# Gaussian belief propagation solvers for nonsymmetric systems of linear equations

Vladimir Fanaskov\*

Center for Design, Manufacturing, and Materials, Skolkovo Institute of Science and Technology, Bolshoy Boulevard 30, bld. 1, Moscow, Russia, 121205.

#### Abstract

In this paper, we argue for the utility of deterministic inference in the classical problem of numerical linear algebra, that of solving a linear system. We show how the Gaussian belief propagation solver, known to work for symmetric matrices can be modified to handle nonsymmetric matrices. Furthermore, we introduce a new algorithm for matrix inversion that corresponds to the generalized belief propagation derived from the cluster variation method (or Kikuchi approximation). We relate these algorithms to LU and block LU decompositions and provide certain guarantees based on theorems from the theory of belief propagation. All proposed algorithms are compared with classical solvers (e.g., Gauss-Seidel, BiCGSTAB) with application to linear elliptic equations. We also show how the Gaussian belief propagation can be used as multigrid smoother, resulting in a substantially more robust solver than the one based on the Gauss-Seidel iterative method.

## 1 Introduction

A basic problem of numerical linear algebra is to solve a linear equation

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{1}$$

with an invertible matrix **A**. The textbook technique is the LU decomposition, equivalent to Gaussian elimination [11, ch. 3]. However, when **A** is large and sparse, algorithms that exploit sparsity are used instead of the direct elimination [8]. Among iterative methods for sparse systems, one can mention classical relaxation techniques such as Gauss-Seidel (GS), Jacobi, Richardson, and projection methods such as conjugate gradients, generalized minimal residuals, biconjugate gradients, and others [25, 24].

<sup>\*</sup>Vladimir.Fanaskov@skoltech.ru

An easy way to understand the projection methods is to reformulate the original equation as an optimization problem [27]. For example, for a symmetric positive-definite matrix  $\mathbf{A}$ , one has

$$\mathbf{x}^{\star} = \arg\min_{\mathbf{x}} \left( \frac{\mathbf{x}^{T} \mathbf{A} \mathbf{x}}{2} - \mathbf{x}^{T} \mathbf{b} \right).$$
(2)

Such a reformulation allows one to apply new techniques and leads to methods of steepest descent, conjugate directions, and cheap and efficient conjugate gradients [14].

Another reformulation of the problem is known, but is less explored. It also goes back to Gauss and his version of elimination. To derive an LU solution of (1), one can consider the probability density function  $p(\mathbf{x})$  of multivariate normal distribution (see also equation (13) below)

$$p(\mathbf{x}) \sim \exp\left(-\frac{\mathbf{x}^T \mathbf{A} \mathbf{x}}{2} + \mathbf{b}^T \mathbf{x}\right).$$
 (3)

We can consider the first component  $x_1$  of  $\mathbf{x}$  and integrate it out in (3) (a process called "marginalization"). The resulting marginal distribution for the remaining components  $x_2, x_3, \ldots$  is again multivariate normal, but with the covariance matrix given by the Schur complement of  $A_{11}^{1}$  and the mean vector modified accordingly, i.e.

$$\mathbf{A_{22}} \leftarrow \mathbf{A_{22}} - \frac{\mathbf{A_{21}A_{12}}}{A_{11}}, \ \mathbf{b_2} \leftarrow \mathbf{b_2} - \frac{\mathbf{A_{21}b_1}}{A_{11}}.$$
 (4)

It is well known that the LU decomposition consists of the very same steps [28]. When  $x_1$  is not a scalar, but a subset of variables, marginalization of multivariate normal distribution results in a block LU decomposition.

Thus, the most popular direct technique for the solution of linear equations with dense matrices is intimately connected with the marginalization problem, which belongs to the class of inference problems. Recently, many other intriguing connections between statistical inference and linear algebra have been pointed out. For instance, in [7], [12], and [4], the authors provide a method to recover the Petrov-Galerkin condition from the Bayesian update and construct a Bayesian version of the conjugate gradients. Paper [19] constructs a state-of-the-art multigrid solver using the game theory and statistical inference. These works demonstrate that ideas from statistical inference allow for new and useful insights into problems of linear algebra. It is thus reasonable to explore how other inference algorithms are translated to the realm of numerical linear algebra. Among them are expectation propagation [18], Markov chain Monte Carlo, mean field, other variational Bayesian approximations [6, chapters 8, 10],

<sup>&</sup>lt;sup>1</sup>In the article we use boldface for matrices or matrix blocks, and regular font for scalar values and matrix components. In this case  $A_{11}$  is an element of the matrix **A** in the first row and the first column, and **A**<sub>22</sub> is a square matrix that contains all elements of **A** excluding the first row and the first column.

[32], and belief propagation with its generalized counterparts. The latter two are the focus of the present work.

The first comparison between classical methods and belief propagation appeared in [33]. Then in [26], authors argued explicitly for the belief propagation as a solver and later, Bickson [5] presented a more systematic treatment of the Gaussian belief propagation (GaBP) in the same context. Among other proposed methods was an algorithm that treats nonsymmetric matrices through diagonal weighting [5, 5.4] and the usual trick from linear algebra,  $\mathbf{A} \to \mathbf{A}^T \mathbf{A}$ . Both techniques are of limited use because of slow convergence in the first case and fill-in in the second. We improve on these results and propose several new algorithms.

In particular, in this work we:

- explain how belief propagation can be applied to nonsymmetric matrices with no computation overhead compared to the original belief propagation (which was limited to symmetric matrices) (Algorithm 1);
- design a family of linear solvers based on the generalized belief propagation (Algorithm 2) and relate them to the block LU decomposition (see Section 3.3);
- introduce a two-layer region graph and derive generalized belief propagation rules (36) that are substantially less demanding computationally compared to the basic generalized belief propagation algorithm (see (44));
- show how proofs of sufficient condition for convergence and consistency for the original GaBP can be modified to hold for the new algorithms (for GaBP see A, B, for generalized GaBP - C, D);
- explain how one can speed up GaBP using multigrid methodology which results in a robust solver (see Figure 4);
- implement the new algorithms [1] and benchmark them against several classical multigrid solvers.

The rest of the paper is organized as follows. In Section 2.1, we start with the intuitive explanation of GaBP based on the connection between the algorithm and Gaussian elimination. The general description of how to treat the problem (1) as an inference problem together with the basic terminology and main facts about Gaussian belief propagation are introduced in Section 2. In Section 2.3 we introduce GaBP that can be used for nonsymmetric matrices, prove consistency of the proposed algorithm, and establish a sufficient condition for convergence in appendices A, B. In Section 3, extensions of the belief propagation are given: in 3.1, we describe the generalized belief propagation algorithm (GaBP) (parent-to-child in [39]); in 3.2, we derive message update rules for region graphs with two layers, and in 3.3, we discuss the generalized GaBP from the elimination perspective and explain why the algorithm can be applied to the case  $\mathbf{A}^T \neq \mathbf{A}$ ; the resulting algorithm is introduced in Section 3.4 and analyzed in C, D.



Figure 1: Both (a) and (b) sketch the graph, corresponding to the matrix **A** from equation (5). We use  $M_{ji}$  to represent the pair of messages  $(\Lambda_{ji}, \mu_{ji})$  (see equation (8)) from the node *i* to *j*. Figures (a) and (b) shows different order of elimination. For example, in case of (a) one first exclude  $x_1$  and  $x_2$  from the equation for  $x_3$  and then solve resulting equation to obtain  $x_5$ .

Section 4 explains how to use GaBP within a multigrid scheme, we discuss smoothing properties, complexity and describe a rather unusual behaviour for singularly perturbed elliptic equations. Section 5 contains numerical examples. In Section 6, we summarize the main results and discuss possible future research.

## 2 Gaussian belief propagation

#### 2.1 GaBP from the elimination perspective

To build intuition about GaBP, we consider connection with the Gaussian elimination first. Ideas of this section are similar to those in [22], but the presentation is more straightforward and after appropriate modifications (see Section 2.3), applies to non-symmetric matrices as well.

To illustrate the main ideas, consider a linear problem  $\mathbf{A}\mathbf{x} = \mathbf{b}$  with the matrix and right-hand side defined as

$$\mathbf{A} = \begin{pmatrix} A_{11} & 0 & A_{13} & 0 & 0\\ 0 & A_{22} & A_{23} & 0 & 0\\ A_{31} & A_{32} & A_{33} & 0 & A_{35}\\ 0 & 0 & 0 & A_{44} & A_{45}\\ 0 & 0 & A_{53} & A_{54} & A_{55} \end{pmatrix}, \ \mathbf{b} \in \mathbb{R}^5.$$
(5)

For simplicity, require that  $\mathbf{A}$  be positive definite and that all elements of  $\mathbf{A}$ , not explicitly indicated as zeros, are nonzero.

To obtain GaBP rules, we introduce a graph of the matrix (5). For this section, it is sufficient to associate the set of vertices with the set of diagonal terms and the collection of edges with nonzero entries  $A_{ij}$ ,  $i \neq j$ . One can

find the resulting graph in figure 1. The correspondence between graphs and matrices is discussed in detail in two subsequent subsections 2.2, 2.3.

Suppose one wants to calculate variable  $x_5$ . To do that, we exclude variables  $x_1$ ,  $x_2$  from the equation for  $x_3$  and then eliminate variables  $x_3$ ,  $x_4$  from the equation for  $x_5$ 

$$\underbrace{\left(A_{33} - \frac{A_{31}A_{13}}{A_{11}} - \frac{A_{32}A_{23}}{A_{22}}\right)}_{=\tilde{A}_{33}} x_3 + A_{35}x_5 = \underbrace{b_3 - \frac{A_{31}b_1}{A_{11}} - \frac{A_{32}b_2}{A_{22}}}_{=\tilde{b}_3}; \quad (6a)$$

$$\left(A_{55} - \frac{A_{53}A_{35}}{\widetilde{A}_{33}} - \frac{A_{54}A_{45}}{A_{44}}\right)x_5 = b_5 - \frac{A_{53}\widetilde{b}_3}{\widetilde{A}_{33}} - \frac{A_{54}b_4}{A_{44}}.$$
(6b)

Figure 1a captures this particular elimination order. In the same vein, to find  $x_3$  one may follow the order presented in figure 1b. The resulting equations are

$$\underbrace{\begin{pmatrix} A_{55} - \frac{A_{54}A_{45}}{A_{44}} \end{pmatrix}}_{=\tilde{A}_{55}} x_5 + A_{53}x_3 = \underbrace{b_5 - \frac{A_{53}b_4}{A_{44}}}_{=\tilde{b}_5};$$

$$\begin{pmatrix} A_{33} - \frac{A_{35}A_{53}}{\tilde{A}_{55}} - \frac{A_{31}A_{13}}{A_{11}} - \frac{A_{32}A_{23}}{A_{22}} \end{pmatrix} x_3 = b_3 - \frac{A_{31}b_1}{A_{11}} - \frac{A_{32}b_2}{A_{22}} - \frac{A_{35}\tilde{b}_5}{\tilde{A}_{55}}.$$
(7a)

 $A_{55}$  (7b)

From these calculations, one can make two observations:

- 1. In the course of elimination one successively changes the diagonal elements  $A_{jj}$  and the right-hand side  $b_j$ .
- 2. The exclusion schemes in figures 1a and 1b share the same computations. For example, terms  $A_{31}A_{13}/A_{11}$ ,  $A_{32}A_{23}/A_{22}$  and  $A_{54}A_{45}/A_{44}$  appear on the way to equation (7b) as well as to (6b). It would be more advantageous to reuse the same computations, not to redo them each time one needs to eliminate a variable.

The first observation suggests that one can introduce corrections to the diagonal terms and  $b_j$ , that come from the elimination of variable *i*. For the sake of convenience, we denote them  $\Lambda_{ji}$  and  $\mu_{ji}\Lambda_{ji}$ , respectively. For example, equation (6a) becomes

$$(A_{33} + \Lambda_{31} + \Lambda_{32}) x_3 + A_{35} x_5 = b_3 + \Lambda_{32} \mu_{32} + \Lambda_{31} \mu_{31}.$$
 (8)

Since corrections are the same for any order of elimination, to reuse them, we can regard  $\Lambda_{ji}$  and  $\mu_{ji}$  as a message that node *i* sends to node *j* along the edge of the graph. Once computed, these messages are in use in expressions like (8) and (9). To complete rewriting the elimination in terms of messages, one needs

to introduce the rules to update messages when a new variable is excluded. To derive the rules, we rewrite equation (7b) using the definition of messages

$$(A_{55} + \Lambda_{53} + \Lambda_{54}) x_5 = b_5 + \Lambda_{53} \mu_{53} + \Lambda_{54} \mu_{54}, \tag{9}$$

and use (8) to get

$$\Lambda_{53} = -\frac{A_{35}A_{53}}{A_{33} + \Lambda_{31} + \Lambda_{32}}, \ \mu_{53} = \frac{b_3 + \Lambda_{31}\mu_{31} + \Lambda_{32}\mu_{32}}{A_{35}}.$$
 (10)

It is easy to see that one needs to accumulate all messages from neighbors of i except for j to send the message from node i to node j. Since the update rule includes only messages from the previous stages of elimination, one can iterate equations like (10) till convergence. And then, when all messages arrive, the solution can be read off as follows

$$x_j = \frac{b_j + \sum\limits_{k \in \text{neighbours of } j} \Lambda_{jk} \mu_{jk}}{A_{jj} + \sum\limits_{k \in \text{neighbours of } j} \Lambda_{jk}}.$$
(11)

Note that (7) and (6) have exactly this form. Equations for the update of messages that we deduced in this section coincide with the GaBP update rules given by (19), which are derived from the probabilistic perspective below.

To summarize, the GaBP rules can be understood as a scheme that propagates messages on the graph, corresponding to the matrix of the linear system under consideration. These messages, namely  $\Lambda_{ji}$  and  $\mu_{ji}$ , represent the corrections to the diagonal terms of matrix **A** and to the right-hand side **b**, resulting from the elimination of variable  $x_i$  from the *j*-th equation,  $A_{jj}x_j + A_{ji}x_i + \cdots = b_j$ . Consistency, convergence, stopping criteria, and other practical matters are discussed in the following two sections.

