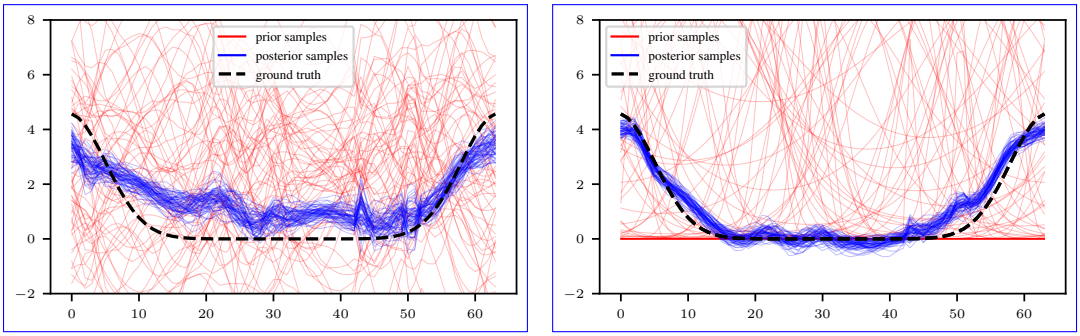
GAUSSIAN ENSEMBLE BELIEF PROPAGATION FOR EFFICIENT INFERENCE IN HIGH-DIMENSIONAL, BLACK-BOX SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Efficient inference in high-dimensional models is a central challenge in machine learning. We introduce the Gaussian Ensemble Belief Propagation (GEnBP) algorithm, which combines the strengths of the Ensemble Kalman Filter (EnKF) and Gaussian Belief Propagation (GaBP) to address this challenge. GEnBP updates ensembles of prior samples into posterior samples by passing low-rank local messages over the edges of a graphical model, enabling efficient handling of high-dimensional states, parameters, and complex, noisy, black-box generation processes. By utilizing local message passing within a graphical model structure, GEnBP effectively manages complex dependency structures and remains computationally efficient even when the ensemble size is much smaller than the inference ~~dimension–a~~ dimension — a common scenario in spatiotemporal modeling, image processing, and physical model inversion. We demonstrate that GEnBP can be applied to various problem structures, including data assimilation, system identification, and hierarchical models, and show through experiments that it outperforms existing belief propagation methods in terms of accuracy and computational efficiency.



(a) GaBP $\widehat{\mathrm{MSE}}$=1.93, $\widehat{\log p}(q|\mathscr{E})=-228$ (b) GEnBP $\widehat{\mathrm{MSE}}$=0.167, $\widehat{\log p}(q|\mathscr{E})=-49.6$

Figure 1: Prior and posterior samples for latent **q** in the 1d system identification problem (Section 4.1). The GEnBP ensemble has size N=64, and N=64 samples are drawn from the GaBP posterior. The GEnBP prior comprises samples; the GaBP is drawn from a Gaussian density with the same moments.

1 INTRODUCTION

We combine the Ensemble Kalman Filter (EnKF) and Gaussian Belief Propagation (GaBP) to perform inference in high-dimensional hierarchical systems. While both methods are well-established, their combination seems novel and empirically outperforms existing approaches in key problems.

~~The EnKF (Evensen, 2003) is widely used for inference in state-space models. It constructs a posterior sample of a hidden state from prior samples by moment-matching the observation-conditional distribution. Although the state update relies on a Gaussian approximation, EnKF never explicitly evaluates the Gaussian density. This approach works well in high-dimensional~~

~~settings, such as climate models (Houtekamer & Zhang, 2016), where computing densities can be prohibitively expensive. However, the standard EnKF is limited to state filtering and doesn't extend to more general inference tasks.~~

GaBP (Yedidia et al., 2005) is a specific Gaussian message-passing algorithm designed for inference in graphical models. It performs density-based inference by leveraging a joint density over the model's state space. This density is approximated as a product of Gaussian factors, defining a graphical model (Koller & Friedman, 2009) over the graph $\mathcal{G}$. Sum-product messages passed along $\mathcal{G}$'s edges help infer target marginals (Pearl, 2008). Message-passing methods shine ~~in~~ various applications, including Bayesian hierarchical models (Wand, 2017), error-correcting codes (Forney, 2001; Kschischang et al., 2001), and Simultaneous Localization and Mapping (SLAM) tasks (Dellaert & Kaess, 2017; Ortiz et al., 2021)~~, and~~ .

Graphical models possess various useful properties, such as permitting inference distributed over many computational nodes ~~Vehtari et al. (2019)~~(Vehtari et al., 2019), domain adaptation (Bareinboim & Pearl, 2013), a natural means for estimating treatment effects (Pearl et al., 2016) , and online, streaming updates (Dellaert & Kaess, 2017; Eustice et al., 2006). They perform ~~exceptionally~~ well with low-dimensional variables but become computationally expensive in high-dimensional scenarios.

The EnKF (Evensen, 2003) is widely used for inference in state-space models. It constructs a posterior sample of a hidden state from prior samples by moment-matching the observation-conditional distribution. Although the state update relies on a Gaussian approximation, EnKF never explicitly evaluates the Gaussian density. This approach works well in high-dimensional settings, such as climate models (Houtekamer & Zhang, 2016), where computing densities can be prohibitively expensive. However, the standard EnKF is limited to state filtering and doesn't extend to more general inference tasks. In recent times, various techniques have extended EnKF to the system-identification setting, where not only states but latent parameters of dynamics are jointly inferred (e.g. Fearnhead & Künsch, 2018; Chen et al., 2023). EnKF is prized for its ability to handle correlated spatial fields efficiently, as with other spatial Gaussian approximations such as the Integrated Nested Laplace Approximation(Rue et al., 2009) and the related Stochastic Partial Differential Equation approach (Lindgren et al., 2011). Unlike those methods, it can handle exploit existing simulators to encode complex physical dynamics, and does not require the jacobian of the simulator. Thus far, there are few methods which extend the attractive simulator-based, high-dimensional inference of the EnKF to the more general setting of graphical models.

