned}$$

Note that the sparsity pattern of $\tilde{\mathbf{T}}_{1n}$ is similar to that of $\tilde{\mathbf{L}}_n$ and contains zeros in all the rows, except for the rows corresponding to the terminal layer of the computation tree (of depth n). Similar to Equation (30), we can show that

$$\mathbf{G}'_n\tilde{\mathbf{T}}_n^{-1}\tilde{\mathbf{T}}_{1n} = \mathbf{G}'_n\tilde{\mathbf{R}}_n^{n-1}\tilde{\mathbf{T}}_n^{-1}\tilde{\mathbf{T}}_{1n},$$

and hence:

$$||\mathbf{G}'_n\tilde{\mathbf{T}}_n^{-1}\tilde{\mathbf{T}}_{1n}||_\infty \leq \kappa_1\kappa_2||\tilde{\mathbf{T}}_{1n}||_\infty[\rho(|\tilde{\mathbf{R}}|)]^{n-1}. \qquad (34)$$

By Equations (33) and (34), we see that:

$$||\tilde{\mathbf{Z}}_{n+1}-\tilde{\mathbf{Z}}_n||_\infty \leq \kappa_1\kappa_2||\tilde{\mathbf{T}}_{1n}||_\infty||\mathbf{V}_n\tilde{\mathbf{T}}_n^{-1}(\bar{\mathbf{E}}_n-\tilde{\mathbf{T}}_{1n}\mathbf{U}_{nn}^{-1}\mathbf{W}_n)-\mathbf{U}_{nn}^{-1}\mathbf{W}_n||_\infty\times[\rho(|\tilde{\mathbf{R}}|)]^{n-1}. \qquad (35)$$

We now make some comments on the matrices present in Equation (35).

1. The matrix $\tilde{\mathbf{T}}_{1n}$ represents the links between parents in layer n and their children in layer $n+1$. Note that a parent in layer n has links to nodes in layer $n+1$ corresponding to its neighbors only, hence $||\tilde{\mathbf{T}}_{1n}||_\infty \leq \max_{i\neq j}\{||\tilde{\mathbf{S}}_{ij}||_\infty\}$.

2. The matrix $\mathbf{U}_{nn}$ is block diagonal, with diagonal blocks corresponding to the cluster diagonal blocks of $\tilde{\mathbf{S}}$ referenced in the final layer of the computation tree (of depth $n+1$). Therefore, $||\mathbf{U}_{nn}^{-1}||_\infty \leq \max_i\{||\tilde{\mathbf{S}}_{ii}^{-1}||_\infty\}$ and $||\mathbf{U}_{nn}||_\infty \leq \max_i\{||\tilde{\mathbf{S}}_{ii}||_\infty\}$.

3. Because $||\tilde{\mathbf{T}}_s^{-1}||_\infty \leq \kappa_2$ for all s, we have $||\bar{\mathbf{T}}_n^{-1}||_\infty \leq \kappa_2$ since $\bar{\mathbf{T}}_n^{-1}$ is a block of $\tilde{\mathbf{T}}_{n+1}^{-1}$.

4. Consider $\mathbf{V}_n = \mathbf{U}_{nn}^{-1}\tilde{\mathbf{T}}_{n1} + \mathbf{U}_{nn}^{-1}\tilde{\mathbf{T}}_{n1}[\tilde{\mathbf{T}}_n - \tilde{\mathbf{T}}_{1n}\mathbf{U}_{nn}^{-1}\tilde{\mathbf{T}}_{n1}]^{-1}\tilde{\mathbf{T}}_{1n}\mathbf{U}_{nn}^{-1}\tilde{\mathbf{T}}_{n1} = \mathbf{U}_{nn}^{-1}\tilde{\mathbf{T}}_{n1} + \mathbf{U}_{nn}^{-1}\tilde{\mathbf{T}}_{n1}\bar{\mathbf{T}}_n^{-1}\tilde{\mathbf{T}}_{1n}\mathbf{U}_{nn}^{-1}\tilde{\mathbf{T}}_{n1}$. Since all the matrices in this expression have an infinity norm bounded by some constant which does not depend on n, there will also be a constant (independent of n) that bounds the infinity norm of $\mathbf{V}_n$.