### 2.2 Conventional belief propagation

Here we give a more traditional introduction to GaBP as a technique for statistical inference in graphical models. Following [5, 26], we reformulate (1) as an inference problem. For this purpose, consider a small subset of undirected graph models that are known as Gauss-Markov random fields. First, we define a pairwise Markov random field. The graph  $\Gamma$  is the set of edges  $\mathcal{E}$  and vertices  $\mathcal{V}$ . Each vertex *i* corresponds to the random variable  $x_i$  (discrete or continuous), and each edge corresponds to interactions between variables. The set of non-negative integrable functions  $\{\phi_i, \psi_{ij}\}$  together with the graph  $\Gamma$ completely specifies the form of the probability density function of a pairwise Markov random field

$$p(x) = \frac{1}{Z} \prod_{i \in \mathcal{V}} \phi_i(x_i) \prod_{(ij) \in \mathcal{E}} \psi_{ij}(x_i, x_j) \equiv \frac{1}{Z} \exp\left(-E(\mathbf{x})\right).$$
(12)

Here, Z is a normalization constant (known in statistical physics as a partition function). The second equation in (12) (that is, the Gibbs distribution) should

be considered as a definition of energy  $E(\mathbf{x})$ . The Gauss-Markov random field is a particular instance of a pairwise Markov model with a joint probability density function given by a multivariate normal distribution

$$p(\mathbf{x}) = \frac{1}{Z} \exp\left(-\frac{\mathbf{x}^T \mathbf{A} \mathbf{x}}{2} + \mathbf{b}^T \mathbf{x}\right) \equiv \mathcal{N}\left(\mathbf{x} | \mathbf{A}_{\text{mean}}^{-1} \mathbf{b}, \mathbf{A}_{\text{covariance}}^{-1}\right), \quad (13)$$

where **A** is a symmetric positive-definite matrix. The edges of  $\Gamma$  correspond to the nonzero  $A_{ij}$ , and note that the splitting of the product in (12) into  $\phi_i$  and  $\psi_{ij}$  is not unique. A common task in the inference process is a computation of a partial distribution (or a marginalization)

$$p_r(\mathbf{x}_r) = \sum_{\mathbf{x} \setminus \mathbf{x}_r} p(\mathbf{x}).$$
(14)

Integrals replace sums if  $\mathbf{x} \in \mathbb{R}^N$ . For the Gauss-Markov model, marginal distributions are known explicitly. For individual components of the vector  $\mathbf{x}$ , which is distributed according to (13), one can obtain distributions in closed form

$$p_i(x_i) = \mathcal{N}\left(x_i | \left(\mathbf{A}^{-1}\mathbf{b}\right)_i, \left(\mathbf{A}^{-1}\right)_{ii}\right) \equiv \mathcal{N}\left(x_i | \mu_i, \beta_i\right).$$
(15)

As the means of marginal distributions for the model (13) coincide with the elements of the solution vector for (1), methods from the domain of probabilistic inference can be applied directly to obtain the solution.

A popular algorithm that exploits the structure of the underlying graph to find the marginal distribution efficiently is Pearl's belief propagation [20]. Pearl's algorithm operates with local messages that spread from node to node along the graph edges, and beliefs (approximate or exact marginals) are computed as a normalized product of all incoming messages after the convergence. More precisely, belief propagation consists of (i) the message update rule

$$m_{ij}(x_j) \leftarrow \sum_{x_i} \phi_i(x_i) \psi_{ij}(x_i, x_j) \prod_{k \in N(i) \setminus j} m_{ki}(x_i),$$
(16)

where  $m_{ij}$  is a message from node *i* to node *j* and N(i) is the set of neighbors of the node *i*, and (ii) the formula for marginals

$$b_i(x_i) \sim \phi_i(x_i) \prod_{k \in N(i)} m_{ki}(x_i).$$
(17)

Although, for continuous random variables the problem of marginalization and the algorithm of belief propagation are harder in general, it is not the case for the normal distribution. Namely, for the Gauss-Markov model, one can parameterize messages in the form of the normal distribution

$$m_{ji}(x_i) \sim \exp\left(-\frac{\Lambda_{ji} \left(x_i - \mu_{ji}\right)^2}{2}\right),$$
(18)

and explicitly derive update rules, means, and precision

$$\mu_{ji}^{(n+1)} = \frac{b_j + \sum_{k \in N(j) \setminus i} \Lambda_{kj}^{(n)} \mu_{kj}^{(n)}}{A_{ji}}, \ \Lambda_{ji}^{(n+1)} = -\frac{A_{ij}A_{ji}}{A_{jj} + \sum_{k \in N(j) \setminus i} \Lambda_{kj}^{(n)}};$$

$$\mu_i^{(n)} = \frac{b_i + \sum_{j \in N(i)} \Lambda_{ji}^{(n)} \mu_{ji}^{(n)}}{A_{ii} + \sum_{j \in N(i)} \Lambda_{ji}^{(n)}}, \ \beta_i^{(n)} = A_{ii} + \sum_{j \in N(i)} \Lambda_{ji}^{(n)}.$$
(19)

These update rules correspond to the flood schedule such that at the current iteration step, each node sends messages to all its neighbours based on messages received at the previous step. Equations for the mean and precision should be put to use only after saturation according to some criteria, for example  $|\mu^{(n+1)} - \mu^{(n)}| \leq$  tolerance, and the same for  $\Lambda$ . Rules (19) are collectively known as Gaussian belief propagation.

Belief propagation was designed to give an exact answer if  $\Gamma$  has no loops. In the presence of loops, the result appears to be approximate if delivered at all. In the case of GaBP, the situation is more optimistic. We briefly recall some useful facts about GaBP that we discuss later in more detail. If GaBP converges on the graph of arbitrary topology, the means are exact, but variances can be incorrect [33]. The best sufficient condition for convergence of the Gauss-Markov model with symmetric positive-definite matrix can be found in [17], we discuss it later in greater detail. The fixed point of GaBP is unique [13]. On the tree, GaBP is equivalent to the Gaussian elimination [22].

Many different schemes that extend belief propagation and GaBP have been developed [38], [9], [29], [18], [10]. Here, we are mainly interested in generalized belief propagation proposed in [38] and subsequently developed in [37], [36], [39]. This new algorithm is significantly more accurate [39, Fig. 15] than Pearl's algorithm, but at the same time it can be computationally costly. In what follows, we show how to use the generalized belief propagation in the context of numerical linear algebra.

## 2.3 GaBP for a nonsymmetric linear system

As explained in Section 2.1, the GaBP rules can be understood as corrections to the right-hand side and the diagonal elements of the matrix under successive elimination of variables. It means that in principle, one can apply the rules to solve at least some nonsymmetric systems. However, there is a problem which is specific to nonsymmetric case. Namely, it is possible to have  $A_{ij} = 0$ and  $A_{ji} \neq 0$ . In this case, rules (19) lead to singularity as  $A_{ij}$  appears in the denominator. Since parametrization of messages is not unique both from the elimination and probabilistic perspectives, it is possible to define new set of messages  $\tilde{\Lambda}$  and  $\mathbf{m}$  as follows

$$m_{ji}^{(n)} \equiv \mu_{ji}^{(n)} \Lambda_{ji}^{(n)}, \ \tilde{\Lambda}_{ji}^{(n)} \equiv \Lambda_{ji}^{(n)} / A_{ji}.$$
 (20)

#### Algorithm 1 GaBP for a nonsymmetric linear system.

Form directed graph  $G = \{\mathcal{V}, \mathcal{E}\}$  based on **A**. while error > tolerance **do** for  $j \in \mathcal{V}$  **do**   $m = b_j + \sum_{k \in N(j)} m_{kj}$   $\Sigma = A_{jj} + \sum_{k \in N(j)} \tilde{\Lambda}_{kj} A_{kj}$   $x_j \leftarrow m/\Sigma$ for  $(j, i) \in \mathcal{E}$  **do**   $\tilde{\Lambda}_{ji} \leftarrow -A_{ij}/(\Sigma - \tilde{\Lambda}_{ij} A_{ij})$   $\tilde{\mu}_{ji} \leftarrow \tilde{\Lambda}_{ji} (m - m_{ij})$ end for error =  $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_{\infty}$ end while

Then update rules become

$$m_{ji}^{(n+1)} = \tilde{\Lambda}_{ji}^{(n+1)} \left( b_j + \sum_{k \in N(j) \setminus i} m_{kj}^{(n)} \right), \ \tilde{\Lambda}_{ji}^{(n+1)} = -\frac{A_{ij}}{A_{jj} + \sum_{k \in N(j) \setminus i} \tilde{\Lambda}_{kj}^{(n)} A_{kj}};$$
$$\mu_i^{(n)} = \frac{b_i + \sum_{j \in N(i)} m_{ji}^{(n)}}{A_{ii} + \sum_{j \in N(i)} \tilde{\Lambda}_{ji}^{(n)} A_{ji}}, \ \beta_i^{(n)} = A_{ii} + \sum_{j \in N(i)} \tilde{\Lambda}_{ji}^{(n)} A_{ji}.$$
(21)

Note that reparametrization (20) has a problem in that it is not one-to-one if  $A_{ji} = 0$ . However, the quick look at the equations (6), (7) makes clear that indeed it is possible to define messages in that way. That is, if  $A_{jk} = 0$ , one does not need to eliminate  $x_k$  from the second equation so the message  $\Lambda_{kj}$  is indeed zero.

For a given matrix  $\mathbf{A} \in \mathbb{R}^{N \times N}$ , we construct a graph with N vertices  $v \in \mathcal{V}$  corresponding to the variables  $x_1, \ldots, x_N$  and the set of directed edges  $\mathcal{E}$ . The edge pointing from the vertex j to the vertex i belongs to the set of edges iff  $A_{ij} \neq 0$ , i.e.  $A_{ij} \neq 0 \Leftrightarrow e_{ji} \in \mathcal{E}$ . This definition fixes the correspondence between directed graphs and nonsymmetric matrices and allows us to use GaBP (see Algorithm 1) beyond its usual domain of applicability.

Algorithm 1 is sequential, but can run in parallel after some modifications. The stopping criteria can be different, for example, it is possible to use different norms, or error =  $\|\mathbf{x}^{(n+1)} - \mathbf{x}^{(n)}\|_{\infty}$ , or

error = max 
$$\left( \left\| \widetilde{\boldsymbol{\mu}}^{(n+1)} - \widetilde{\boldsymbol{\mu}}^{(n)} \right\|_{\infty}, \left\| \widetilde{\boldsymbol{\Lambda}}^{(n+1)} - \widetilde{\boldsymbol{\Lambda}}^{(n)} \right\|_{\infty} \right).$$
 (22)

Note that the update of  $\widehat{\Lambda}$  decouples from the one for  $\widetilde{\mu}$ . So it is possible to construct an algorithm that computes only messages  $\widehat{\Lambda}$  and returns diagonal elements for the inverse matrix. Later, these messages can be used in the course of all successive iterations if one resorts to the error correction scheme. We discuss how the algorithm of this kind can be utilized to decrease the number of floating point operations in the context of a multigrid scheme.

One of the central results of the present work is that two classical theorems from GaBP theory, summarized below, can be readily established for nonsymmetric matrices.

**Theorem 2.1.** If there is  $N \in \mathbb{N}$  such that  $\widetilde{\mu}_e^{(N+k)} = \widetilde{\mu}_e^{(N)}$ ,  $\widetilde{\Lambda}_e^{(N+k)} = \widetilde{\Lambda}_e^{(N)}$  for all  $e \in \mathcal{E}$  and for any  $k \in \mathbb{N}$ , then  $\mu_i^{(N+k)} = \mu_i^{(N)} = (\mathbf{A}^{-1}\mathbf{b})_i$ .

The analogous result for symmetric matrices first appeared in [33]. In A, we show how to extend the proof for the nonsymmetric case.

**Theorem 2.2.** If  $A_{ii} \neq 0 \ \forall i, \ |\widetilde{R}|_{ij} = (1 - \delta_{ij}) \frac{|A_{ij}|}{|A_{ii}|}$ , and  $\rho(|\widetilde{\mathbf{R}}|) < 1$ , then the Algorithm 1 converges to the solution  $\mathbf{x}^* = \mathbf{A}^{-1}\mathbf{b}$  for any  $\mathbf{b}$ .

Sufficient condition for symmetric positive-definite matrices was established in [17]. Section B contains the proof with necessary modifications that holds for nonsymmetric matrices.

To make connections with the classical theory of iterative methods, we give another (less general) sufficient condition.

**Corollary 2.0.1.** If **A** is the *M*-matrix (see [24, Definition 1.30, Theorem 1.31]), Algorithm 1 converges to the solution  $\mathbf{x}^* = \mathbf{A}^{-1}\mathbf{b}$  for any **b**.

*Proof.* For *M*-matrix  $\rho(\mathbf{I} - \mathbf{D}^{-1}\mathbf{A}) < 1$ ,  $A_{ij} \leq 0$ ,  $i \neq j$  and  $A_{ii} > 0$ , where **D** is a diagonal of **A**. It means that  $\widetilde{\mathbf{R}} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A} = \left|\widetilde{\mathbf{R}}\right|$  and  $\rho\left(\left|\widetilde{\mathbf{R}}\right|\right) < 1$ .  $\Box$ 

## 3 Generalized GaBP solvers

#### 3.1 Parent-to-child schedule

We present a particular version of the parent-to-child schedule from [39] applied to the pairwise Markov graphical models.

First, we define a region r as a connected subgraph of the original graph  $\Gamma$ . Each vertex  $a_1$  in the region graph is a region that may be connected by a directed edge with another vertex  $a_2$  if  $a_2 \subset a_1$ . The direction of the edge is from a larger region to smaller. If there is a directed edge from a to b, then a is a parent of b, and b is a child of a. If the vertices a and b are connected by a directed path starting from a, then a is an ancestor of b, and b is a descendent of a. The set of all vertices of the factor graph is  $\mathcal{R}$ , the set of all edges is  $\mathcal{E}_{\mathcal{R}}$ , the set of all parents, children, ancestors, descendants of a are P(a), C(a),



Figure 2: Pairwise Markov model and valid region graph with counting numbers. Shaded nodes belong to the shadow of 5689, and nodes with thick borders to the Markov blanket of 5689. See Section 3.1 for details.

A(a), and D(a), respectively. We supplement each region  $r \in \mathcal{R}$  with a counting number,

$$c_r = 1 - \sum_{i \in A(r)} c_i, \tag{23}$$

and require that each vertex  $v \in \mathcal{V}$  and each edge  $e \in \mathcal{E}$  of the original graph be counted exactly once,

$$\sum_{\in \mathcal{R}, v \in r} c_r = \sum_{r \in \mathcal{R}, e \in r} c_r = 1.$$
(24)

The definitions of a counting number (23) and condition (24) are justified in the framework of the cluster variation method [21] (or the Kikuchi approximation [16]). Equation (24) is a result of the Möbius inversion formula applied to the sum over partially ordered sets [3], and (24) can be proven using definitions of Möbius and Zeta functions [3, equation 16].

A sample region graph is shown in Figure 2. For example, by region 5689, we mean all nodes and all links between them that are present on the original graph. It is easy to see that the counting number condition (24) is satisfied for all the links and nodes.

The last two definitions that we need are the shadow of the region  $S(r) = D(r) \cup r$  and a Markov blanket of the region  $B(r) = P(S(r)) \setminus S(r)$ . An example of both the shadow and Markov blanket of region 5689 can be found in Figure 2b.