GEnBP borrows strength from both EnKF and GaBP, achieving the EnKF's efficiency in high-dimensional data processing and GaBP's capability to handle complex graphical model structures (Table 1). GEnBP is capable of ~~managing~~ inference models with noisy and moderately non-linear observation processes, unknown process parameters, and deeply nested dependencies among latent variables, scaling to millions of dimensions in variables and observations. The key insight is that, despite the name, Gaussian Belief Propagation as commonly used is not truly generic in the class of possible Gaussian approximations (Section 2.2.3). Like GaBP and EnKF, GEnBP uses Gaussian approximations. However, it relies on empirical samples for statistics over non-linear nodes, similar to EnKF, rather than the ~~linearisations~~ linearisation at the mode, used in GaBP.

By using ensembles to represent Gaussian distributions, GEnBP avoids the computational burden of full covariance matrices, enabling efficient belief propagation with potentially superior accuracy.

This approach is highly relevant for many practical problems, including physical and geospatial systems like computational fluid dynamics, geophysical model inversion, and weather prediction. Such systems typically feature high-dimensional representations with spatial correlations induced by known physical dynamics that create a low-rank structure, which empirical covariance approximations can effectively ~~capture~~exploit.

Main Contributions

1. We introduce Gaussian Ensemble Belief Propagation (GEnBP), a novel message-passing method for inference in high-dimensional graphical models
2. GEnBP leverages EnKF-like ensemble statistics instead of the traditional linearization used in GaBP, improving the handling of non-linear relationships and reducing computational complexity.

3. We demonstrate that GEnBP scales to higher dimensions than traditional GaBP while achieving comparable or better accuracy in practically important physical problems, such as spatiotemporal modeling and physical model inversion.
4. GEnBP is enabled by our development of:
   (a) rank-efficient techniques for propagating high-dimensional Gaussian beliefs efficiently, and
   (b) compute-efficient methods for converting between Gaussian beliefs and ensemble approximations without high-dimensional matrix operations,

   which are, to our knowledge, novel contributions in this context.

2 Preliminaries

We ~~introduce the essential notation and~~ will now introduce our notation and essential concepts. Deeper background on belief propagation may be found in Appendix C and on the Ensemble Kalman filter in Appendix D.

2.1 Models as Generative Processes and Densities

We denote random variables in sans-serif font (e.g., x), their corresponding values in boldface (e.g., $\boldsymbol{x}$), and assume all variables have densities, writing $\mathsf{x} \sim p(\boldsymbol{x})$.

The overall model state vector is divided into variables connected through a structural equation model $\mathcal{M}$ (Wright, 1934). Specifically, $\mathcal{M}$ consists of J generating equations $\{\mathcal{P}_j\}_{1 \leq j \leq J}$, where each $\mathcal{P}_j$ is defined as[1]

$$\mathcal{P}_j : \mathbb{R}^{D_{\mathscr{I}_j}} \to \mathbb{R}^{D_{\mathscr{O}_j}}, \quad \mathsf{x}_{\mathscr{I}_j} \mapsto \mathsf{x}_{\mathscr{O}_j}. \tag{1}$$

Each $\mathcal{P}_j$ establishes a relationship between *input* $\mathscr{I}_j$ and *output* $\mathscr{O}_j$ sets.[2] The *ancestral* variables $\mathscr{A} := \bigcup_{\mathscr{I}_j = \emptyset} \mathscr{O}_j$ are those without any inputs. We categorize variables into three types:

- *Evidence* $\mathscr{E}$: Variables that are observed.
- *Latent* $\mathscr{L}$: Unobserved or 'nuisance' variables.
- *Query* $\mathscr{Q}$: Variables whose evidence-conditional distribution is of primary interest.

We ~~focus on ancestral variablesas our queries (~~will assume that queries are always ancestral variables, i.e. that $\mathscr{Q} = \mathscr{A}$~~)~~.

Each generating equation $\mathcal{P}_j$ defines how the outputs $\mathscr{O}_j$ are produced from the inputs $\mathscr{I}_j$. This structure allows us to represent the joint density of all variables as a product of conditional densities,

$$p(\boldsymbol{x}) = \prod_{j=1}^{J} p(\boldsymbol{x}_{\mathscr{O}_j} \mid \boldsymbol{x}_{\mathscr{I}_j}).$$

Our main objective is to compute the posterior distribution

$$p(\boldsymbol{x}_{\mathscr{Q}} | \mathsf{x}_{\mathscr{E}} = \boldsymbol{x}^*_{\mathscr{E}}) = \frac{\int p(\boldsymbol{x}_{\mathscr{Q}}, \boldsymbol{x}_{\mathscr{L}}, \boldsymbol{x}_{\mathscr{E}} = \boldsymbol{x}^*_{\mathscr{E}}) \mathrm{d}\boldsymbol{x}_{\mathscr{L}}}{\int p(\boldsymbol{x}_{\mathscr{Q}}, \boldsymbol{x}_{\mathscr{L}}, \boldsymbol{x}_{\mathscr{E}} = \boldsymbol{x}^*_{\mathscr{E}}) \mathrm{d}\boldsymbol{x}_{\mathscr{Q}} \mathrm{d}\boldsymbol{x}_{\mathscr{L}}}. \tag{2}$$

i.e. to update our beliefs about the ancestral variables by assimilating observations of the evidence variables while accounting for the latent variables.