5. Note that $||\bar{\mathbf{E}}_n||_\infty \leq 1$ and $||\mathbf{W}_n||_\infty \leq 1$ (recall the definition of $\bar{\mathbf{E}}_n$ in the proof of Theorem 2).

The above considerations imply the existence of a constant K (independent of n) such that $\kappa_1\kappa_2||\tilde{\mathbf{T}}_{1n}||_\infty||\mathbf{V}_n\tilde{\mathbf{T}}_n^{-1}(\bar{\mathbf{E}}_n - \tilde{\mathbf{T}}_{1n}\mathbf{U}_{nn}^{-1}\mathbf{W}_n) - \mathbf{U}_{nn}^{-1}\mathbf{W}_n||_\infty \leq K$. Therefore, $||\tilde{\mathbf{Z}}_{n+1} - \tilde{\mathbf{Z}}_n||_\infty \leq K[\rho(|\tilde{\mathbf{R}}|)]^{n-1}$, hence $\tilde{\mathbf{Z}}_n$ is a Cauchy sequence. ■

Appendix C. Examples for Section 3

In this appendix we give some examples for the considerations of Section 3. We use the following precision matrix and potential vector for our illustrations:

$$\mathbf{S} = \begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} & \mathbf{S}_{13} & \mathbf{0} \\ \mathbf{S}_{21} & \mathbf{S}_{22} & \mathbf{S}_{23} & \mathbf{S}_{24} \\ \mathbf{S}_{31} & \mathbf{S}_{32} & \mathbf{S}_{33} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_{42} & \mathbf{0} & \mathbf{S}_{44} \end{bmatrix}$$

and

$$\boldsymbol{b} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \boldsymbol{b}_3 \\ \boldsymbol{b}_4 \end{bmatrix}.$$

We consider a computation tree, of depth 4, constructed for cluster 1. The different topologies corresponding to the MG, the computation tree and the line topology are illustrated in Figure 2. The precision matrix and potential vector for the computation tree are given by:

$$\mathbf{T}_4 = \begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} & \mathbf{S}_{13} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{S}_{21} & \mathbf{S}_{22} & \mathbf{0} & \mathbf{S}_{23} & \mathbf{S}_{24} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{S}_{31} & \mathbf{0} & \mathbf{S}_{33} & \mathbf{0} & \mathbf{0} & \mathbf{S}_{32} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_{32} & \mathbf{0} & \mathbf{S}_{33} & \mathbf{0} & \mathbf{0} & \mathbf{S}_{31} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_{42} & \mathbf{0} & \mathbf{0} & \mathbf{S}_{44} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{S}_{23} & \mathbf{0} & \mathbf{0} & \mathbf{S}_{22} & \mathbf{0} & \mathbf{S}_{21} & \mathbf{S}_{24} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{S}_{13} & \mathbf{0} & \mathbf{0} & \mathbf{S}_{11} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{S}_{12} & \mathbf{0} & \mathbf{S}_{11} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{S}_{42} & \mathbf{0} & \mathbf{0} & \mathbf{S}_{44} \end{bmatrix}$$

$$\mathbf{t}_4 = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \boldsymbol{b}_3 \\ \boldsymbol{b}_3 \\ \boldsymbol{b}_4 \\ \boldsymbol{b}_2 \\ \boldsymbol{b}_1 \\ \boldsymbol{b}_1 \\ \boldsymbol{b}_4 \end{bmatrix}.$$

C.1. Special Matrices

Some special matrices were introduced in Section 3.1. The row extractor matrix for cluster 1 is $\mathbf{F}_1 = \begin{bmatrix} \mathbf{I}_{d_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$. Note that $\mathbf{F}_1\mathbf{S} = \begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} & \mathbf{S}_{13} & \mathbf{S}_{14} \end{bmatrix}$, and hence we are extracting the rows corresponding to cluster 1. The row extractor matrix for our running

example is:

$$\mathbf{E}_4 = \begin{bmatrix} \mathbf{F}_1 \\ \mathbf{F}_2 \\ \mathbf{F}_3 \\ \mathbf{F}_3 \\ \mathbf{F}_4 \\ \mathbf{F}_2 \\ \mathbf{F}_1 \\ \mathbf{F}_1 \\ \mathbf{F}_4 \end{bmatrix}.$$