A parent-to-child algorithm consists of three elements: (i) messages that propagate along the directed edges of the region graph

$$\mathbf{m}_{a \to b}(\mathbf{x}_b), \tag{25}$$

where  $x_b$  corresponds to variables belong to the cluster b, (ii) a formula for the cluster beliefs

$$\mathbf{b}_{r}(\mathbf{x}_{r}) \sim \prod_{i \in \mathcal{V}_{r}} \phi_{i}(x_{i}) \prod_{(ij) \in \mathcal{E}_{r}} \psi_{ij}(x_{i}, x_{j}) \prod_{a \in B(r), b \in S(r)} \mathbf{m}_{a \to b}(\mathbf{x}_{b}), \qquad (26)$$



Figure 3: An example of a region graph. (a) - the original graph partitioning, (b) - the region graph, and (c) - the hypergraph structure. In Section 5, this partitioning is referred to as "line GaBP".

and (iii) message update rules that follow from the consistency conditions

$$\forall l, r \in \mathcal{R}, \ l \subset r \Rightarrow \sum_{x_r \setminus x_l} \mathbf{b}_r(\mathbf{x}_r) = \mathbf{b}_l(\mathbf{x}_l).$$
(27)

In [39, equation 114], one can find message update rules for the general situation, but for our simple region graph, conditions (27) suffice.

#### 3.2 Two-layer generalized GaBP

In this section, we consider the simplest possible valid region graph that consists of two layers, Figure 3b. The large regions, the horizontal and vertical stripes in 3a, are parents of small regions presented by the individual nodes. To proceed, we need to establish some further notation. First, we define a projector on the region k,

$$(\Pi_k)_{ij} = \begin{cases} \delta_{ij}, \ i, j \in k, \\ 0, \text{ otherwise.} \end{cases}$$
(28)

Here, |j| = |k| and |i| > |k| is chosen to be conformable depending on the context. Ordering is global, i.e., it is fixed for the whole graph and maintained the same way in all manipulations. We also introduce brackets,

$$\mathbf{\Pi}_{k}^{T}\mathbf{A}\mathbf{\Pi}_{k} = \left[\mathbf{A}\right]_{k}, \mathbf{\Pi}_{k}^{T}\mathbf{b} = \left[\mathbf{b}\right]_{k};$$
(29a)

$$\mathbf{\Pi}_{k}\mathbf{C}\mathbf{\Pi}_{k}^{T} = \left]\mathbf{C}\right]_{k}, \mathbf{\Pi}_{k}\mathbf{b} = \left]\mathbf{b}\right]_{k}, \qquad (29b)$$

where  $\mathbf{C} \in \mathbb{R}^{|k| \times |k|}$  and according to our notation, the sizes of the matrix and the vector (29b) depend of the context whereas in (29a) the size of the matrix is  $|k| \times |k|$  and the size of the vector is |k|.

Let  $\{L_i\}$  and  $\{l_i\}$  be sets of large and small regions in the two-layer region graph. For messages, we use the following parameterizations

$$m_{ab}(\mathbf{x}_b) = \mathcal{N}\left(\mathbf{x}_b \left| \boldsymbol{\mu}_{ab}, \boldsymbol{\Lambda^{-1}}_{ab} \right.\right) = \mathcal{N}\left(\mathbf{x}_b \left| \boldsymbol{\mu}_{ab}, \boldsymbol{\Sigma}_{ab} \right.\right)$$
(30)

According to (26), the belief of any region reads

$$\mathbf{b}_{L}(\mathbf{x}_{L}) = \mathcal{N}\left(\mathbf{x}_{L} | \boldsymbol{\Sigma}_{L} \mathbf{m}_{L}, \boldsymbol{\Sigma}_{L} \right), \qquad (31a)$$

$$\mathbf{m}_{L} = [\mathbf{b}]_{L} + \sum_{\substack{a \in \mathcal{B}(L) \\ b \in \mathcal{S}(L)}} ]\mathbf{\Lambda}_{ab} \boldsymbol{\mu}_{ab}[_{b}, \qquad (31b)$$

$$\boldsymbol{\Sigma}_{L} = \left( [\mathbf{A}]_{L} + \sum_{\substack{a \in \mathcal{B}(L) \\ b \in \mathcal{S}(L)}} ]\boldsymbol{\Lambda}_{ab}[_{b} \right)^{-1}.$$
 (31c)

For the small region l, the shadow is S(l) = l and the Markov blanket is  $\mathcal{B}(l) = P(l)$ , whereas for the large region L, the shadow is S(L) = C(L) and the Markov blanket is  $\mathcal{B}(L) = P(C(L)) \setminus L$ . Consistency condition (27) allows one to derive update rules for each message sent from the parent region to the child region,

$$\mathbf{m}_{Ll}(x_l) = \int \frac{\mathbf{d}\mathbf{x}_L}{\mathbf{d}\mathbf{x}_l} \mathcal{N}\left(\mathbf{x}_l \left| \widetilde{\boldsymbol{\Sigma}} \widetilde{\boldsymbol{b}}, \widetilde{\boldsymbol{\Sigma}} \right. \right),$$
(32a)

$$\widetilde{\mathbf{b}} = [\mathbf{b}]_{L} - ][\mathbf{b}]_{l}[_{l} + \sum_{\substack{a \in \mathcal{B}(L) \\ b \in \mathcal{S}(L) \setminus l}} ]\mathbf{\Lambda}_{ab} \boldsymbol{\mu}_{ab}[_{b}, \qquad (32b)$$

$$\widetilde{\boldsymbol{\Sigma}} = \left( [\mathbf{A}]_L - ][\mathbf{A}]_l [_l + \sum_{\substack{a \in \mathcal{B}(L) \\ b \in \mathcal{S}(L) \setminus l}} ]\boldsymbol{\Lambda}_{ab} [_b \right)^{-1}.$$
(32c)

Using standard results for marginals of the normal distribution, we can deduce

$$\boldsymbol{\Lambda}_{Ll} = \left( \left[ \widetilde{\boldsymbol{\Sigma}} \right]_l \right)^{-1}, \ \boldsymbol{\mu}_{Ll} = \left[ \widetilde{\boldsymbol{\Sigma}} \widetilde{\boldsymbol{b}} \right]_l.$$
(33)

If region L has many children and  $|l| \ll |L|$ , the direct use of equation (32c) is not efficient. Instead, we use the formula following from the Woodbury matrix identity,

$$\left[ \left( \mathbf{A} + \left] \left[ \mathbf{B} \right]_{l} \right]_{l} \right]_{l} = \left( \left( \left[ \mathbf{A}^{-1} \right]_{l} \right)^{-1} + \left[ \mathbf{B} \right]_{l} \right)^{-1}, \tag{34a}$$

$$\left[ \left( \mathbf{A} + \left] \left[ \mathbf{B} \right]_{l} \right]_{l} \right]^{-1} \left( \mathbf{b} + \left] \left[ \mathbf{c} \right]_{l} \right]_{l} =$$

$$(34b)$$

$$= \left( \left( \left[ \mathbf{A}^{-1} \right]_l \right)^{-1} + \left[ \mathbf{B} \right]_l \right)^{-1} \left( \left( \left[ \mathbf{A}^{-1} \right]_l \right)^{-1} \left[ \mathbf{A}^{-1} \mathbf{b} \right]_l + \left[ \mathbf{c} \right]_l \right),$$
(61)

and split for each child region l,

$$\widetilde{\mathbf{b}} = \left( [\mathbf{b}]_L + \sum_{\substack{a \in \mathcal{B}(L) \\ b \in \mathcal{S}(L)}} ]\mathbf{\Lambda}_{ab} \boldsymbol{\mu}_{ab} [_b \right) - \left( ] [\mathbf{b}]_l [_l + \sum_{a \in P(l) \setminus L} ]\mathbf{\Lambda}_{al} \boldsymbol{\mu}_{al} [_l \right) = \widetilde{\mathbf{b}}_0 - \widetilde{\mathbf{b}}_l,$$
(35a)

$$\widetilde{\boldsymbol{\Sigma}}^{-1} = \left( [\mathbf{A}]_L + \sum_{\substack{a \in \mathcal{B}(L) \\ b \in \mathcal{S}(L)}} ]\boldsymbol{\Lambda}_{ab}[_b \right) - \left( ][\mathbf{A}]_l[_l + \sum_{a \in P(l) \setminus L} ]\boldsymbol{\Lambda}_{al}[_l \right) = \widetilde{\boldsymbol{\Lambda}}_0 - \widetilde{\boldsymbol{\Lambda}}_l.$$
(35b)

Then, for precision and mean of messages, we obtain

$$\mathbf{\Lambda}_{Ll} = \left( \left[ \widetilde{\mathbf{\Lambda}}_0^{-1} \right]_l \right)^{-1} - \widetilde{\mathbf{\Lambda}}_l, \tag{36a}$$

$$\boldsymbol{\mu}_{Ll} = \boldsymbol{\Lambda}_{Ll}^{-1} \left( \left( \left[ \widetilde{\boldsymbol{\Lambda}}_0^{-1} \right]_l \right)^{-1} \left[ \widetilde{\boldsymbol{\Lambda}}_0^{-1} \widetilde{\mathbf{b}}_0 \right]_l - \widetilde{\mathbf{b}}_l \right).$$
(36b)

Equations (36) are especially useful in the situation when the small regions consist of the single node, i.e. the case of graph in Figure 3. In this situation, one needs to invert the matrix corresponding to the large region only once whereas the direct application of (33) leads to |C(a)| inversions.

From formulae (36), one can derive GaBP rules. As the large regions in the Bethe approximation consist of two vertices with the edge connecting them, we can rewrite messages as

$$\Lambda_{(ij)i} \equiv \Lambda_{ji}, \ \mu_{(ij)i} \equiv \mu_{ji}. \tag{37}$$

Then, the Gaussian belief propagation rules follow from

$$\widetilde{\mathbf{\Lambda}}_{0} = \begin{pmatrix} A_{ii} + \sum_{k \in N(i) \setminus j} \Lambda_{ki} & A_{ij} \\ A_{ji} & A_{jj} + \sum_{k \in N(j) \setminus i} \Lambda_{kj} \end{pmatrix}, \quad (38a)$$

$$\widetilde{\Lambda}_i = A_{ii} + \sum_{k \in N(i) \setminus j} \Lambda_{ki}, \tag{38b}$$

$$\left]\widetilde{\Lambda}_{i}\right[_{i} = \begin{pmatrix} A_{ii} + \sum_{k \in N(i) \setminus j} \Lambda_{ki} & 0\\ 0 & 0 \end{pmatrix}.$$
(38c)

The validity of the presented rules for nonsymmetric linear problems does not follow from the derivation above. Nevertheless, one can apply the generalized GaBP rules with no modifications to solve them, as we explain in the next sections.

#### 3.3 Elimination perspective

First, we define a hypergraph  $\mathcal{G}$  based on the region graph. The set of nodes coincides with the set of large regions, and each common child corresponds to the edge in the graph. The example of such a hypergraph is shown in Figure 3c. With this definition, messages from equations (32) can be redefined with no reference to the small region as long as a one-to-one correspondence between edges of  $\mathcal{G}$  and child regions of the original region graph are established. Now we study the single message from region j to region i with  $k = j \cap i$  and  $\overline{j} = j \setminus k$ . According to (32) and (33), the precision part of the message is

$$\mathbf{\Lambda}_{ji} = \left( \left( \begin{pmatrix} \mathbf{A}_{\overline{jj}} & \mathbf{A}_{\overline{j}k} \\ \mathbf{A}_{k\overline{j}} & 0 \end{pmatrix}^{-1} \right)_{kk} \right)^{-1} = -\mathbf{A}_{k\overline{j}} \mathbf{A}_{\overline{jj}}^{-1} \mathbf{A}_{\overline{j}k}, \tag{39}$$

and the mean is

$$\boldsymbol{\mu}_{ji} = \left( \begin{pmatrix} \mathbf{A}_{\overline{jj}} & \mathbf{A}_{\overline{j}k} \\ \mathbf{A}_{k\overline{j}} & 0 \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{b}_{\overline{j}} \\ 0 \end{pmatrix} \right)_{k} = -\mathbf{A}_{ji}^{-1} \mathbf{A}_{k\overline{j}} \mathbf{A}_{\overline{jj}}^{-1} \mathbf{b}_{\overline{j}}.$$
(40)

So the subset k of  $\mathbf{A}_i$  and  $\mathbf{b}_i$  receive a correction from region j,

$$\Delta \left( \mathbf{A}_{i} \right)_{kk} = -\mathbf{A}_{k\overline{j}} \mathbf{A}_{\overline{j}\overline{j}}^{-1} \mathbf{A}_{\overline{j}\overline{k}}, \tag{41a}$$

$$\Delta \left( \mathbf{b}_{i} \right)_{k} = -\mathbf{A}_{k\overline{j}} \mathbf{A}_{\overline{j}\overline{j}}^{-1} \mathbf{b}_{\overline{j}}.$$
 (41b)

The corrections above are the same as in the ordinary block LU decomposition,

$$\begin{pmatrix} \mathbf{A}_{\overline{j}\overline{j}} & \mathbf{A}_{\overline{j}k} \\ \mathbf{A}_{k\overline{j}} & \mathbf{A}_{kk} \end{pmatrix} = \begin{pmatrix} \mathbf{I} & 0 \\ \mathbf{A}_{k\overline{j}} \mathbf{A}_{\overline{j}\overline{j}}^{-1} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{A}_{\overline{j}\overline{j}} & \mathbf{A}_{\overline{j}\overline{k}} \\ 0 & \mathbf{A}_{kk} - \mathbf{A}_{k\overline{j}} \mathbf{A}_{\overline{j}\overline{j}}^{-1} \mathbf{A}_{\overline{j}\overline{k}} \end{pmatrix}$$
(42)

Thus, we have a correspondence between the block LU operations and the generalized GaBP for the two-layer region graph. As in the case of the regular GaBP, LU applies in the fashion of dynamic programming, i.e., one does not just solve a smaller subproblem as in the block iterative scheme, but rather forms a recursive procedure that decouples different blocks from each other. Again, one should reparametrize messages for them to be valid for the arbitrary invertible matrix **A**. Note that if  $\mathbf{A}_{jk} = 0$ , then  $\mathbf{A}_{ji}$  is not invertible. However, this is not a problem because in this case  $\mathbf{A}_i$  does not receive corrections  $\Delta(\mathbf{A}_i)_{kk}$ , and the reparametrization for the mean messages  $\mathbf{m}_{Ll} \equiv \mathbf{A}_{Ll} \boldsymbol{\mu}_{Ll}$  resolves the issue.