2.2 Belief Propagation in Graphical Models

In graphical models, we associate a graph structure $\mathcal{G}$ with the factorization of the model density to compute the target distribution Equation 2. In this work, we use Belief Propagation (BP) to mean specifically *loopy belief propagation in factor graphs* (Kschischang et al., 2001). Additional details are in Appendix C.

[1] Each $\mathcal{P}_j$ can be a stochastic function, meaning that it is a deterministic function $\mathsf{x}_{\mathscr{O}_j} = \mathcal{P}_j(\mathsf{x}_{\mathscr{I}_j}, \mathsf{n}_j)$ with an unobservable noise term n_j. For brevity, we omit the noise terms in the notation.

[2] These sets can be empty.

The factor graph is constructed by representing each conditional probability $p(\boldsymbol{x}_{\mathscr{O}_j} \mid \boldsymbol{x}_{\mathscr{I}_j})$ as a *factor potential* f_j, defining $\mathscr{N}_j \coloneqq \mathscr{O}_j \cup \mathscr{I}_j$, such that

$$p(\boldsymbol{x}) = \prod_j p\left(\boldsymbol{x}_{\mathscr{O}_j} | \boldsymbol{x}_{\mathscr{I}_j}\right) = \prod_j f_j\left(\boldsymbol{x}_{\mathscr{N}_j}\right). \tag{3}$$

The factor graph $\mathcal{G}$ is bipartite, consisting of

- **Factor nodes**: One for each factor potential $f_j(\boldsymbol{x}_{\mathscr{N}_j})$.
- **Variable nodes**: One for each variable $\mathbf{x}_k$.

Edges connect each factor node j with each variable node in its neighborhood $\mathscr{N}_j$.

Belief Propagation (BP) estimates the *belief* over a query node $b_{\mathcal{G}}(\boldsymbol{x}_k)$ by integrating out all other variables,

$$b_{\mathcal{G}}\left(\boldsymbol{x}_k\right) \approx \int p(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}_{\setminus k}. \tag{4}$$

Proposition 1 (Belief Propagation on Factor Graphs)**.** *By iteratively and synchronously propagating the following messages between all nodes in the factor graph,*

$$m_{f_j \to \mathbf{x}_k} = \int \left(f_j\left(\boldsymbol{x}_{\mathscr{N}_j}\right) \prod_{i \in \mathscr{N}_j \setminus k} m_{\mathbf{x}_i \to f_j} \right) \mathrm{d}\boldsymbol{x}_{\mathscr{N}_j \setminus k}, \tag{5}$$

$$m_{\mathbf{x}_k \to f_j} = \prod_{s \in \mathscr{N}_k \setminus j} m_{f_s \to \mathbf{x}_k}. \tag{6}$$

BP approximates the marginals for each variable by the product of incoming messages,

$$b_{\mathcal{G}}\left(\boldsymbol{x}_k\right) = \prod_{s \in \mathscr{N}_k} m_{f_s \to \mathbf{x}_k} \approx \int p(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}_{\setminus k}. \tag{7}$$

Proof: See Yedidia et al. (2000) for proof. □

While theoretical analysis of BP's approximation is complex (Yedidia et al., 2005; Weiss & Freeman, 2001), it delivers state-of-the-art results in important applications (Davison & Ortiz, 2019).

2.2.1 The Posterior Graph

So far, we ~~'ve~~ have discussed the prior graph $\mathcal{G}$ associated with the generative model. However, in practice, we ~~'re~~ are more interested in the evidence-conditional posterior graph $\mathcal{G}^*$, which incorporates the conditioning on $\mathbf{x}_{\mathscr{E}} = \boldsymbol{x}^*_{\mathscr{E}}$ as described in Equation 2.

To construct $\mathcal{G}^*$, we modify the factors f_j for all $j \in \mathscr{N}_{\mathscr{E}}$ in the prior graph. Specifically, each affected factor f_j is replaced with a conditioned factor f_j^* defined as:

$$f_j(\boldsymbol{x}_j) \leftarrow f_j^*(\boldsymbol{x}_{\mathscr{N}_j \setminus \mathscr{E}}) \coloneqq p(\boldsymbol{x}_{\mathscr{N}_j \setminus \mathscr{E}} | \mathbf{x}_{\mathscr{E}} = \boldsymbol{x}^*_{\mathscr{E}}). \tag{8}$$

We delete variables that are ~~obseved~~ observed and their associated edges. The target marginal distribution we aim to compute is then

$$p\left(\boldsymbol{x}_{\mathscr{Q}} \mid \mathbf{x}_{\mathscr{E}} = \boldsymbol{x}^*_{\mathscr{E}}\right) \approx b_{\mathcal{G}^*}\left(\boldsymbol{x}_{\mathscr{A}}\right),$$

where $b_{\mathcal{G}^*}(\boldsymbol{x}_{\mathscr{A}})$ is the belief over the ancestral variables in the posterior graph $\mathcal{G}^*$. For a visual representation, refer to Figure 11 in the Appendix.