The sixth node of the computation tree is the rightmost node in layer 3 in the middle graph of Figure 2. The node extractor matrix for this node is $[\mathbf{G}_4^{(6)}]' = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_{d_2} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$. We see that $[\mathbf{G}_4^{(6)}]'\mathbf{T}_4$ extracts the rows of $\mathbf{T}_4$ corresponding to node 6 in the computation tree. Lemma 2 can be validated using our running example:

$$\mathbf{T}_4\mathbf{E}_4 = \begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} & \mathbf{S}_{13} & \mathbf{0} \\ \mathbf{S}_{21} & \mathbf{S}_{22} & \mathbf{S}_{23} & \mathbf{S}_{24} \\ \mathbf{S}_{31} & \mathbf{S}_{32} & \mathbf{S}_{33} & \mathbf{0} \\ \mathbf{S}_{31} & \mathbf{S}_{32} & \mathbf{S}_{33} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_{42} & \mathbf{0} & \mathbf{S}_{44} \\ \mathbf{S}_{21} & \mathbf{S}_{22} & \mathbf{S}_{23} & \mathbf{S}_{24} \\ \mathbf{S}_{11} & \mathbf{S}_{12} & \mathbf{S}_{13} & \mathbf{0} \\ \mathbf{S}_{11} & \mathbf{S}_{12} & \mathbf{S}_{13} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_{42} & \mathbf{0} & \mathbf{S}_{44} \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{S}_{12} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\mathbf{S}_{13} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} = \mathbf{E}_4\mathbf{S} + \mathbf{L}_4.$$

C.2. Tree-pruning Procedure

We now illustrate the tree-pruning procedure discussed in Sections 3.2 and 3.3. A visual illustration is given in Figure 5.

In Step 1 we have the original computation tree. We select node 6 of the computation tree as our bereaved parent. To move to Step 2, we prune the children of node 6 and adjust the precision matrix and potential vector associated with node 6, as discussed in Section 3.2. If we perform inference on the tree in Step 2, then the marginal at the root node is equivalent to the corresponding marginal in Step 1. We repeat this process until all nodes in the final layer have been removed. We then proceed to the next layer and continue until we are left only with the root node, as in Step 6.

We consider how the computation tree preserves the message-passing structure for our running example. Consider moving from Step 1 to Step 2 in Figure 5. Noting that $\mathbf{P}_{12}^{(0)} = \mathbf{S}_{11}$

and $\mathbf{P}_{42}^{(0)} = \mathbf{S}_{44}$, we see that the tree-pruning procedure requires us to perform inference on:

$$\mathbf{M} = \begin{bmatrix} \mathbf{S}_{22} & \mathbf{S}_{21} & \mathbf{S}_{24} \\ \mathbf{S}_{12} & \mathbf{S}_{11} & \mathbf{0} \\ \mathbf{S}_{42} & \mathbf{0} & \mathbf{S}_{44} \end{bmatrix} = \begin{bmatrix} \mathbf{S}_{22} & \mathbf{S}_{21} & \mathbf{S}_{24} \\ \mathbf{S}_{12} & \mathbf{P}_{12}^{(0)} & \mathbf{0} \\ \mathbf{S}_{42} & \mathbf{0} & \mathbf{P}_{42}^{(0)} \end{bmatrix}$$
$$\mathbf{m} = \begin{bmatrix} \boldsymbol{b}_2 \\ \boldsymbol{b}_1 \\ \boldsymbol{b}_4 \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_2 \\ \mathbf{z}_{12}^{(0)} \\ \mathbf{z}_{42}^{(0)} \end{bmatrix}.$$