#### 3.4 The algorithm

In the case of generalized GaBP, messages propagate on the region graph. To remind,  $\{L_i\}$  is the set of large regions, and  $\{l_i\}$  is the set of small regions. The algorithm, as we describe it in this section, acts on a given region graph. We do not provide an algorithm or recommendations on how to build a region graph. Some observations about the influence of a particular choice of the large regions

Algorithm 2 Generalized two-layer GaBP for a nonsymmetric linear system.

For a given two-layer region graph  $G = \{\mathcal{R}, \mathcal{E}_{\mathcal{R}}\}.$ while error > tolerance do for  $L \in \{L_i\} \subset \mathcal{R}$  do  $\widetilde{\mathbf{b}}_0 = [\mathbf{b}]_L$   $\widetilde{\mathbf{A}}_0 = [\mathbf{A}]_L$ for  $l \in C(L)$  do  $\widetilde{\mathbf{b}}_0 \leftarrow \widetilde{\mathbf{b}}_0 + \sum_{\substack{L' \in P(C(l)) \setminus L}} ]\mathbf{M}_{L'l}[_l$ end for  $[\mathbf{x}]_L \leftarrow \widetilde{\mathbf{A}}_0^{-1} \widetilde{\mathbf{b}}_0$ for  $l \in C(L)$  do  $\mathbf{m}_{Ll} \leftarrow \left(\left[\widetilde{\mathbf{A}}_0^{-1}\right]_l\right)^{-1} [\mathbf{x}]_l - ][\mathbf{b}]_l[_l - \sum_{\substack{L' \in P(C(l)) \setminus L}} \mathbf{m}_{L'l}$   $\mathbf{A}_{Ll} \leftarrow \left(\left[\widetilde{\mathbf{A}}_0^{-1}\right]_l\right)^{-1} - ][\mathbf{A}]_l[_l - \sum_{\substack{L' \in P(C(l)) \setminus L}} \mathbf{A}_{L'l}$ end for end for error =  $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_{\infty}$ end while

can be found in Section 5. Regarding complexity, for each region, one needs to solve a linear system of |L| equations, and also find a submatrix of the inverse matrix of size  $|L| \times |L|$ . We do not specify how to do it. But generally, the former task is hard to accomplish asymptotically faster than the whole inverse, so the computational cost of the entire scheme depends mostly on this operation. For this reason, it what follows we mostly consider region graphs with small regions each of which contains a single node. In this situation, one can avoid fill-in and estimate only the diagonal of the inverse matrix which is usually more straightforward. Also, there is no need for an additional inverse step during the message update. Moreover, when all small regions are single nodes, the whole algorithm is either a method that speeds up GaBP or a particular schedule of GaBP depending on the chosen way of finding the inverse and the solution of a linear system.

To perform a worst-case analysis both of Algorithm 2 and update rules (36), for a given region with N variables, we denote the number of children by M, the number of variables of each child region by  $n_i$ , the number of parents for each child by  $p_i$ , and use the LU decomposition to find an inverse. For the Algorithm 2, the number of operations is

$$\#_1 = \sum_{i=1}^{M} \left[ \frac{3}{2} n_i^3 + 2n_i \left( n_i + 1 \right) \left( p_i - 1 \right) \right] + \frac{3}{2} N^3.$$
(43)

The first term in brackets is due to the inverse during the message update stage, the second term in brackets is due to the message update and message accumulation steps, the last term is from the inverse of a matrix for the large region. For update rules (36), we obtain

$$\#_2 = \sum_{i=1}^{M} \left[ \frac{3}{2} n_i^3 + (M-1)n_i \left( n_i + 1 \right) \left( p_i - 1 \right) \right] + M \frac{3}{2} N^3.$$
(44)

Since

$$\#_2 - \#_1 = (M-3) \sum_{i=1}^M n_i (n_i + 1) (p_i - 1) + (M-1) \frac{3}{2} N^3, \qquad (45)$$

one obtains a speed-up if M > 3 for an arbitrary region graph. For certain regular partitions, for example the one in Figure 3a, M scales like N, and in this cases, the Algorithm 2 performs  $O(N^3)$  operations whereas rules (36) perform  $O(N^4)$  operations for each large region.

It is easy to see that if the LU method is employed, the number of operations for the single sweep is O(K), where K is an overall number of variables, only if the number of variables, for some regular partition for which the limit makes sense, in each large region scales like O(1). Clearly, LU is not the best option for all cases, for example, matrices can have a particular structure (tridiagonal as Figure 3a), or it may be more advantageous to use probing or other techniques of estimation of certain subblocks of the inverse matrix [30] in combination with some iterative scheme for the solution of linear system.

The Algorithm 2 can be justified theoretically on the basis of the following two theorems.

**Theorem 3.1.** If there is  $N \in \mathbb{N}$  such that  $\mathbf{m}_e^{(N+k)} = \mathbf{m}_e^{(N)}$ ,  $\mathbf{\Lambda}_e^{(N+k)} = \mathbf{\Lambda}_e^{(N)}$ for all  $e \in \mathcal{E}_{\mathcal{R}}$  and for any  $k \in \mathbb{N}$ , then for each large region  $[\mathbf{x}]_L \equiv \widetilde{\mathbf{\Lambda}}_0^{-1} \widetilde{\mathbf{b}}_0 = [\mathbf{A}^{-1}\mathbf{b}]_L$  (see Algorithm 2 for details).

That is, the steady state of the message flow, if it exists, corresponds to the exact solution. The proof is given in C.

For the sufficient condition for convergence we need additional definitions. First, based on a given region graph  $\{\mathcal{R}, \mathcal{E}_{\mathcal{R}}\}$ , we define a set of variable subsets

$$F \equiv \left(\bigcup_{i} \{l_i\}\right) \cup \left(\bigcup_{j} \left\{L_j \setminus \bigcup_{p \in C(L_j)} \{l_p\}\right\}\right),\tag{46}$$

where  $l_i$  is a small region and  $L_j$  is a large region. Using F, we form a partition of the matrix **A** and the right-hand-side vector **b** 

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{ii} & \mathbf{A}_{ij} & \dots \\ \mathbf{A}_{ji} & \mathbf{A}_{jj} & \dots \\ \vdots & \vdots & \ddots \end{pmatrix}, \mathbf{b} = \begin{pmatrix} \mathbf{b}_i \\ \mathbf{b}_j \\ \vdots \end{pmatrix},$$
(47)

where each diagonal block corresponds to the element of the set F. We also define

$$\widetilde{\mathbf{A}}_{ij} = \mathbf{A}_{ii}^{-1} \mathbf{A}_{ij} \equiv \mathbf{I}_{ij} - \widetilde{\mathbf{R}}_{ij}, \\ \left\| \widetilde{\mathbf{R}} \right\|_{ij} \equiv \left\| \widetilde{\mathbf{R}}_{ij} \right\|, \ \widetilde{\mathbf{b}}_{i} = \mathbf{A}_{ii}^{-1} \mathbf{b}_{i}.$$

$$(48)$$

Note that the second line in the preceding equation contains a definition of matrix  $\|\widetilde{\mathbf{R}}\|$ , which depends on the operator norm  $\|\cdot\|$  (see [15, ch. 5, Definition 5.6.3]).

The following statement gives sufficient conditions for convergence.

**Theorem 3.2.** If for matrix (47) which is based on partition (46) det  $\mathbf{A}_{ii} \neq 0 \forall i$ and  $\rho\left(\left\|\widetilde{\mathbf{R}}\right\|\right) < 1$  in some operator norm, then two-layer generalized GaBP (algorithm 2) converges to the exact solution  $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ .

The proof of this theorems appears in D. From the second part of the argument in D.2, one can deduce the following

**Corollary 3.0.1.** Generalized GaBP (algorithm 2) converges whenever GaBP converges (Theorem 2.2) and all submatrices corresponding to the large blocks are invertible (see equations (46), (47)).

The opposite does not hold. For example, consider a matrix

$$\mathbf{A} = \begin{pmatrix} 10 & 1.5 & 2 & 2 & 0 & 2 & 0 \\ 2 & 4 & 2.5 & 0 & 2 & 0 & 0 \\ 2 & 3 & 5 & 0 & 0 & 0 & 1 \\ 2 & 0 & 0 & 10 & 0.5 & 1 & 0 \\ 0 & 2 & 0 & 0.5 & 5 & 0 & 1 \\ 2 & 0 & 0 & 1 & 0 & 7 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 2 \end{pmatrix} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{A}_{13} \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \mathbf{A}_{23} \\ \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{A}_{33} \end{pmatrix}, \quad (49)$$
$$\mathbf{A}_{11} \in \mathbb{R}^{3 \times 3}, \ \mathbf{A}_{22} \in \mathbb{R}^{2 \times 2}, \ \mathbf{A}_{33} \in \mathbb{R}^{2 \times 2}.$$

In this case, the spectral radius of matrix  $|\widetilde{\mathbf{R}}|$  defined in Theorem 2.2 equals ~ 1.03 and GaBP diverges<sup>2</sup>. On the other hand, the spectral radius of  $|\widetilde{\mathbf{R}}|$  defined by (48) and the partition given in (49) is smaller than one in  $l_{\infty}$  and spectral norms [15, Examples 5.6.5, 5.6.6] (see [1] for further details).

## 4 GaBP as a smoother for the multigrid method

The most straightforward view on the geometric multigrid is to describe it as an acceleration scheme for classical iterative methods. For completeness, we briefly recall the main ideas.

<sup>&</sup>lt;sup>2</sup>Note that the divergence of GaBP does not follow from  $|\widetilde{\mathbf{R}}| > 1$  as Theorem 2.2 provides only sufficient conditions. For this particular case, the pathological behavior of GaBP follows from the numerical experiment (see [1] for details).

Algorithm 3 GaBP as a smoother.

Compute a residual $\mathbf{r}^n = \mathbf{b} - \mathbf{A}\mathbf{x}^n$ .
Apply $\mu$ sweeps of Algorithm 1 or 2 to the linear system $\mathbf{Ae} = \mathbf{r}^n$ .
Perform an error correction $\mathbf{x}^{n+1} = \mathbf{x}^n + \mathbf{e}^{\mu}$ .

The multigrid consists of four essential elements: a projection operator  $\mathbf{I}_{V}^{V'}: V \to V'$  (V, V' are linear spaces) that reduces the number of degrees of freedom, an interpolation operator  $\mathbf{I}_{V'}^{V}: V' \to V$  that acts in the "inverse" way, a smoothing operator  $\mathbf{S}_{V}: V \to V$  which is usually a classical relaxation method, and a set of linear operators  $\mathbf{A}_{V'}$  that approximate  $\mathbf{A}$  on coarse spaces V'. What we describe next is a two-grid cycle.

- For the current approximation  $\mathbf{x}^n$  of solutions of  $\mathbf{A}\mathbf{x} = \mathbf{b}$ , one performs several relaxation steps  $\overline{\mathbf{x}} = \mathbf{S}^{\nu} \mathbf{x}^n$ .
- Then, based on properties of **S**, the linear space V' and the projection operator  $\mathbf{I}_{V}^{V'}: V \to V'$  are constructed. The purpose of this space is to represent the residual  $\mathbf{r} = \mathbf{b} \mathbf{A}\overline{\mathbf{x}}$  and an error  $\mathbf{e} = \mathbf{x}_{\text{exact}} \overline{\mathbf{x}}$  accurately using fewer degrees of freedom |V'| < |V|.
- Having the space V', one constructs an operator A' that approximates A and solves the error equation A'e' = I<sub>V</sub><sup>V'</sup>r.
- The error, after projection back to V, gives the next approximation to the exact solution,  $\mathbf{x}^{n+1} = \mathbf{S}^{\mu} \left( \mathbf{x}^n + \mathbf{I}_{V'}^V \mathbf{e}' \right)$ .

The multigrid utilizes a two-grid cycle to solve the error equation  $\mathbf{A}' \mathbf{e}' = \mathbf{I}_V^{V'} \mathbf{r}$ itself. It produces the chain of spaces (grids in the geometric setup), projection operators that allow moving between them, and a set of approximate linear operators. For more details, we refer the reader to other resources: a simple introduction to geometric multigrid can be found in [25, ch. 13], for the algebraic multigrid a recent review [35], physical considerations about algebraic multigrid can be found in the introduction of [23], and among other books on the subject, [31] provides a comprehensive introduction for practitioners.

Here, we consider only linear systems of equations arising from finite difference discretization of elliptic equations with smooth coefficients in two space dimensions. In this case it is possible to use grids in place of linear spaces. Let the finest grid contain  $2^J + 1$  points  $\mathcal{G}_J = \left\{ (i+j)h | h = 2^{-J}, i, j = \overline{0, 2^J} \right\}$ , then the coarser grid  $\mathcal{G}_{J-1}$  contains each second points along both directions. As we are working in the physical space, the restriction operator  $\mathbf{I}_{V}^{V'}$  computes a weighted average of neighbouring points, operator  $\mathbf{I}_{V'}^{V}$  performs interpolation, and  $\mathbf{A}$  on the grid  $\mathcal{G}_{J'}$  is a finite difference approximation of the differential operator. In this article we always use full weighting restriction [31, eq. 2.3.3] and bilinear interpolation [31, eq. 2.3.7]. The smoother should be a mapping  $\mathbf{S} : \mathbf{x}^n \to \mathbf{x}^{n+1}$ . Although GaBP is not of this form, one can use an error correction scheme as explained in Algorithm 3. In the next two subsections we analyze smoothing properties of Algorithm 3 and estimate its computational complexity.