2.2.2 Necessary operations for Belief Propagation

To compute the target marginal $b_{\mathcal{G}^*}(\boldsymbol{x}_{\mathscr{A}})$, we perform conditioning, marginalization, and multiplication operations on factor and variable densities:

Definition 1. *Consider a concatenated state space $\boldsymbol{x}^\top = \begin{bmatrix}\boldsymbol{x}_k^\top & \boldsymbol{x}_\ell^\top\end{bmatrix}$ at a node in a factor graph, with a parametric density $f(\boldsymbol{x};\boldsymbol{\theta})$. The following operations are sufficient for finding marginals by BP in an observation-conditional factor graph (Eq. 8):*

Conditioning: $$f\left(\boldsymbol{x};\boldsymbol{\theta}\right), \boldsymbol{x}_k^* \mapsto f^*\left(\boldsymbol{x}_\ell;\boldsymbol{\theta}_\ell^*\right) \coloneqq f\left(\boldsymbol{x}_\ell \mid \mathbf{x}_k = \boldsymbol{x}_k^*\right); \tag{9}$$

Marginalisation: $$f\left(\boldsymbol{x};\boldsymbol{\theta}\right) \mapsto f\left(\boldsymbol{x}_k;\boldsymbol{\theta}_k\right) \coloneqq \int f\left(\boldsymbol{x};\boldsymbol{\theta}\right) \mathrm{d}\boldsymbol{x}_\ell; \tag{10}$$

Multiplication: $$f\left(\boldsymbol{x};\boldsymbol{\theta}\right), f\left(\boldsymbol{x}_k;\boldsymbol{\theta}_k\right) \mapsto f\left(\boldsymbol{x};\boldsymbol{\theta}'\right) \coloneqq f\left(\boldsymbol{x};\boldsymbol{\theta}\right) f\left(\boldsymbol{x}_k;\boldsymbol{\theta}_k\right). \tag{11}$$

2.2.3 Gaussian Distributions in Belief Propagation

When the relationships between variables in the model are linear with additive Gaussian noise, and the ancestral distributions are Gaussian, the density over each factor is also Gaussian. In such cases, all the necessary operations outlined in Definition 1 have closed-form solutions (Bickson, 2009; Yedidia et al., 2000). This property is leveraged in classic Gaussian Belief Propagation (GaBP) (Dellaert & Kaess, 2017), which we briefly review here.

We represent Gaussian densities in two forms:

Definition 2 (Gaussian Density Forms)**.**

Moments Form: $$\phi_M(\boldsymbol{x}; \boldsymbol{m}, \mathrm{K}) = |2\pi\mathrm{K}|^{-1/2} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{m})^\top \mathrm{K}^{-1}(\boldsymbol{x}-\boldsymbol{m})\right), \tag{12}$$

Canonical Form: $$\phi_C(\boldsymbol{x}; \boldsymbol{n}, \mathrm{P}) = \left|\frac{\mathrm{P}}{2\pi}\right|^{1/2} \exp\left(-\frac{1}{2}\boldsymbol{x}^\top \mathrm{P}\boldsymbol{x} + \boldsymbol{n}^\top \boldsymbol{x} - \frac{1}{2}\boldsymbol{n}^\top \mathrm{P}^{-1}\boldsymbol{n}\right). \tag{13}$$

Here, $\boldsymbol{m}$ and K represent the mean and covariance, respectively, while $\mathrm{P} = \mathrm{K}^{-1}$ and $\boldsymbol{n} = \mathrm{P}\boldsymbol{m}$ denote the precision and information vectors, assuming the necessary inverses exist. Further details can be found in Appendix C.3.

Among the operations defined earlier, **multiplication** is the most challenging. In the canonical form, multiplication is straightforward:

$$\phi_C(\boldsymbol{x}; \boldsymbol{n}, \mathrm{P})\phi_C(\boldsymbol{x}_k; \boldsymbol{n}', \mathrm{P}') \propto \phi_C\left(\boldsymbol{x}; \boldsymbol{n}+\boldsymbol{n}', \mathrm{P}+\mathrm{P}'\right). \tag{14}$$

When factor potentials arise from a nonlinear simulator $\mathcal{P}$, the joint covariance is no longer Gaussian. In such cases, standard GaBP practices (e.g., Eustice et al., 2006; Ranganathan et al., 2007; Dellaert & Kaess, 2017) employ the δ-method (Dorfman, 1938) to find approximating Gaussian densities. The δ-method, or *propagation of error*, (see Appendix C.4) estimates the covariance of $\mathbf{g}(\hat{\boldsymbol{\theta}})$ with a first-order Taylor expansion

$$\mathrm{Cov}(\mathbf{g}(\hat{\boldsymbol{\theta}})) \approx \mathbf{J}_g(\boldsymbol{\theta})\,\mathrm{Cov}(\hat{\boldsymbol{\theta}})\,\mathbf{J}_g(\boldsymbol{\theta})^\top,$$

where $\mathbf{J}_g$ is the Jacobian of $\mathbf{g}$ at $\boldsymbol{\theta}$. ~~.~~

The accuracy of the δ-method is challenging to analyze for nonlinear $\mathcal{P}$. Additionally, GaBP scales unfavorably with the dimensionality of nodes, incurring a memory cost of $\mathcal{O}(D^2)$ and a time cost of $\mathcal{O}(D^3)$ whenever a $D \times D$ covariance matrix is inverted.

2.3 Ensemble Kalman Filtering

The Ensemble Kalman Filter (EnKF) mitigates the high computational costs associated with large D by representing prior distributions through ensembles of samples.