Equations (16) and (17) imply that $r_P(\mathcal{M}) = \mathbf{P}_{23}^{(1)}$ and $r_\mu(\mathcal{M}) = [\mathbf{P}_{23}^{(1)}]^{-1}\mathbf{z}_{23}^{(1)}$. We remove nodes 8 and 9 of the computation tree (1 and 4 in the red rectangle) and adjust the potential and precision of node 6 (2 in red rectangle) to $r_P(\mathcal{M})r_\mu(\mathcal{M}) = \mathbf{z}_{23}^{(1)}$ and $r_P(\mathcal{M}) = \mathbf{P}_{23}^{(1)}$ respectively. The potential vector and precision matrix of the trimmed tree are given in Equations (36) and (37) (the vector and matrix to the right of the arrows indicate the potential vector and precision matrix of the tree obtained after the next pruning step):

$$\begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \boldsymbol{b}_3 \\ \boldsymbol{b}_3 \\ \boldsymbol{b}_4 \\ \mathbf{z}_{23}^{(1)} \\ \boldsymbol{b}_1 \end{bmatrix} \rightarrow \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \boldsymbol{b}_3 \\ \mathbf{z}_{32}^{(1)} \\ \mathbf{z}_{42}^{(1)} \\ \mathbf{z}_{23}^{(1)} \end{bmatrix} \tag{36}$$

$$\begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} & \mathbf{S}_{13} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{S}_{21} & \mathbf{S}_{22} & \mathbf{0} & \mathbf{S}_{23} & \mathbf{S}_{24} & \mathbf{0} & \mathbf{0} \\ \mathbf{S}_{31} & \mathbf{0} & \mathbf{S}_{33} & \mathbf{0} & \mathbf{0} & \mathbf{S}_{32} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_{32} & \mathbf{0} & \mathbf{S}_{33} & \mathbf{0} & \mathbf{0} & \mathbf{S}_{31} \\ \mathbf{0} & \mathbf{S}_{42} & \mathbf{0} & \mathbf{0} & \mathbf{S}_{44} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{S}_{23} & \mathbf{0} & \mathbf{0} & \mathbf{P}_{23}^{(1)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{S}_{13} & \mathbf{0} & \mathbf{0} & \mathbf{S}_{11} \end{bmatrix} \rightarrow \begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} & \mathbf{S}_{13} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{S}_{21} & \mathbf{S}_{22} & \mathbf{0} & \mathbf{S}_{23} & \mathbf{S}_{24} & \mathbf{0} \\ \mathbf{S}_{31} & \mathbf{0} & \mathbf{S}_{33} & \mathbf{0} & \mathbf{0} & \mathbf{S}_{32} \\ \mathbf{0} & \mathbf{S}_{32} & \mathbf{0} & \mathbf{P}_{32}^{(1)} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_{42} & \mathbf{0} & \mathbf{0} & \mathbf{P}_{42}^{(1)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{S}_{23} & \mathbf{0} & \mathbf{0} & \mathbf{P}_{23}^{(1)} \end{bmatrix}. \tag{37}$$

Note that, due to the specific sparsity pattern, we can write $\mathbf{z}_{42}^{(1)} = \mathbf{z}_{42}^{(0)} = \boldsymbol{b}_2$ and $\mathbf{P}_{42}^{(1)} = \mathbf{P}_{42}^{(0)} = \mathbf{S}_{44}$. We can repeat this process until we are left with the following potential vector and precision matrix:

$$\begin{bmatrix} \boldsymbol{b}_1 \\ \mathbf{z}_{21}^{(2)} \\ \mathbf{z}_{31}^{(2)} \end{bmatrix}; \begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} & \mathbf{S}_{13} \\ \mathbf{S}_{21} & \mathbf{P}_{21}^{(2)} & \mathbf{0} \\ \mathbf{S}_{31} & \mathbf{0} & \mathbf{P}_{31}^{(2)} \end{bmatrix}.$$