#### 4.1 Local Fourier Analysis

Local Fourier Analysis allows us to compute the spectral radius of the two-grid cycle, the smoothing factor of the relaxation scheme, and the error contraction in a chosen norm [31, ch. 4]. In this subsection, we apply the analysis to the central difference discretization of the Laplace equation in two spatial dimensions

$$\frac{1}{h^2} \begin{bmatrix} -1 & \\ -1 & 4 & -1 \\ & -1 \end{bmatrix} u_{ij} = f_{ij}.$$
(50)

If after  $\mu$  sweeps of Algorithm 1 the solution has a form  $\mathbf{Sr}^n$ , then the output of Algorithm 3 is  $\mathbf{x}^{n+1} = \mathbf{S} (\mathbf{b} - \mathbf{Ax}^n) + \mathbf{x}^n$ . Thus, for the error we have

$$\mathbf{e}^{n+1} = (\mathbf{I} - \mathbf{S}\mathbf{A})\,\mathbf{e}^n. \tag{51}$$

Now we need to find a stencil of the operator  $\mathbf{S}$ . For GaBP it differs for parallel and sequential versions. For one and two sweeps of parallel version on the infinite lattice we have

$$\mathbf{S}_{\text{parallel}}^{1}u_{ij} = \frac{h^{2}}{4}u_{ij}, \mathbf{S}_{\text{parallel}}^{2}u_{ij} = \frac{h^{2}}{12} \begin{bmatrix} 1 & 1 \\ 1 & 4 & 1 \\ 1 & 1 \end{bmatrix} u_{ij}.$$
 (52)

It is easy to compute symbols

$$\mathbf{S}_{\text{parallel}}^{1}(\boldsymbol{\theta}) = \frac{h^{2}}{4}, \mathbf{S}_{\text{parallel}}^{2}(\boldsymbol{\theta}) = \frac{h^{2}}{3} \left( 1 + \cos\left(\frac{\theta_{1} + \theta_{2}}{2}\right) \cos\left(\frac{\theta_{1} - \theta_{2}}{2}\right) \right),$$
  
$$\mathbf{A}(\boldsymbol{\theta}) = \frac{4}{h^{2}} \left( 1 - \cos\left(\frac{\theta_{1} + \theta_{2}}{2}\right) \cos\left(\frac{\theta_{1} - \theta_{2}}{2}\right) \right).$$
 (53)

As the smoothing factor is

$$\mu = \max_{\boldsymbol{\theta} \in \text{high}} \left| 1 - \mathbf{S}\left(\boldsymbol{\theta}\right) \mathbf{A}\left(\boldsymbol{\theta}\right) \right|, \text{ high} = \left[ -\pi, \pi \right]^2 \setminus \left[ -\frac{\pi}{2}, \frac{\pi}{2} \right]^2, \tag{54}$$

we can conclude that the parallel version of GaBP shows no smoothing properties. This conclusion is confirmed by our numerical experiments.

In the sequential case, one can deduce the form of  $\mathbf{S}$  based on the elimination perspective. When one starts to move along the lattice, messages correspond to the elimination of variables, which means that

$$\mathbf{S}_{\text{sequential}}^{-1} = \frac{1}{h^2} \begin{bmatrix} -1 & 4 \\ & -1 \end{bmatrix}.$$
 (55)



Figure 4: Convergence histories for different anisotropies  $\epsilon$ : (a) two GaBP sweeps and (b) three GaBP sweeps for presmoothing and postsmoothing. In both cases, the fine grid consists of  $2^6 + 1$  points, and the coarsest grid consists of  $2^3 + 1$  points along each coordinate line. For comparison, if  $\epsilon = 10^{-3}$ , multigrid with Gauss-Seidel smoother (3 presmoothing and postsmoothing sweeps) converges after ~ 400 iterations. The sharp drops of the residual are the result of the cumulative effect that eludes explanation via Local Fourier Analysis. Namely, for small epsilon, vertical lines are effectively decoupled from each other. Scheme (a) needs 15 iterations to solve exactly systems of linear equations for each line, and scheme (b) need 10 iterations. See Section 4.1 for details.

Then, for the smoothing factor we have

$$\mu = \max_{\theta \in \text{high}} \sqrt{\left| \frac{\cos(\theta_1 - \theta_2) + 1}{4\cos(\theta_1) + 4\cos(\theta_2) - \cos(\theta_1 - \theta_2) - 9} \right|} = \frac{1}{2}, \quad (56)$$

for  $\theta_2 = \frac{\pi}{2}$  and  $\theta_1 = 2 \arctan \frac{1}{3}$ . This means that the smoothing factor for the sequential GaBP coincides with the one for sequential Gauss-Seidel iteration scheme [31, Example 4.3.4]. It is also clear that for the anisotropic problem

$$\frac{1}{h^2} \begin{bmatrix} -1 \\ -\epsilon & 2(1+\epsilon) \\ -1 \end{bmatrix} u_{ij} = f_{ij}, \tag{57}$$

both Gauss-Seidel scheme and sequential GaBP lose their smoothing properties when  $\epsilon \to 0$ . However, numerical experiments (figure 4) show that the convergence rate of GaBP does not depend on  $\epsilon$ . An explanation for this particular case is straightforward. For sufficiently small  $\epsilon$  equations for each vertical line (i.e., in y direction) are independent. GaBP is an exact solver for trees. The single multigrid iteration eliminates variables from 2+2=4 neighbours in case 4a and from 3+3=6 neighbours in case 4b. When messages cover the whole line of  $2^6 - 1 = 63$  nodes, the system of linear equations for each vertical line is solved exactly. It gives  $63/4 \sim 15$  iterations for 4a, and  $63/6 \sim 10$  iterations for 4b.

Sweeps	Stencil	GaBP	line GaBP	$\operatorname{GS}$	x/y-GS
1	5 points	18N	38N	9N	14N
1	9 points	32N	N/A	17N	21N
9	5 points	30N	65N	18N	28N
2	9 points	56N	N/A	34N	42N
M > 3	5 points	$12N \cdot M + 6N$	$28N \cdot M + 9N$	$9N \cdot M$	$14N \cdot M$
$M \geq 0$	9 points	$24N \cdot M + 8N$	N/A	$17N \cdot M$	$21N \cdot M$

Table 1: Computational complexity of GaBP (error correction scheme) with precomputed  $\Lambda$  messages in comparison with the classical Gauss-Seidel relaxation schemes. Line GaBP refers to the partition presented in Figure 3a and x/y-GS is a classical line smoother. As one can see from the theory of generalized GaBP, it is not possible to apply line GaBP for 9 points stencil, because large regions do not cover all edges of the original graph. However, one indeed can construct line GaBP smoothers for this case, too, but we do not consider them here.

The effect displayed in Figure 4 is a manifestation of the dynamic nature of GaBP. Even as part of the multigrid it maintains information about all previous iterations. More convergence histories can be found below, in the section with numerical examples. Overall, we conclude that sequential GaBP as part of the multigrid behaves similarly to Gauss-Seidel in the absence of anisotropy, but is substantially more robust in the presence of anisotropy. The behavior captured in Figure 4 also illustrates that Local Fourier Analysis is not an appropriate tool to analyze GaBP.

#### 4.2 Reducing computational complexity

The number of floating point operations per iteration for algorithms 1 and 2 depends on the graph of the matrix A. Here, we consider the operator with the dense 9 point stencil

$$A = \begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix},$$
 (58)

which can come from the second order finite difference approximation of a differential operator containing second and first derivatives. The same analysis for the 5 points stencil is straightforward. For convenience, we split Algorithm 1 (sequential version) into three parts:

- Accumulation stage.  $\Sigma$  and m are computed.
- Update stage. New messages  $\widetilde{\Lambda}$  and  $\widetilde{\mu}$  are constructed from the previous ones.
- Termination stage. The final answer  $m/\Sigma$  is obtained.

We also neglect all effects from boundaries. Under these assumptions, the number of floating point operations for the single sweep GaBP is

$$\#_{\text{GaBP}_{1}} = \underbrace{4N + 8N}_{\text{accumulate}} + \underbrace{8N + 12N}_{\text{update}} + \underbrace{N}_{\text{terminate}} + \underbrace{18N}_{\mathbf{r}=\mathbf{b}-\mathbf{A}\mathbf{x}_{0}} + \underbrace{N}_{\mathbf{x}_{0}} = 52N.$$
(59)

Here, we perform only a half of accumulation stage and a half of update stage, because we do not need to receive messages from nodes that we have not visited yet, nor we need to send messages to already visited nodes.

For lexicographical Gauss-Seidel scheme, the number of floating point operations is  $\#_{\text{LEX GS}} = 17N$ . It means that a single sweep of 1 takes slightly fewer floating point operations than three sweeps of Gauss-Seidel smoother  $\#_{\text{GaBP}_1} \sim 3\#_{\text{LEX GS}}$ . For M sweeps of GaBP one has  $\#_{\text{GaBP}_M} = N(64M - 12)$ . In the context of multigrid, it is important to have a cheap smoother, but even 52N is too expensive. However, it is possible to reduce computational complexity by precomputing all required messages  $\tilde{\Lambda}$ , which depend only on the matrix **A** and not on the right-hand-side vector. For M sweeps of GaBP with precomputed  $\tilde{\Lambda}$ , we have  $\#_{\text{GaBP}_M}^{\tilde{\Lambda}} = N(24M + 8)$ . We summarize all these results regarding the complexity of GaBP in Table 1.

## 5 Numerical examples

In this section, we present numerical experiments with matrices that arise from second-order finite difference approximations of two-dimensional elliptic differential equations with  $(x, y) \in [0, 1]^2$ . The grid is assumed to be uniform and consists of  $2^6 - 1$  inner points along each direction. We use Dirichlet boundary conditions in all the examples. These conditions are not specified directly and should be extracted from the exact solution. In the same vein, the form of the source term g(x, y) (ride-hand side) can be derived from the exact solution and is not given explicitly. In all the experiments, the stopping criterion is  $||r||_{\infty} \leq 2 \cdot 10^{-4}$ . In the tables below,  $N_{\rm it}$  denotes the number of iterations and N is the number of variables. Before we begin the main discussion, we summarize the main properties of the solvers that are used.

#### 5.1 Note about solvers and smoothers

In addition to the number of floating point operations (FLOP) (see Table 1), an important characteristic of a solver is its degree of parallelism, which is provided for various solvers in the following table (for classical methods see e.g. [31, ch. 6]):

Solvers	Process in parallel
parallel GaBP, Jacobi	all point
sequential GaBP, GS	the single point
red-black GaBP, red-black GS	the half of all points
4-colors GaBP, 4-colors GS	the quarter of all points
x-  or  y-GS	a single line
zebra-line GS, alternating zebra GS	a half of all lines
line GaBP	all lines

Note that the line version of GaBP possesses a better degree of parallelism than GS versions. When we consider GaBP as a multigrid component, we always use V-cycle, bilinear restriction, and prolongation operators and LU as a coarse-grid solver. In our notation  $V(J_1, J_2)$  means that the fine grid consists of  $2^{J_1} - 1$  points, the coarse grid of  $2^{J_1-(J_2-1)} - 1$  points; numbers (n,m) before the smoother name refer to the number of pre- and post-smoothing steps.

#### 5.2 GaBP as a stand-alone solver

Classical relaxation methods are rarely used outside the AMG (algebraic multigrid) or GMG (geometric multigrid) to solve linear systems. Nevertheless, we present an example of their performance below. As a linear problem we use the following elliptic boundary value problem:

$$\left(a(x,y)\frac{\partial^2}{\partial x^2} + b(x,y)\frac{\partial^2}{\partial y^2} + \alpha(x,y)\frac{\partial}{\partial x} + \beta(x,y)\frac{\partial}{\partial y}\right)\phi = g, (x,y) \in [0,1]^2;$$

$$a = e^{-x(y+2)} + 10, \ \alpha = \cos\left(\pi\left(x + \frac{y}{2}\right)\right)\cos(2\pi x) + 4;$$

$$b = e^{-2x+2y}\cos^2\left(2\pi\left(2x + \frac{y}{2}\right)\right) + 3, \ \beta = e^{2x-2y}$$
(60)

with g(x, y) and boundary conditions chosen such that  $\phi_{\text{exact}} = \cos(\pi x) \cos(\pi y)$  is the exact solution (this method of manufactured solutions is used in the remaining examples as well). The performance of various methods on this problem is shown in the following table:

Solver	$N_{ m it}$	FLOP, $10^3 \cdot N$
sequential GaBP	1548	99
parallel GaBP	3299	211
GS	3102	53
4-colors GS	2620	45
4-colors GaBP	1865	119
Jacobi	4746	81
error correction 4-colors GaBP $(3)$	706	56

The last line in the table corresponds to the three sweeps of the error correction scheme with precomputed messages  $\Lambda$  (see Section 4.2). We see that GaBP does not provide particular advantages over classical relaxation methods as a standalone solver, even though there are some reports of its excellent performance (see, e.g., [33], [5]). The main bottleneck here is the computational complexity of the scheme. To some extent, one can mitigate this problem by precomputing  $\Lambda$  before the iteration process begins. Still, in this particular situation, both the 4-colors GS and Jacobi provide better alternatives due to their low cost and high degree of parallelism. We observe the analogous behavior for other elliptic problems.

#### 5.3 GaBP as a multigrid smoother

As a rule, relaxation solvers become applicable to real large-scale problems and are competitive with projection methods only in the framework of multigrid schemes. We now present several situations that could potentially challenge state-of-the-art GMG smoothers. Note that in this section, we always use the error correction version of GaBP. Since we apply the same version of multigrid, we compute the FLOP score solely for the smoother and on the fine level only.

#### 5.4 Large mixed derivative

The first equation of interest is of the form

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + (2 - \epsilon) \frac{\partial^2}{\partial x \partial y}\right) \phi = g, \ (x, y) \in [0, 1]^2,$$
  
$$\phi_{\text{exact}} = 2x^3 y^4.$$
(61)

The main problem here is that for small  $\epsilon$ , the ellipticity is almost lost. Below one can see the table with the best in terms of FLOP V-cycle solver of each kind:

	$\epsilon = 0.01$		$\epsilon = -0.01$	
Solver, $V(6,6)$	N <sub>it</sub>	FLOP, $N$	$N_{\rm it}$	FLOP, $N$
4-color GaBP $(0,4)$	23	2392	28	2917
4-color GS $(1,1)$	70	2380	84	2856
zebra-line GS $(0, 1)$	104	2184	127	2667
alternating-zebra GS $(0,1)$	64	2688	78	3276

We can see that the performance of GaBP is comparable with the 4-color GS smoother. So GaBP can be considered to be robust for the almost non-elliptic equations. We also stress that both 4-color GS and GaBP are preferable over the line smoothers for this problem because of their better degree of parallelism.

## 5.5 Boundary layers

The other practically relevant case that is a challenge for standard geometrical smoothers is an advection-diffusion problem in which advection dominates. We take as an example the following problem:

$$\left(-\epsilon \frac{\partial^2}{\partial x^2} - \epsilon \frac{\partial^2}{\partial y^2} + \frac{\partial}{\partial x} + \frac{\partial}{\partial y}\right) \phi(x, y) = 0, \quad (x, y) \in [0, 1]^2,$$

$$\phi_{\text{exact}} = \frac{2e^{-1/\epsilon} - e^{(x-1)/\epsilon} - e^{(y-1)/\epsilon}}{e^{-1/\epsilon} - 1}.$$
(62)

As one can see, there are two boundary layers near x = 1 and y = 1, each of width  $\sim \epsilon$ . The solution is not large, but the derivative is  $\sim 1/\epsilon$ . The table below shows the performance results.