An ensemble is a matrix of N samples $\mathrm{X} = [\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}]$, where $\boldsymbol{x}^{(n)} \sim \phi_M(\boldsymbol{m}, \mathrm{K})$. We define the ensemble mean $\overline{\mathrm{X}} = \mathrm{XA}$ and deviation $\breve{\mathrm{X}} = \mathrm{X} - \overline{\mathrm{X}}\mathrm{B}$, where $\mathrm{A} = \frac{1}{N}\mathbf{1}$ and $\mathrm{B} = \mathbf{1}^\top$.

By overloading the mean and variance operations to describe the empirical moments of ensembles, we define:

$$\widehat{\mathbb{E}}\mathrm{X} = \overline{\mathrm{X}}, \qquad \widehat{\mathrm{Var}}_{\mathrm{V}}\mathrm{X} = \tfrac{1}{N-1}\breve{\mathrm{X}}\breve{\mathrm{X}}^\top + \mathrm{V}, \qquad \widehat{\mathrm{Cov}}(\mathrm{X}, \mathrm{Y}) = \tfrac{1}{N-1}\breve{\mathrm{X}}\breve{\mathrm{Y}}^\top \tag{15}$$

Here, V is a diagonal matrix known as the *nugget* term, typically set to $\sigma^2\mathrm{I}$. Setting $\sigma > 0$ is useful for numerical stability and to encode model uncertainty. These diagonal terms are usually treated as algorithm hyperparameters and also appear in GaBP.

The statistics of ensemble X define an implied Gaussian density:

$$\boldsymbol{x} \sim \phi_M(\boldsymbol{x}; \overline{\mathrm{X}}, \widehat{\mathrm{Var}}_{\mathrm{V}}[\mathrm{X}]).$$

Prior model ensembles are sampled via ancestral sampling using the generative model described in Equation 28.

Two of the Belief Propagation (BP) operations from Definition 1 are applicable to the EnKF.

Proposition 2. *Partition* $\mathbf{x}^\top = \begin{bmatrix}\mathbf{x}_k^\top & \mathbf{x}_\ell^\top\end{bmatrix}$ *such that* $\mathrm{X}^\top = \begin{bmatrix}\mathrm{X}_k^\top & \mathrm{X}_\ell^\top\end{bmatrix}$. *Assume the ensemble* X *follows the Gaussian distribution:*

$$\mathrm{X} \sim \phi_M\left(\begin{bmatrix}\boldsymbol{x}_k\\ \boldsymbol{x}_\ell\end{bmatrix}; \begin{bmatrix}\overline{\mathrm{X}}_k\\ \overline{\mathrm{X}}_\ell\end{bmatrix}, \begin{bmatrix}\widehat{\mathrm{Var}}_{\mathrm{V}}\mathrm{X}_k & \widehat{\mathrm{Cov}}(\mathrm{X}_\ell, \mathrm{X}_k)\\ \widehat{\mathrm{Cov}}(\mathrm{X}_k, \mathrm{X}_\ell) & \widehat{\mathrm{Var}}_{\mathrm{V}}(\mathrm{X}_\ell)\end{bmatrix}\right). \tag{16}$$

In ensemble form, conditioning (Eq. 9) is performed as:

$$\mathrm{X}, \boldsymbol{x}_k^* \mapsto \mathrm{X}_\ell + \widehat{\mathrm{Cov}}(\mathrm{X}_\ell, \mathrm{X}_k)\widehat{\mathrm{Var}}_{\mathrm{V}}^{-1}(\mathrm{X}_k)(\boldsymbol{x}_k^*\mathrm{B} - \mathrm{X}_k) \tag{17}$$

The computational cost of solving Equation 17 is $\mathcal{O}(N^3 + N^2 D_{\mathbf{x}_k})$. *Marginalization (Eq. 10) is simply truncation, i.e.,* $\mathrm{X} \mapsto \mathrm{X}_k$.

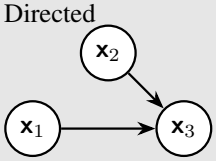
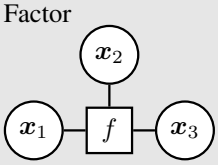
Proof: See Appendix D. □

While the EnKF reduces computational costs and potentially improves approximation accuracy over GaBP for nonlinear relationships, it does not generalise to other model structures without the multiplication operation (Eq. 11) needed for BP.

In the following sections, we extend the EnKF-like approach to the BP setting by defining the necessary operations to handle general graphical models.

Table 1: Relations in Gaussian Ensemble Belief Propagation

Empirical statistics →

	Generative	**Density-based**
Operations	• Sample • Condition	• Propagate
Graph type	Directed 	Factor 
Decomposition	$\mathbf{x}_3 = \mathcal{P}(\mathbf{x}_1, \mathbf{x}_2)$	$f(x_1, x_2, x_3)$
Node Parameters	Empirical moments $\boldsymbol{m}, \mathrm{K}$	Canonical parameters $\boldsymbol{n}, \mathrm{P}$

← Ensemble recovery

3 Gaussian Ensemble Belief Propagation

GEnBP proceeds by: (1) sampling from the generative prior to obtain an ensemble; (2) converting ensemble statistics to canonical form; (3) performing BP using low-rank representations; and (4) recovering ensemble samples matching the updated beliefs. This is diagrammed in Figure 1. Throughout it exploits efficient matrix representations and operations to handle high-dimensional problems. Although this is conceptually simple, there are delicate details in the implementation; we discuss these in overview in this section, and in depth in Appendix H.