We then apply one more pruning step in order to obtain:

$$\begin{aligned} r_P(\mathcal{T}_4) &= \mathbf{S}_{11} - \mathbf{S}_{12}\mathbf{P}_{21}^{(2)}\mathbf{S}_{21} - \mathbf{S}_{13}\mathbf{P}_{31}^{(2)}\mathbf{S}_{31} = \mathbf{S}_{11} + \mathbf{Q}_{21}^{(3)} + \mathbf{Q}_{31}^{(3)} = \mathbf{P}_1^{(3)} \\ r_\mu(\mathcal{T}_4) &= [\mathbf{P}_1^{(3)}]^{-1}[\boldsymbol{b}_1 - \mathbf{S}_{12}[\mathbf{P}_{21}^{(2)}]^{-1}\mathbf{z}_{21}^{(2)} - \mathbf{S}_{13}[\mathbf{P}_{31}^{(2)}]^{-1}\mathbf{z}_{31}^{(2)}] \\ &= [\mathbf{P}_1^{(3)}]^{-1}[\boldsymbol{b}_1 + \mathbf{v}_{21}^{(2)} + \mathbf{v}_{31}^{(2)}] = \boldsymbol{\mu}_1^{(3)}. \end{aligned}$$

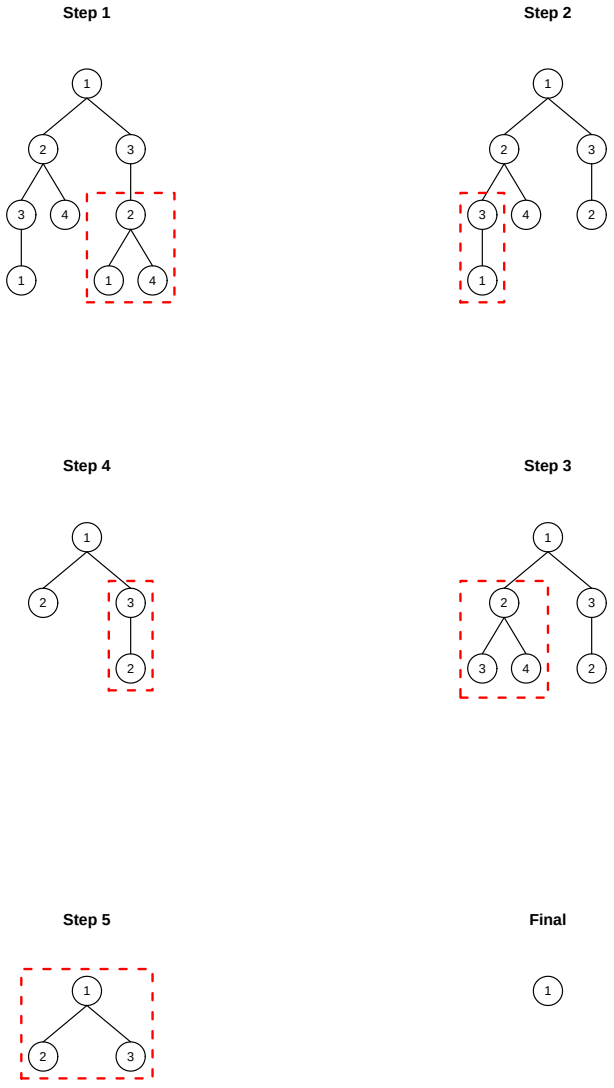

Figure 5: Visualization of the process used to invert a tree-structured precision matrix/solve a linear system through node pruning, where the block of the inverse/solution of the root node is of interest.

References

Danny Bickson. *Gaussian Belief Propagation: Theory and Application*. PhD thesis, The Hebrew University of Jerusalem, October 2008.

Venkat Chandrasekaran, Jason K. Johnson, and Alan S. Willsky. Estimation in Gaussian graphical models using tractable subgraphs: a walk-sum analysis. *IEEE Transactions on Signal Processing*, 56:1916–1930, 2008.

Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2):407–499, 2004.

Yousef El-Kurdi, Dennis Giannacopoulos, and Warren J. Gross. Relaxed Gaussian belief propagation. In *Proceedings IEEE International Symposium on Information Theory*, September 2012a.

Yousef El-Kurdi, Warren J. Gross, and Dennis Giannacopoulos. Efficient implementation of Gaussian belief propagation solver for large sparse diagonally dominant linear systems. *IEEE Transactions on Magnetics*, 48:471–474, February 2012b.