	$\epsilon =$	= 0.02	$\epsilon = 0.01$		
Solver, $V(6,6)$	$N_{\rm it}$	FLOP, $N$	N <sub>it</sub>	FLOP, $N$	
red-black GaBP $(5,0)$	5	330	3	198	
red-black GS $\forall (n,m)$	diverge		diverge		
zebra-line GS $(2,2)$	5	280	diverge		
alternating-zebra GS $(1,1)$	4	224	3	168	
line GaBP $(0,2)$	5	325	5	325	

We observe two interesting trends. First, for  $\epsilon = 0.02$ , it is not possible to apply the red-black GS. However, it is still possible to construct a smoother from GaBP, using a sufficiently large number of sweeps. Second, the line GaBP smoother with a reasonable amount of steps performs nearly as well as the alternating-zebra GS. Still, since the GABP can process all lines simultaneously, we conclude that it can outperform classical geometric smoothers for such advection-dominated elliptic problems. Furthermore, if the precomputation of  $\Lambda$  is affordable from the perspective of the additional storage required, it is far better to use the red-black GaBP smoother.

#### 5.6 Inner layers

We consider another advection-dominated diffusion problem

$$\left(\epsilon \frac{\partial^2}{\partial x^2} + \epsilon \frac{\partial^2}{\partial y^2} + x \frac{\partial}{\partial x} + y \frac{\partial}{\partial y}\right) \phi = g, \ (x, y) \in [0, 1]^2,$$
  
$$\phi_{\text{exact}} = e^{-(x+y-1)^2/\epsilon}.$$
(63)

This equation differs from (62) in two respects: 1) the solution has two inner layers, and 2) they are not aligned with the coordinate lines. Now the performance is as follows.

	$\epsilon =$	0.015	$\epsilon = 0.01$	
Solver, $V(6,6)$	$N_{\rm it}$	FLOP, $N$	$N_{\rm it}$	FLOP, $N$
red-black GaBP $(3,0)$	7	294	13	546
red-black GS $\forall (n,m)$	diverge		diverge	
zebra-line GS $(2,0)$	9	252	diverge	
alternating-zebra GS $(1,1)$	4	224	5	280
line GaBP $(0,2)$	8	520	8	520

For this problem we can see the same pattern as for the previous example. The classical red-black solver is unable to smooth the error, whereas the GaBP-based color iteration scheme works fine. Additionally, due to its excellent degree of parallelism, the red-black GaBP significantly outperforms the alternating-zebra GS smoother.

### 5.7 Stretched grid

Another situation of practical interest is given by the following problem:

$$\left(u\left(x|p,\eta\right)\frac{\partial^2}{\partial x^2} + u\left(y|p,\eta\right)\frac{\partial^2}{\partial y^2}\right)\phi = g, (x,y)\in[0,1]^2,$$

$$u\left(x|p,\eta\right) = 1 + \left(\left(x - \frac{1}{2}\right)^2 + \eta\right)^p/\epsilon,$$

$$\phi_{\text{exact}} = \cos(2\pi(x+y))\sin(2\pi(x-y)).$$
(64)

To understand this problem, consider the finite difference discretization of the Laplace equation on a grid which is highly concentrated near the edges of the domain. If one denotes  $a_{\rm in}$  and  $a_{\rm bn}$  to be characteristic scales of the coefficients, related to inner and boundary points respectively, the ratio  $a_{\rm bn}/a_{\rm in}$  will be large. We achieve the same effect in equation (64) on the uniform grid by multiplying second derivatives by positive terms of the form  $\left(1 + \left(\left(x - \frac{1}{2}\right)^2 + \eta\right)^p / \epsilon\right)\right)$  which are approximately equal to 1 inside the domain, but grow rapidly to large values near the boundaries. The table below shows thee results.

	$p, \eta,$	$\epsilon = 20, 1/2, 10^{-6}$	$p, \eta,$	$\epsilon = 20, 1/2, 8 \cdot 10^{-8}$
Solver, $V(6,6)$	$N_{\rm it}$	FLOP, $N$	$N_{\rm it}$	FLOP, $N$
red-black GaBP $(3,0)$	18	756	23	966
red-black GS $(3,0)$	68	1836	97	2619
zebra-line GS $(4,0)$	50	2800	72	4032
alternating-zebra GS $(1,1)$	10	560	12	672
line GaBP $(0,2)$	20	1300	23	1495

In this table, the first set of parameters  $p, \eta, \epsilon = 20, 1/2, 10^{-6}$  corresponds to the linear stretching of the grid by a factor of ~ 40 and the second  $p, \eta, \epsilon =$ 

 $20, 1/2, 8 \cdot 10^{-8}$  to the linear stretching by a factor of ~ 160. We conclude that there is a version of the red-black GaBP with excellent convergence rate and good degree of parallelism. The same is true for the line GaBP smoother. Both of them can be used as an alternative to the classical alternating-zebra smoother.

#### 5.8 Comparison with a projection method

For the sake of completeness, we also give an example of the performance of BiCGSTAB and GaBP-based multigrid for problem (60).

Solver	$N_{\rm it}$	FLOP, $10^3 \cdot N$
V(6,6), 4-color GaBP $(1,1)$	21	$\sim 3$
BiCGSTAB	255	$\sim 38$

As one may have anticipated, the projection method without a suitable preconditioner cannot outperform the geometric multigrid.

Overall, based on the presented result, we conclude that different versions of GaBP perform either comparably well or better than the state-of-the-art smoothers for GMG. The main disadvantage of GaBP is its computational complexity. Even though with a precomputed  $\Lambda$ , one can substantially decrease the number of FLOPs, the cost of a single iteration is still higher than for the classical smoothers. However, its clear advantages are the robustness and the degree of parallelism. The former allows one to construct new point-based relaxation schemes for situations where classical point-based relaxation methods fail. And the latter enables the line GaBP to outperform the alternating-zebra GS smoother.

## 6 Conclusions

In this paper, we have introduced a new class of solvers for linear systems that are based on the generalized belief propagation algorithm. The solvers work for both symmetric and nonsymmetric matrices. We show how to reduce the complexity of the resulting algorithm in comparison to that of the straightforward application of the generalized belief propagation. A clear connection between the block LU decomposition and the new algorithm is established. Existing proofs for symmetric systems are generalized to nonsymmetric systems, and two new proofs for a block version of the GaBP are given. Furthermore, we show how to use the geometric multigrid to accelerate the GaBP, which with a precomputed  $\tilde{\Lambda}$  results in a robust solver with the same computational complexity as the one based on the Gauss-Seidel smoother.

We have demonstrated the performance of the new algorithm with several examples of boundary-value problems of varying complexity. The numerical experiments show that the GaBP is a competitive alternative to classical relaxation schemes. The reason for the good performance of GaBP is that it retains some information about all the previous stages, whereas the Gauss-Seidel, Jacobi, and Richardson solvers do not. Even though large computational overhead is a disadvantage of GaBP, the problem can be alleviated within the framework of the multigrid scheme at the expense of additional storage and precomputation of some messages. Moreover, as part of the multigrid, the GaBP not only smooths high-frequency components of the error, but also effectively decreases the low-frequencies. Our numerical experiments indicate that this feature promotes additional robustness. For example, the convergence rate of GaBP for anisotropic model elliptic problem given by  $(\partial_x^2 + \epsilon \partial_y^2) u(x, y) = 0$  does not depend on  $\epsilon$ , which is a somewhat unexpected result. Moreover, in the case of sharp inner and boundary layers, stretched grids, and large mixed derivatives, GaBP retains smoothing properties and performs better than the state-of-theart geometrical smoothers.

The generalized GaBP introduced in the present work is in some sense a block version of the regular GaBP. Our numerical experiments show that the convergence rate increases with the size of the blocks such that after certain scale the generalized GaBP can compete with Krylov subspace methods (see a numerical example at [1]). However, the computational cost increases as well. In practical applications, one should balance these two tendencies to construct an optimal solver. Some considerations about the choice of the blocks can be found in [34], [37], but the issue is not yet fully resolved.

The generalized GaBP and GaBP as its particular case come from a domain of variational inference. To the best of our knowledge, there is currently no systematic analysis of the general relationship between deterministic/probabilistic inference and linear solvers. In our opinion, such a link may be useful for new interpretation of known techniques and provide insights that may lead to more efficient algorithms for numerical linear algebra.

## Acknowledgement

The author is indebted to Dr. Aslan Kasimov for valuable suggestions.

## References

- [1] https://github.com/VLSF/GaBP\_solvers.
- [2] Herbert Amann, Joachim Escher, Silvio Levy, and Matthew Cargo. Analysis, volume 1. Springer, 2005.
- [3] Guozhong An. A note on the cluster variation method. Journal of Statistical Physics, 52(3-4):727-734, 1988.

- [4] Simon Bartels, Jon Cockayne, Ilse CF Ipsen, and Philipp Hennig. Probabilistic linear solvers: A unifying view. arXiv preprint arXiv:1810.03398, 2018.
- [5] Danny Bickson. Gaussian belief propagation: Theory and aplication. arXiv preprint arXiv:0811.2518, 2008.
- [6] Christopher M. Bishop. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag, Berlin, Heidelberg, 2006.
- [7] Jon Cockayne, Chris J Oates, Ilse CF Ipsen, Mark Girolami, et al. A bayesian conjugate gradient method. *Bayesian Analysis*, 2018.
- [8] Timothy A Davis, Sivasankaran Rajamanickam, and Wissam M Sid-Lakhdar. A survey of direct methods for sparse linear systems. Acta Numerica, 25:383–566, 2016.
- [9] Yousef El-Kurdi, Dennis Giannacopoulos, and Warren J Gross. Relaxed gaussian belief propagation. In 2012 IEEE International Symposium on Information Theory Proceedings, pages 2002–2006. IEEE, 2012.
- [10] Gal Elidan, Ian McGraw, and Daphne Koller. Residual belief propagation: Informed scheduling for asynchronous message passing. In *Proceedings* of the Twenty-Second Conference on Uncertainty in Artificial Intelligence, UAI'06, pages 165–173, Arlington, Virginia, United States, 2006. AUAI Press.
- [11] Gene H Golub and Charles F Van Loan. Matrix computations, volume 3. JHU press, 2012.
- [12] Philipp Hennig. Probabilistic interpretation of linear solvers. SIAM Journal on Optimization, 25(1):234–260, 2015.
- [13] Tom Heskes. On the uniqueness of loopy belief propagation fixed points. Neural Computation, 16(11):2379–2413, 2004.
- [14] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. Journal of research of the National Bureau of Standards, 49:409–436, 1952.
- [15] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, New York, NY, USA, 2nd edition, 2012.
- [16] Ryoichi Kikuchi. A theory of cooperative phenomena. *Physical review*, 81(6):988, 1951.
- [17] 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(Oct):2031–2064, 2006.

- [18] Thomas P Minka. Expectation propagation for approximate bayesian inference. In Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence, pages 362–369. Morgan Kaufmann Publishers Inc., 2001.
- [19] Houman Owhadi. Multigrid with rough coefficients and multiresolution operator decomposition from hierarchical information games. SIAM Review, 59(1):99–149, 2017.
- [20] Judea Pearl. Probabilistic reasoning in intelligent systems: networks of plausible inference. Elsevier, 2014.
- [21] Alessandro Pelizzola. Cluster variation method in statistical physics and probabilistic graphical models. *Journal of Physics A: Mathematical and General*, 38(33):R309, 2005.
- [22] Kurt Hermann Plarre and PR Kumar. Extended message passing algorithm for inference in loopy gaussian graphical models. Ad Hoc Networks, 2(2):153–169, 2004.
- [23] Dorit Ron, Ilya Safro, and Achi Brandt. Relaxation-based coarsening and multiscale graph organization. *Multiscale Modeling & Simulation*, 9(1):407–423, 2011.
- [24] Y. Saad. Iterative Methods for Sparse Linear Systems. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edition, 2003.
- [25] Yousef Saad and Henk A Van Der Vorst. Iterative solution of linear systems in the 20th century. In Numerical Analysis: Historical Developments in the 20th Century, pages 175–207. Elsevier, 2001.
- [26] Ori Shental, Paul H Siegel, Jack K Wolf, Danny Bickson, and Danny Dolev. Gaussian belief propagation solver for systems of linear equations. In *Information Theory*, 2008. ISIT 2008. IEEE International Symposium on, pages 1863–1867. IEEE, 2008.
- [27] Jonathan Richard Shewchuk et al. An introduction to the conjugate gradient method without the agonizing pain, 1994.
- [28] Gilbert W Stewart. Matrix Algorithms: Volume 1: Basic Decompositions, volume 1. Siam, 1998.
- [29] Erik B Sudderth, Martin J Wainwright, and Alan S Willsky. Embedded trees: Estimation of gaussian processes on graphs with cycles. *IEEE Trans*actions on Signal Processing, 52(11):3136–3150, 2004.
- [30] Jok M Tang and Yousef Saad. A probing method for computing the diagonal of a matrix inverse. Numerical Linear Algebra with Applications, 19(3):485–501, 2012.

- [31] Ulrich Trottenberg, Cornelius W Oosterlee, and Anton Schuller. *Multigrid*. Elsevier, 2000.
- [32] Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. Foundations and Trends® in Machine Learning, 1(1-2):1-305, 2008.
- [33] Yair Weiss and William T Freeman. Correctness of belief propagation in gaussian graphical models of arbitrary topology. In Advances in neural information processing systems, pages 673–679, 2000.
- [34] Max Welling. On the choice of regions for generalized belief propagation. In Proceedings of the 20th conference on Uncertainty in artificial intelligence, pages 585–592. AUAI Press, 2004.
- [35] Jinchao Xu and Ludmil Zikatanov. Algebraic multigrid methods. Acta Numerica, 26:591–721, 2017.
- [36] Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Bethe free energy, Kikuchi approximations, and belief propagation algorithms. Technical Report TR2001-16, MERL - Mitsubishi Electric Research Laboratories, Cambridge, MA 02139, May 2001.
- [37] Jonathan S Yedidia, William T Freeman, and Yair Weiss. Generalized belief propagation. In Advances in neural information processing systems, pages 689–695, 2001.
- [38] Jonathan S Yedidia, William T Freeman, and Yair Weiss. Understanding belief propagation and its generalizations. *Exploring artificial intelligence* in the new millennium, 8:236–239, 2003.
- [39] Jonathan S Yedidia, William T Freeman, and Yair Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on information theory*, 51(7):2282–2312, 2005.

# Appendices

## A Consistency of GaBP

Here, following [33] we prove

**Theorem 2.1.** If there is  $N \in \mathbb{N}$  such that  $\widetilde{\mu}_e^{(N+k)} = \widetilde{\mu}_e^{(N)}$ ,  $\widetilde{\Lambda}_e^{(N+k)} = \widetilde{\Lambda}_e^{(N)}$  for all  $e \in \mathcal{E}$  and for any  $k \in \mathbb{N}$ , then  $\mu_i^{(N+k)} = \mu_i^{(N)} = (\mathbf{A}^{-1}\mathbf{b})_i$ .

That is, if there is a steady state under mapping (21), the solution given by the GaBP rules is exact.



Figure 5: (a) – matrix with nonzero elements denoted by \*; (b) – directed graph corresponding to the matrix. Note that by our convention  $e_{ij}$  agrees with  $A_{ji}$  not  $A_{ij}$ ; (c) – computation tree of depth 3 for the first node  $T_3(x_1)$  generated by a flood schedule. The subtree inside the box is  $T_2(x_1)$ .