Throughout this section, we consider an approximation $\phi_M(\hat{\boldsymbol{m}}, \hat{\mathrm{K}})$ to a target Gaussian $\phi_M(\boldsymbol{m}, \mathrm{K})$ to be optimal if $\hat{\boldsymbol{m}} = \boldsymbol{m}$ and the Frobenius norm $\|\hat{\mathrm{K}} - \mathrm{K}\|_F$ of the covariance difference is minimized. This corresponds to minimizing the Maximum Mean Discrepancy (MMD) with a second-order polynomial kernel between these distributions (Sriperumbudur et al., 2010, Example 3).

3.1 Rank-efficient Gaussian Parameterisations

A key component of GEnBP is the use of *Diagonal Matrix with Low-rank perturbation* (DLR) representations of Gaussian distributions. These representations are derived from ensemble samples and

enable efficient computation and transformation between different parameterizations (e.g., moments and canonical forms) without explicitly forming high-dimensional covariance matrices.

We use Diagonal Matrix with Low-rank perturbation (DLR) representations, where a symmetric positive-definite matrix $\mathrm{K} \in \mathbb{R}^{D \times D}$ is expressed as $\mathrm{K} = \mathrm{V} + s\mathrm{L}\mathrm{L}^\top$, with $\mathrm{L} \in \mathbb{R}^{D \times N}$, V diagonal and the sign $s \in \{-1, +1\}$. We exploit many computationally expedient properties of such matrices (see in Appendix F~~.)~~). Here we outline the ones that are not standard in Belief Propagation.

In the context of the EnKF, the empirical covariance matrix computed from ensemble samples is naturally DLR (Eq. 15). Throughout this section N is the size fo the ensemble or ~~equiavlently~~equivalently, the rank of the DLR factors, and D is the dimension of the factor.

To initialize the BP stage of GEnBP, we set the distribution for each factor f_j^* to the empirical distribution of the ensemble at that factor, X_j:

$$f_j^* \sim \phi_M(\boldsymbol{x}; \overline{\mathrm{X}}_j, \widehat{\mathrm{Var}}_{\gamma^2 \mathrm{I}} \mathrm{X}_j). \tag{18}$$

We note that of the many possible ways that DLR approximations might be used in a BP, the one arising from the ensemble representation is most favourable. ~~See for a comparison with a naive attempt to do without the~~ Appendix I compares the computational cost of the GEnBP with an attempt to exploit DLR factorisations without using an ensemble.

Proposition 3. *Suppose* $\mathsf{x} \sim \phi_M(\boldsymbol{x}; \boldsymbol{m}, \mathrm{K})$ *has a DLR covariance* $\mathrm{K} = \mathrm{L}\mathrm{L}^\top + \mathrm{V}$*, with* $\mathrm{L} \in \mathbb{R}^{D \times N}$ *and* V *diagonal. Then, we can find the canonical form parameters* $\boldsymbol{n}$ *and* P *efficiently:*

$$\boldsymbol{n} = \mathrm{P}\boldsymbol{m}, \quad \mathrm{P} = \mathrm{K}^{-1} = \mathrm{U} - \mathrm{R}\mathrm{R}^\top,$$

where U *is diagonal and* $\mathrm{R} \in \mathbb{R}^{D \times N}$*. Both the conversion to canonical form and the recovery of moments from the canonical form can be performed with time complexity* $\mathcal{O}(N^3 + N^2 D)$*.*

Proof: Standard application of the Woodbury identity. See Appendix F.3. □

3.2 Belief Propagation with DLR Representations

In standard GaBP, multiplying Gaussian densities (as required in message updates) involves operations with full covariance or precision matrices, which is computationally expensive in high dimensions. However, in GEnBP, we exploit the DLR structure to perform these operations efficiently.

Let K and K′ be two DLR matrices,

$$\mathrm{K} = \mathrm{V} + s\mathrm{L}\mathrm{L}^\top, \quad \mathrm{K}' = \mathrm{V}' + s\mathrm{L}'\mathrm{L}'^\top.$$

Then, their sum is also DLR,

$$\mathrm{K} + \mathrm{K}' = (\mathrm{V} + \mathrm{V}') + s\,[\mathrm{L} \quad \mathrm{L}']\,[\mathrm{L} \quad \mathrm{L}']^\top. \tag{19}$$

Thus we can efficiently perform density multiplication of Gaussian (as required in BP message updates) by adding DLR precision matrices as per Equation 14, maintaining the DLR moment representation. Other useful operations such as vector products, rank reduction and marginalisation are also efficient – see details in Appendix F.

The most computationally intensive step in BP is the factor-to-variable message (Eq. 5). Suppose we have a factor node f_j connected to K variable nodes, and we wish to compute the message from f_j to variable node x_k. We find the product density of the incoming messages (in canonical form) from the other $K - 1$ variable nodes using the additive property of DLR precision matrices, and repeated application of Equation 14. This can be performed with time complexity $\mathcal{O}(DN^2K^2 + N^3K^3)$ by exploiting the DLR structure – see Algorithm 3 in Appendix H.

3.3 Ensemble Conformation

After propagating messages, we need to update the ensemble samples to reflect the new beliefs at the query nodes. This step, called *ensemble conformation*, finds an affine transformation of the prior ensemble that matches the posterior mean and covariance as closely as possible.