Robert G. Gallager. *Low-Density Parity-Check Codes*. MA: MIT Press, Cambridge, 1963.

Qinghua Guo and Defeng Huang. EM-based joint channel estimation and detection for frequency selective channels using Gaussian message passing. *IEEE Transactions on Signal Processing*, 59:4030–4035, 2011.

Qinghua Guo and Li Ping. LMMSE turbo equalization based on factor graphs. *IEEE Journal on Selected Areas in Communications*, 26:311–319, 2008.

Jason K. Johnson, Danny Bickson, and Danny Dolev. Fixing convergence of Gaussian belief propagation. In *Proceedings IEEE International Symposium on Information Theory*, Seoul, South Korea, 2009.

Francois Kamper. An empirical study of Gaussian belief propagation and application in the detection of F-formations. In *Proceedings of the ACM Multimedia 2017 Workshop on South African Academic Participation*, October 2017.

Francois Kamper, Johan A. du Preez, Sarel J. Steel, and Stephan Wagner. Regularized Gaussian belief propagation. *Statistics and Computing*, 28(3):653–672, May 2018.

Ying Liu. Feedback message passing for inference in Gaussian graphical models. Master's thesis, Massachusetts Institute of Technology, June 2010.

Ying Liu, Venkat Chandrasekaran, Animashree Anandkumar, and Alan S. Willsky. Feedback message passing for inference in Gaussian graphical models. *IEEE Transactions on Signal Processing*, 60(8):4135–4150, 2012.

Dmitry M. Malioutov. *Approximate inference in Gaussian graphical models*. PhD thesis, MIT, 2008.

Dmitry M. Malioutov, Jason K. Johnson, and Alan S. Willsky. Walk-sums and belief propagation in Gaussian graphical models. *Journal of Machine Learning Research*, 7: 2031–2064, October 2006.

Ciamac C. Moallemi and Benjamin Van Roy. Convergence of min-sum message passing for quadratic optimization. *IEEE Transactions on Information Theory*, 55(5):2413–2423, May 2009.

Ciamac C. Moallemi and Benjamin Van Roy. Convergence of min-sum message passing for convex optimization. *IEEE Transactions on Information Theory*, 56(4):2041–2050, April 2010.

Andrea Montanari, Balaji Prabhakar, and David Tse. Belief propagation based multi-user detection. In *Proceedings IEEE Information Theory Workshop*, Punta del Este, Uruguay, March 2006.

Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference.* Morgan Kaufmann, San Francisco, CA, USA, 1988.

Nicholas Ruozzi and Sekhar Tatikonda. Message-passing algorithms for quadratic minimization. *Journal of Machine Learning Research*, 14:2287–2314, 2013.

Nicholas Ruozzi, Justin Thaler, and Sekhar Tatikonda. Graph covers and quadratic minimization. In *Proceedings of the 47th Annual Allerton Conference on Communication, Control, and Computing*, Allerton, IL, September 2009.

Matthias W. Seeger and David P. Wipf. Variational Bayesian inference techniques. *IEEE Signal Processing Magazine*, 27:81–91, November 2010.

Ori Shental, Paul H. Siegel, Jack K. Wolf, Danny Bickson, and Danny Dolev. Gaussian belief propagation solver for systems of linear equations. In *Proceedings IEEE International Symposium on Information Theory*, pages 1863–1867, 2008.

Jonathan R. Shewchuk. *An introduction to the conjugate gradient method without the agonizing pain.* School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, August 4 1994.

Qinliang Su and Yik-Chung Wu. On convergence conditions of Gaussian belief propagation. *IEEE International Transactions on Signal Processing*, 63:1144–1155, March 2015.

Tianju Sui, Damian E. Marelli, and Minyue Fu. Convergence analysis of Gaussian belief propagation for distributed state estimation. In *Proceedings IEEE Annual Conference on Decision and Control*, Osaka, Japan, Dec. 2015.

Yair Weiss and William T. Freeman. Correctness of belief propagation in Gaussian graphical models of arbitrary topology. *Neural Computation*, 13(10):2173–2200, 2001.