We note that the proof in [33] also holds for the nonsymmetric case. We present a slightly different version of their reasoning, without referencing graphical models for normal distribution.

The first concept that we need is a computation tree, which captures the order of operations under the GaBP iteration scheme. The computation tree contains copies of vertices and edges of the graph corresponding to **A**. The matrix is supposed to be fixed so the computation tree depends on the root node i and the number of steps n. We denote it by  $T_n(x_i)$ . To obtain  $T_n(x_i)$  from  $T_{n-1}(x_i)$ , we consider each vertex  $m \in \mathcal{V}_{T_{n-1}(x_i)}$  that has no incidence edges, find the corresponding variable on the graph of A, add to  $T_{n-1}(x_i)$  copies of each neighbour k of m such that  $e_{km} \in \mathcal{E}_A$  except for l for which  $e_{ml} \in \mathcal{E}_{T_{n-1}(x_i)}$ . The example of the tree  $T_3(x_1)$  is in figure 5c, the  $T_2(x_1)$  in the dashed box exemplifies the recursion process.

By the construction of the computation tree, the following proposition is true.

**Proposition A.1.** If  $x_i^{(n)}$  is the solution on the n-th step of the algorithm 1, then it coincides with the one obtained after the elimination of all variables but  $x_i$  (the root) from the computation tree  $T_n(x_i)$ .

To relate the matrix **B** of the computation tree  $T_n(x_i)$  to the matrix **A**, we define the matrix **O** [33, eq. 15] that connects original variables with copies

$$\mathbf{y} = \mathbf{O}\mathbf{x}, \mathbf{d} = \mathbf{O}\mathbf{b},\tag{65}$$

or, more precisely,  $y_j$  is a copy of  $x_i \Rightarrow O_{ji} = 1$  and  $\sum_i O_{ji} = 1$ . For example,

matrix **O** for the tree in figure 5c is

$$\mathbf{O}^{T} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}.$$
 (66)

Now it is not hard to establish the connection between **B** and **A** [33, eq. 17]

$$\mathbf{BO} + \mathbf{E} = \mathbf{OA},\tag{67}$$

where **E** is nonzero only for the subset of variables that n steps away from the root node on the computation tree  $T_n(x_i)$ . The final part of the proof depends on the following statement [33, Periodic beliefs lemma].

**Proposition A.2.** If there is  $N \in \mathbb{N}$  such that  $\widetilde{\mu}_e^{(N+k)} = \widetilde{\mu}_e^{(N)}$ ,  $\widetilde{\Lambda}_e^{(N+k)} = \widetilde{\Lambda}_e^{(N)}$ for all  $e \in \mathcal{E}$  and for any  $k \in \mathbb{N}$ , then it is possible to construct an arbitrary large computation tree  $T_M(x_i)$  for any root node  $x_i$  such that  $\mathbf{O}\boldsymbol{\mu}^{(N)} = \widetilde{\mathbf{B}}^{-1}\widetilde{\mathbf{d}}$ . Where  $\widetilde{B}_{ij} \neq B_{ij}$  and  $\widetilde{d}_i \neq d_i$  only for i = j that are M steps away from the root node.

The crucial part here is that not only the solution for the root node coincides with the steady state solution of GaBP, but also the same is true for all the variables on the modified computation tree.

The proof is as follows. First, following the recursion procedure, we construct a computation tree of desired depth M. Then we continue to grow the tree till the subtrees of nodes M steps away from the root reach the depth N which corresponds to the steady state of GaBP. Now, elimination of subtrees results in the desired modified tree with the matrix  $\tilde{\mathbf{B}}$  and the right-hand side  $\tilde{\mathbf{d}}$ .

Since we can construct an arbitrary modified computation tree, we can always get for arbitrary large  ${\cal M}$ 

$$\mathbf{BO} = \mathbf{OA} \text{ for the first } M \text{ rows.}$$
(68)

And we know that by construction of the modified computation tree

$$\widetilde{\mathbf{BO}}\boldsymbol{\mu}^{(N)} = \widetilde{\mathbf{d}}.$$
(69)

So we conclude that

$$\mathbf{OA}\boldsymbol{\mu}^{(N)} = \mathbf{Ob} \text{ for the first } M \text{ rows.}$$
 (70)

Note, that  $\mathbf{O}^T \mathbf{O}$  is a diagonal matrix that counts the number of copies of each variable, therefore we can always choose M large enough to make det  $(\mathbf{O}^T \mathbf{O}) \neq 0$  and  $\mathbf{A}\boldsymbol{\mu}^{(N)} = \mathbf{b}$  which means that the iterative scheme defined by the algorithm 1 is consistent.

## **B** Convergence of GaBP

Here we present the version of the proof from [17] that extends to nonsymmetric matrices. Our modifications are relatively minor, but for the sake of logical coherence, we reproduce here the minimal set of arguments from [17] tuning definitions and proposition when needed. The main result of this section is

**Theorem 2.2.** If  $A_{ii} \neq 0 \ \forall i$ ,  $|\widetilde{R}|_{ij} = (1 - \delta_{ij}) \frac{|A_{ij}|}{|A_{ii}|}$ , and  $\rho(|\widetilde{\mathbf{R}}|) < 1$ , then the Algorithm 1 converges to the solution  $\mathbf{x}^* = \mathbf{A}^{-1}\mathbf{b}$  for any  $\mathbf{b}$ .

The whole idea of the proof [17] is to relate GaBP operations with the recursive update of the weights of walks on the graph, corresponding to the matrix **A**. For the start, we define a walk w as a an ordered set of vertices  $w = (i_1, i_2, \ldots, i_{l(w)})$  where l(w) is a length of the walk w and  $\forall k < l(w) \Rightarrow e_{i_k i_{k+1}} \in \mathcal{E}$ . Each walk possesses a weight

$$\phi(w) = A_{i_{l(w)}i_{l(w)-1}} \cdots A_{i_{3}i_{2}}A_{i_{2}i_{1}}.$$
(71)

Note that the order is backward, which is a consequence of our definition of the directed graph. For the symmetric matrix, when the order is not essential, the equation (71) coincides with the weight defined in [17, 3.1]. Now, if we have a system  $\mathbf{A}\mathbf{x} = \mathbf{b}$ , we can rescale it using  $\tilde{b}_j = b_j/A_{jj}$ . This procedure is valid for any  $\mathbf{A}$  with nonzero diagonal and results in the equivalent system

$$\widetilde{\mathbf{A}}\mathbf{x} = \widetilde{\mathbf{b}}, \ \widetilde{A}_{ij} = \delta_{ij} + (1 - \delta_{ij}) \frac{A_{ij}}{A_{ii}} \equiv \delta_{ij} - \widetilde{R}_{ij}$$
(72)

It is possible to represent the solution of (72) in the form of Neumann series (see [15, ch. 5]) because  $\rho\left(\widetilde{\mathbf{R}}\right) < 1$  and therefore

$$\widetilde{\mathbf{A}}^{-1} = \left(\mathbf{I} - \widetilde{\mathbf{R}}\right)^{-1} = \sum_{n=0}^{\infty} \widetilde{\mathbf{R}}^n.$$
(73)

However, for being able to rearrange terms in the sum as necessary, which is sufficient to rewrite the inverse matrix using walks, one needs to require absolute convergence which is  $\rho(|\widetilde{\mathbf{R}}|) < 1$ . Having this condition it is not hard to prove [17, Proposition 1, Proposition 5]

**Proposition B.1.** If 
$$\rho(|\widetilde{\mathbf{R}}|) < 1$$
, then  $\widetilde{A}_{ij}^{-1} = \sum_{w:j \to i} \phi(w)$  and  $x_i^* \equiv \left(\widetilde{\mathbf{A}}^{-1}\widetilde{\mathbf{b}}\right)_i = \sum_{k \in \mathcal{V}} \sum_{w:k \to i} \phi(w)\widetilde{b}_k$ .

Here, by  $w: j \to i$  we mean the set of walks which start from the vertex j and end at the vertex i. If one defines [17, 3.2] sets of single-visit  $k \stackrel{\backslash i}{\to} i$  and single-revisit  $i \stackrel{\backslash i}{\to} i$  walks by all walks which are not visiting the node i in

between given start and end points, the sum over walks can be decomposed [17, eq. 12, 13; Proposition 9]

$$x_{i}^{\star} = \left(\widetilde{b}_{i} + \sum_{k \in \mathcal{V}} \left[\widetilde{b}_{k} \sum_{w: k \stackrel{\backslash i}{\to} i} \phi(w)\right]\right) / \left(1 - \sum_{w: i \stackrel{\backslash i}{\to} i} \phi(w)\right).$$
(74)

The decomposition follows from "topological" considerations alone which depend only on the structure of walks and not on the particular definition of the weight. The last result that we need is [17, Lemma 18]

**Proposition B.2.** For each finite length walk  $k \to j$  on directed graph of the matrix A there is n and unique walk on the computation tree  $T_n(x_i)$ .

Now, if we can relate update rules (21) with the recursive structure of walks on a tree, the proof of the proposition 2.2 is done.

On the tree, for each vertex i, the sum over single-revisit walks splits into sums over subtrees  $T_{k\cup i}$ , which are maximal connected parts that contain i and among N(i), only k. Then

$$\sum_{\substack{w:i \to i} \\ w : i \to i} \phi(w) = \sum_{k \in N(i)} \sum_{\substack{w:i \to i \\ w \in T_{k \cup i}}} \phi(w), \tag{75}$$

but the sums over subtrees  $T_{k\cup i}$  can be written as a sum over  $T_{k\setminus i} \equiv T_{k\cup i}\setminus\{i\}$ ,

$$\sum_{\substack{w:i\stackrel{i}{\to}i\\w\in T_{k\cup i}}}\phi(w) = \frac{\widetilde{R}_{ki}\widetilde{R}_{ik}}{1 - \sum_{\substack{w:k\stackrel{i}{\to}k\\w\in T_{k\setminus i}}}\phi(w)} = \frac{\widetilde{R}_{ki}\widetilde{R}_{ik}}{1 - \sum_{\substack{m\in N(k)\setminus i\\w\in K_{k\setminus k}}}\phi(w)},$$
(76)

where we used [17, eq. 12, Proposition 9]

$$\sum_{w:k \to k} \phi(w) = \frac{1}{1 - \sum_{w:k \to k} \phi(w)}.$$
(77)

Using the definition of  $\tilde{\mathbf{R}}$ , it is easy to see that if one denotes

$$-\frac{A_{ii}}{A_{ki}}\sum_{\substack{w:i\stackrel{i}{\to}i\\w\in T_{k\cup i}}}\phi(w) = \widetilde{\Lambda}_{ki},\tag{78}$$

then the update rule (76) coincides with the one for  $\hat{\Lambda}$  in (21). Note that (78) is well defined because if  $A_{ki} = 0$ , there is no contribution from this particular subtree, and we do not need to use the walk from there. In the same vein, the sum in the numerator of (74) can be decomposed

$$\sum_{k \in \mathcal{V}} \left[ \widetilde{b}_k \sum_{\substack{w:k \to i \\ w:k \to i}} \phi(w) \right] = \sum_{m \in N(i)} \sum_{\substack{k \in T_{m \cup i}}} \left[ \widetilde{b}_k \sum_{\substack{w:k \to i \\ w \in T_{m \cup i}}} \phi(w) \right].$$
(79)



Figure 6: (a) – partition of the original graph on large regions; (b) – flat representation of the two-layer region graph; (c) – computation tree for the generalized GaBP.

Again, using the sum over subtrees  $T_{k \setminus i}$ 

$$\sum_{k \in T_{m \cup i}} \left[ \widetilde{b}_k \sum_{\substack{w: k \stackrel{\backslash i}{\to} i \\ w \in T_{m \cup i}}} \phi(w) \right] = \widetilde{R}_{im} \sum_{\substack{k \in T_{m \setminus i}}} \left[ \widetilde{b}_k \sum_{\substack{w: k \to m \\ w \in T_{m \setminus i}}} \phi(w) \right],$$
(80)

decomposition on single-visit walks [17, eq. 13] and equations (78), (76), we obtain

$$\frac{\widetilde{\mu}_{mi}\widetilde{\Lambda}_{mi}}{A_{ii}} \equiv \sum_{k \in T_{m \cup i}} \left| \widetilde{b}_k \sum_{\substack{w: k \stackrel{i}{\to} i \\ w \in T_m \cup i}} \phi(w) \right| = 
= \frac{\widetilde{\Lambda}_{mi}A_{mm}}{A_{ii}} \left( \widetilde{b}_m + \sum_{l \in N(m) \setminus i} \sum_{k \in T_{l \cup m}} \left[ \widetilde{b}_k \sum_{\substack{w: k \stackrel{i}{\to} m \\ w \in T_{l \cup m}}} \phi(w) \right] \right).$$
(81)

The parameterization introduced in (81) leads to the same update rule for  $\tilde{\mu}$  as in (21). With that, the sufficient condition, given in proposition 2.2, is established.

## C Consistency of generalized GaBP

Here we prove that the two-layer generalized GaBP is consistent.

**Theorem 3.1.** If there is  $N \in \mathbb{N}$  such that  $\mathbf{m}_{e}^{(N+k)} = \mathbf{m}_{e}^{(N)}$ ,  $\mathbf{\Lambda}_{e}^{(N+k)} = \mathbf{\Lambda}_{e}^{(N)}$ for all  $e \in \mathcal{E}_{\mathcal{R}}$  and for any  $k \in \mathbb{N}$ , then for each large region  $[\mathbf{x}]_{L} \equiv \widetilde{\mathbf{\Lambda}}_{0}^{-1} \widetilde{\mathbf{b}}_{0} = [\mathbf{A}^{-1}\mathbf{b}]_{L}$  (see Algorithm 2 for details). The idea of the proof is the same as for the regular GaBP. One needs to relate, considering the operations of generalized GaBP, equations that the generalized GaBP solves during the N-th step, with the original system of linear equations, and then to show that those systems coincide for a sufficiently large N if steady state exists.

To do so, we introduce a flat version of the region graph (an example is shown in figure 6b) that provides less detailed information about parent-child structure. The flat region graph is an undirected graph  $\{\mathcal{V}, \mathcal{E}\}$ , where  $\mathcal{V}$  is the set of large regions and  $(L, L') \in \mathcal{E}$  if L and L' has at least one common child (the example is in figure 6b).

Now one can introduce the computation tree exactly in the same way as for GaBP. The only difference is that, because of an overlap between large regions, when we add a leaf node, we include overlapping variables to the root node. An example of the computation tree  $T_3(B)$  as well as the  $T_2(B)$  is in figure 6c. By construction of the computation tree, we know that the following is true.