We seek an affine transformation $T_{\boldsymbol{\mu},\mathrm{T}} : \mathrm{X} \mapsto \boldsymbol{\mu}\mathrm{B} + \breve{\mathrm{X}}\mathrm{T}$, where $\boldsymbol{\mu} \in \mathbb{R}^D$ is set to the posterior mean $\boldsymbol{m}$, and $\mathrm{T} \in \mathbb{R}^{N\times N}$ is chosen to minimize the difference between the transformed ensemble covariance and the DLR target covariance, $\mathrm{L}\mathrm{L}^\top + \mathrm{V}$,

$$\mathrm{T} := \arg\min_{\mathrm{T}^*} \left\| \widehat{\mathrm{Var}}_{\eta^2\mathrm{I}}(\boldsymbol{\mu}\mathrm{B} + \breve{\mathrm{X}}\mathrm{T}^*) - (\mathrm{L}\mathrm{L}^\top + \mathrm{V}) \right\|_F^2 . \tag{20}$$

Noting that T is identifiable only up to a unitary transform, we use the following result:

Proposition 4. *Any symmetric positive semi-definite matrix* $\mathrm{G} = \mathrm{T}\mathrm{T}^\top$ *satisfying*

$$\breve{\mathrm{X}}^\top \breve{\mathrm{X}}\, \mathrm{G}\, \breve{\mathrm{X}}^\top \breve{\mathrm{X}} = (N-1)\left(\breve{\mathrm{X}}^\top \mathrm{L}(\breve{\mathrm{X}}^\top \mathrm{L})^\top - \breve{\mathrm{X}}^\top(\mathrm{V} - \eta^2\mathrm{I})\breve{\mathrm{X}}\right)$$

defines an affine transformation that minimizes the Frobenius loss in Equation 20. Such a G *can be found with memory complexity* $\mathcal{O}(M^2 + N^2)$ *and time complexity* $\mathcal{O}(N^3 + DN^2 + DM^2)$.

Proof: See Appendix H.3. □

For a variable with K neighbors, in the worst case $M = KN$, ~~when~~ and the cost becomes $\mathcal{O}(N^3 + DN^2K^2)$. The cost of finding the target covariance from the belief precision is $\mathcal{O}(M^3 + DM^2) = \mathcal{O}(K^3N^3 + DK^2N^2)$, leading to a total cost for ensemble recovery of $\mathcal{O}(K^3N^3 + DN^2K^2)$.

3.4 Computational Complexity

The total computational cost of GEnBP depends on the graph structure and the choice of ensemble size N. In many practical applications ~~, ($N \ll D$),~~ it holds that $N \ll D$, resulting in significant computational savings compared to GaBP ~~,~~ which scales poorly with ~~D.~~

D. Table 2 summarizes the computational costs of GaBP and GEnBP for various operations. Notably, GEnBP avoids the $\mathcal{O}(D^3)$ costs associated with full covariance matrix operations in GaBP, making it ~~more suitable~~ better suited for high-dimensional problems.

Table 2: Computational complexities for Gaussian Belief Propagation (GaBP) and Gaussian Ensemble Belief Propagation (GEnBP), where D is the node dimension, K is the node degree, and N is the ensemble size. All variables are assumed to have the same dimension for simplicity. Time complexity measures floating-point multiplications.

Operation	GaBP	GEnBP
Time Complexity		
Simulation	$\mathcal{O}(1)$	$\mathcal{O}(N)$
Error propagation	$\mathcal{O}(D^3)$	—
Jacobian calculation	$\mathcal{O}(D)$	—
Covariance matrix	$\mathcal{O}(D^2)$	$\mathcal{O}(ND)$
Factor-to-node message	$\mathcal{O}(D^3)$	$\mathcal{O}(K^3N^3 + DN^2K^2)$
Node-to-factor message	$\mathcal{O}(D^2)$	$\mathcal{O}(1)$
Ensemble recovery	—	$\mathcal{O}(K^3N^3 + DN^2K^2)$
Canonical-Moments conversion	$\mathcal{O}(D^3)$	$\mathcal{O}(N^3 + N^2D)$
Space Complexity		
Covariance matrix	$\mathcal{O}(D^2)$	$\mathcal{O}(ND)$
Precision matrix	$\mathcal{O}(D^2)$	$\mathcal{O}(NDK)$

~~While GEnBP scales favorably~~ GEnBP operations scale favourably with D, ~~but~~ at worst $\mathcal{O}(D)$, compared to the worst $\mathcal{O}(D^3)$ for GaBP. They scale unfavourably with respect to the node degree ~~K, scaling as~~ , at worst $\mathcal{O}(K^3)$ ~~for some operations. However, in~~ , compared to $\mathcal{O}(1)$. In practice, large factor nodes can be decomposed into chains of smaller nodes using techniques like *Forney factorization* (Forney, 2001), which reduces K without altering the model's marginal distributions (de Vries & Friston, 2017). ~~This helps manage computational costs even in graphs with high node degrees. In our experiments, despite~~

We note however that should the K scaling prove prohibitive, we are still only marginally worse off than in GaBP, since we can always convert to a full covariance matrix and use the GaBP update

(Appendix I). We have not found this necessary in our experiments. Despite the potential increase in the number of messages due to decomposition, the overall computational cost is reduced because the messages themselves become less expensive to compute.

4 EXPERIMENTS

~~We~~ In this section, we compare GEnBP against alternative ~~methods~~belief propagation method, GaBP and for reference, a global Laplace approximation (Mackay, 1992)~~, in two~~ . We use synthetic benchmarks designed to compare the methods' performance in high-dimensional, nonlinear dynamical systems. In both, the graph structure is a ~~random~~ randomised system identification task (Appendix A.1), where a static parameter of interest influences a noisily observed, nonlinear dynamical system. Although GEnBP is applicable beyond the system identification setting, this benchmark is chosen for its simplicity popularity. More sophisticated graphical model problems are outlined in Appendix C, and in Appendix B.3 we demonstrate the utility of GEnBP applied to a different graphical structure, few shot domain adaptation of a system dynamics emulator on unobserved states.