**Proposition C.1.** Elimination of all the variables on the computation tree  $T_N(B)$  leads to the solution  $\mathbf{x}_B$  that coincides with the one on the N-th step of generalized GaBP.

The relation between the matrix  $\mathbf{B}$ , corresponding to the computation tree, and the original matrix  $\mathbf{A}$  is the same as in the equation (67) if one introduces the matrix  $\mathbf{O}$ 

$$O_{ij} = \begin{cases} 1 \text{ if } y_i \text{ is the copy of } x_j, \\ 0 \text{ otherwise.} \end{cases}$$
(82)

Here,  $\mathbf{x}$  are variables on the graph of matrix  $\mathbf{A}$ , and  $\mathbf{y}$  are the ones on the computation tree.

Having the same relation between  $\mathbf{A}$  and  $\mathbf{B}$ , we can repeat the rest of the proof, using the same arguments as in Section A. So it follows that generalized GaBP is consistent and proposition 3.1 is true.

## D Convergence of generalized GaBP

In this section, we present a sufficient condition for the convergence of the twolayer generalized GaBP.

**Theorem 3.2.** If for matrix (47) which is based on partition (46) det  $\mathbf{A}_{ii} \neq 0 \forall i$ and  $\rho\left(\left\|\widetilde{\mathbf{R}}\right\|\right) < 1$  in some operator norm, then two-layer generalized GaBP (algorithm 2) converges to the exact solution  $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ .

The proof consists of two parts. In the first one, we show that single-visit and single-revisit walks on a tree possess the same update rules as generalized GaBP messages. In the second part, we show that it is always possible to reorganize walks on the graph coming from the partition F (see equations (46) and (47)) to restore each walk on a computation tree.

#### D.1 Walk structure on a tree

To complete the first part, we define for a given partition F (equation (46)) of a matrix **A** the weight of a walk  $w = (i_1 i_2 \dots i_L)$  by the product of matrices

$$\phi(w) = \widetilde{\mathbf{R}}_{i_L i_{L-1}} \cdots \widetilde{\mathbf{R}}_{i_3 i_2} \widetilde{\mathbf{R}}_{i_2 i_1}.$$
(83)

In the view of the standard result [2, ch. 8, Theorem 8.9] on absolute convergence in complete finite metric spaces it is possible to rearrange terms of the sum, such that we can formulate the following statement.

**Proposition D.1.** If  $\rho\left(\left\|\widetilde{\mathbf{R}}\right\|\right) < 1$ , then  $\left(\widetilde{\mathbf{A}}^{-1}\right)_{ii} = \sum_{n=0}^{\infty} \left(\widetilde{\mathbf{R}}^{n}\right)_{ii} = \sum_{w:i \to i} \phi(w)$ ,  $\mathbf{x}_{i} \equiv \sum_{j \in \mathcal{V}} \left(\widetilde{\mathbf{A}}^{-1}\right)_{ij} \widetilde{\mathbf{b}}_{j} = \sum_{j \in \mathcal{V}} \sum_{w:j \to i} \phi(w) \widetilde{\mathbf{b}}_{j}$ .

Here we used the same definition for the set of walks as in the Appendix B. Again, [17, eq. 12, 13] allows us to rewrite the diagonal blocks of the inverse matrix and the solution vector using single-visit and single-revisit walks

$$\left(\widetilde{\mathbf{A}}^{-1}\right)_{ii} = \left(\mathbf{I}_{ii} - \sum_{w:i \stackrel{i}{\to} i} \phi(w)\right)^{-1},$$

$$\mathbf{x}_{i} = \left(\widetilde{\mathbf{A}}^{-1}\right)_{ii} \left(\widetilde{\mathbf{b}}_{i} + \sum_{j \in \mathcal{V}} \sum_{w:j \stackrel{i}{\to} i} \phi(w)\widetilde{\mathbf{b}}_{j}\right).$$
(84)

On the tree we can split the sums over contributions from subtrees  $T_{k\cup i}$  for each  $k \in N(i)$ . Therefore, from comparison with algorithm 2, we can deduce that

$$\sum_{\substack{j \in T_{k\cup i} \\ w: j \xrightarrow{\downarrow i} \\ w \in T_{k\cup i}}} \sum_{\substack{w: j \xrightarrow{\downarrow i} \\ w \in T_{k\cup i}}} \phi(w) \widetilde{\mathbf{b}}_j = (\mathbf{A}_{ii})^{-1} \mathbf{m}_{ki}, \quad \sum_{\substack{w: i \xrightarrow{\downarrow i} \\ w \in T_{k\cup i}}} \phi(w) = -(\mathbf{A}_{ii})^{-1} \mathbf{\Lambda}_{ki}.$$
(85)

Messages in algorithm 2 propagate along edges of the region graph, whereas messages that we have just defined flow along edges of a graph of the matrix  $\tilde{\mathbf{R}}$ . To have a more straightforward connection between them, we consider  $\tilde{\mathbf{R}}$  as a matrix originates from the computation tree itself. Under this set of circumstances, there is a one-to-one correspondence between messages (85) and the ones in algorithm 2.

For single-revisit walks, one has

$$\sum_{\substack{w: i \to i \\ w \in T_{k \cup i}}} \phi(w) = (\mathbf{A}_{ii})^{-1} \mathbf{A}_{ik} \sum_{\substack{w: k \to k \\ w \in T_{k \setminus i}}} \phi(w) (\mathbf{A}_{kk})^{-1} \mathbf{A}_{ki} =$$
$$= (\mathbf{A}_{ii})^{-1} \mathbf{A}_{ik} \left( \mathbf{I}_{kk} - \sum_{\substack{w: k \to k \\ w \in T_{k \setminus i}}} \phi(w) \right)^{-1} (\mathbf{A}_{kk})^{-1} \mathbf{A}_{ki} \Rightarrow \qquad (86)$$
$$\Rightarrow \mathbf{A}_{ki} = -\mathbf{A}_{ik} \left( \mathbf{A}_{kk} + \sum_{m \in N(k) \setminus i} \mathbf{A}_{mk} \right)^{-1} \mathbf{A}_{ki}.$$

If we consider the following matrix

$$\left( \begin{pmatrix} 0 & \mathbf{A}_{ik} \\ \mathbf{A}_{ki} & \left( \mathbf{A}_{kk} + \sum_{m \in N(k) \setminus i} \mathbf{\Lambda}_{mk} \right) \end{pmatrix}^{-1} \right)_{ii} = \mathbf{\Lambda}_{ki}^{-1}, \quad (87)$$

one can immediately see that update rules (86) indeed coincide with (33).

For single-visit walks, we have

$$\sum_{\substack{j \in T_{k \cup i} \\ w \in T_{k \cup i}}} \sum_{\substack{w: j \xrightarrow{\lambda i} \\ w \in T_{k \cup i}}} \phi(w) \widetilde{\mathbf{b}}_{j} = - (\mathbf{A}_{ii})^{-1} \mathbf{A}_{ik} \sum_{\substack{j \in T_{k \setminus i} \\ w \in T_{k \setminus i}}} \sum_{\substack{w: j \rightarrow k \\ w \in T_{k \setminus i}}} \phi(w) \widetilde{\mathbf{b}}_{j} = - (\mathbf{A}_{ii})^{-1} \mathbf{A}_{ik} \left( \mathbf{I}_{kk} - \sum_{\substack{w: k \xrightarrow{\lambda k} \\ w \in T_{k \setminus i}}} \phi(w) \right)^{-1} \left( \widetilde{\mathbf{b}}_{k} + \sum_{\substack{j \in T_{k \setminus i} \\ w \in T_{k \setminus i}}} \sum_{\substack{w: j \xrightarrow{\lambda k} \\ w \in T_{k \setminus i}}} \phi(w) \widetilde{\mathbf{b}}_{j} \right),$$
(88)

or using (85), we get

$$\mathbf{m}_{ki} = -\mathbf{A}_{ik} \left( \mathbf{A}_{kk} + \sum_{m \in N(k) \setminus i} \mathbf{\Lambda}_{mk} \right)^{-1} \left( \mathbf{b}_k + \sum_{p \in N(k) \setminus i} \mathbf{m}_{pk} \right).$$
(89)

Since  $\mathbf{m}_{ki} = \mathbf{\Lambda}_{ki} \boldsymbol{\mu}_{ki}$  and

$$\boldsymbol{\mu}_{ki} = \left( \begin{pmatrix} 0 & \mathbf{A}_{ik} \\ \mathbf{A}_{ki} & \left( \mathbf{A}_{kk} + \sum_{m \in N(k) \setminus i} \mathbf{A}_{mk} \right) \end{pmatrix}^{-1} \left( \mathbf{b}_k + \sum_{p \in N(k) \setminus i} \mathbf{m}_{pk} \right) \right)_{ii}$$
(90)

we recover update rules (33). So we conclude that on the computation tree update rules for the two-layer generalized GaBP coincide with recursive relations for the single-visit and single-revisit walks.



Figure 7: (a) – graph of the matrix  $\mathbf{A}$ , each node corresponds to the diagonal block; (b) – refined version of (a), submatrix  $\mathbf{A}_{ii}$  is split by four blocks  $\mathbf{A}_{i_1i_1}, \mathbf{A}_{i_1i_2}, \mathbf{A}_{i_2i_1}, \mathbf{A}_{i_2i_2}$ .

#### D.2 Walk-sums and the graph refinement

The second part of the proof establishes the connection between sets of walks on the graph of the matrix  $\tilde{\mathbf{R}}$  and walks on the computation tree. First, for the matrix (47) we split a single region *i* into two parts  $i_1$  and  $i_2$ 

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{i_1i_1} & \mathbf{A}_{i_1i_2} & \mathbf{A}_{i_1j} & \dots \\ \mathbf{A}_{i_2i_1} & \mathbf{A}_{i_2i_2} & \mathbf{A}_{i_2j} & \dots \\ \mathbf{A}_{ji_1} & \mathbf{A}_{ji_2} & \mathbf{A}_{jj} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}, \mathbf{b} = \begin{pmatrix} \mathbf{b}_{i_1} \\ \mathbf{b}_{i_2} \\ \mathbf{b}_{j} \\ \vdots \end{pmatrix}.$$
(91)

The transformation of the graph is in figure 7. We refer to this procedure as to the elementary refinement of the region i. From the construction of the refined matrix **A**, the following proposition holds.

**Proposition D.2.** There is a one-to-one correspondence between walks on the graph of  $\widetilde{\mathbf{R}}$  and the one obtained by the elementary refinement of the region *i* excluding three situations: 1) walk crosses *i*, 2) walk ends at *i*, 3) walk starts at *i*.

We discuss each of these situations separately. First, we need to introduce a new notation. Let  $k \xrightarrow{M} l$  be the set of walks, where each walk starts from k, ends at l and newer leaves the subset M. It is easy to see that on the refined graph

$$\phi\left(k \stackrel{\{i_1,i_2\}}{\longrightarrow} l\right) = \left(\left(\mathbf{A}_{ii}\right)^{-1}\right)_{lk} \mathbf{A}_{kk}, \text{ where } l, k = \{i_1, i_2\}.$$
(92)

• Walk on  $\widetilde{\mathbf{R}}$  that crosses *i* has a form  $w_{\text{cross}} = (\dots jik \dots)$  (see figure 7a). The weight of this walk is

$$\phi(w_{\text{cross}}) = \cdots \left(\mathbf{A}_{kk}\right)^{-1} \mathbf{A}_{ki} \left(\mathbf{A}_{ii}\right)^{-1} \mathbf{A}_{ij} \cdots .$$
(93)

On the refined graph we can consider the set of all walks that coincides with w outside i. The sum of weight of all these walks is

$$\phi(w)_{\text{refined}} = \sum_{l,k \in \{i_1, i_2\}} \cdots (\mathbf{A}_{kk})^{-1} \mathbf{A}_{kl} \phi\left(l \xrightarrow{\{i_1, i_2\}} k\right) (\mathbf{A}_{ll})^{-1} \mathbf{A}_{lj} \cdots$$
(94)

We see that due to equation (92), weights are the same.

• Walk on  $\widetilde{\mathbf{R}}$  that ends at *i* has a form  $w_{\text{end}} = (\dots ji)$  and a weight

$$\phi(w) = \left(\mathbf{A}_{ii}\right)^{-1} \mathbf{A}_{ij} \cdots .$$
(95)

On the refined graph we have two set of walks

$$\phi(w)_{\text{refined}}^{p} = \sum_{l \in \{i_{1}, i_{2}\}} \phi\left(l \xrightarrow{\{i_{1}, i_{2}\}} p\right) \left(\mathbf{A}_{ll}\right)^{-1} \mathbf{A}_{lj} \cdots, \ p \in \{i_{1}, i_{2}\}$$
(96)

that can be combined to have the same weight. Namely, using (92) we find that

$$\left[ \left( \mathbf{A}_{ii} \right)^{-1} \mathbf{A}_{ij} \cdots \right]_{l\star} = \left( \phi(w)_{\text{refined}}^l \right)_{\star}, \ l = \{i_1, i_2\}.$$
(97)

• Walk on  $\widetilde{\mathbf{R}}$  that starts at *i* has a form  $w_{\text{start}} = (ij \dots)$  and a weight

$$\phi(w_{\text{start}}) = \cdots \left(\mathbf{A}_{jj}\right)^{-1} \mathbf{A}_{ji}.$$
(98)

It is possible to relate this walk to two sets of walks  $w_1 = (i_1 j \dots), w_2 = (i_2 j \dots)$  on the refined graph multiplying by the corresponding inverse matrices

$$\left(\phi(w_{\text{start}})\left(\mathbf{A}_{ii}\right)^{-1}\right)_{\star l} = \sum_{k=\{i_1,i_2\}} \left(\phi(w_k)\phi\left(l \stackrel{\{i_1,i_2\}}{\longrightarrow} k\right)\left(\mathbf{A}_{ll}\right)^{-1}\right)_{\star}, \quad (99)$$

where  $l = \{i_1, i_2\}$ . The re-weight is needed because the original linear system and the refined one are multiplied by different block diagonal matrices and have different inverses.

We know the following two propositions to be true.

**Proposition D.3.** Any computation tree can be, by the set of elementary refinements, turned to a computation tree of GaBP under a proper schedule (see discision before [17, Lemma 18]) operating on the graph of the matrix (47) partitioned according to F.

**Proposition D.4.** For each walk on the graph of the matrix (47), there is a unique walk on a sufficiently large computation tree formed by a proper schedule.

Hence for each walk on the computation tree, it is always possible to find a unique set of walks on the graph of the matrix (47) that has the same weight after the multiplication by an appropriate inverse matrix (see 99). It allows us to conclude that if it is possible to define a walk-sum for matrix (47) (see proposition D.1), walk-sum on the computation tree converges too, so the proposition 3.2 is proven.