For $t = 1, 2, \cdots, T$ we define

$$\mathbf{x}_t = \mathcal{P}_{\mathbf{x}}(\mathbf{x}_{t\text{-}1}, \mathbf{q}), \qquad \mathbf{y}_t = \mathcal{P}_{\mathbf{y}}(\mathbf{x}_t) \tag{21}$$

where evidence is $\mathscr{E} = \{\mathbf{y}_t = \boldsymbol{y}_t^*\}_{1 \le t \le T}$ and the query is $\mathscr{Q} = \mathbf{q}$.[3]

The GaBP implementation is modified from Ortiz et al. (2021) to allow arbitrary prior covariance. Hyperparameters are not directly comparable between the methods. We ameliorate this by choosing favourable values for each algorithm. We measure performance by mean squared error (MSE) of the posterior mean estimate $\mathbb{E}b_{\mathcal{G}*}(\mathbf{q}) - q_0$ and the log-likelihood $\log b_{\mathcal{G}*}(q_0; \mathbf{q})$ at ground truth q_0. The estimated posterior covariance is not necessarily positive definite in gradient-based methods such as Laplace or GaBP, and thus the posterior log-likelihood may fail to be defined; we do not plot it in that case. Additional experiments and details are in Appendix B.

4.1 1D TRANSPORT MODEL

The transport problem (Appendix A.2) is a simple ~~, 1d~~ nonlinear dynamical system defined over a 1-dimensional spatial extend, chosen for ease of visualisation~~()~~. States are subject to both transport and diffusion, where the transport term introduces nonlinearity. Observations are subsampled state vectors perturbed by additive Gaussian noise. Figure 1 shows exemplary samples from the prior and posterior distributions ~~. GEnBP estimates are substantially more accurate than GaBP , in terms of both posterior likelihood and MSE, as can be seen in the tighter clustering of the posterior samples about ground-truth. In this relatively low-dimensional problem, the Laplace approximation is also available for comparison; We see that the Laplace approximationis superior to both GaBP and GEnBP in terms of execution time. Its MSE performance is intermediate in quality, but its posterior likelihood, while similar to GaBP, is even less stable, and both are inferior to GEnBP. Influence of dimension $D_{\mathscr{Q}}$ in the transport example. Error bars are empirical 90% intervals from $n = 40$ runs.~~ induced by GaBP and GEnBP methods, and demonstrate a substantially improved ability for GEnBP to recover the posterior in this setting. Analysis of this contrived exampled is continued in Appendix A.2.1 where it is compared also against a global Laplace approximation.

4.2 2D FLUID DYNAMICS MODEL

In the computational fluid dynamics (CFD) problem (Appendix A.3), states are governed by discretised Navier-Stokes equations over a 2d spatial domain. The parameter of interest is a static, latent forcing field $\mathbf{q}$. In this setting, the GaBP algorithm is unable to complete the problem for $D_{\mathscr{Q}} > 1024$, and the Laplace approximation is not available, since estimated posterior covariance is far from positive definite. In our 2d incompressible equations the state field and forcing fields are scalar fields over a $d \times d$ spatial domain; we stack them into vectors so that $D_{\mathscr{Q}} = D_{\mathbf{x}_t} = d^2$. ~~Influence of dimension $D_{\mathscr{Q}}$. Error bars are empirical 50% intervals from $n = 10$ runs. In grey-shaded regions, GaBP ran out of memory.~~

[3] If $\mathbf{q}$ were known, estimating $\{\mathbf{x}_t\}_t$ from $\{\mathbf{y}_t\}_t$ would be a filtering problem, soluble by EnKF.

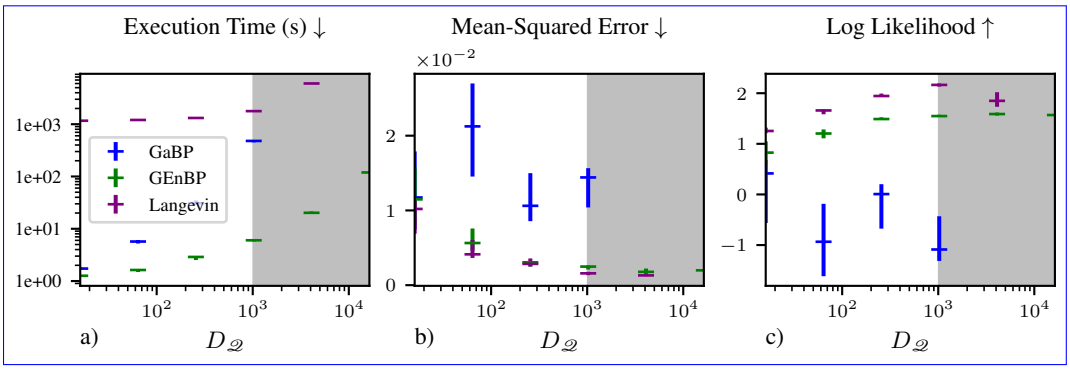


Figure 2: Influence of dimension $D_{\mathcal{Q}}$ in the 2D fluid dynamics model. Error bars are empirical 50% intervals from $n = 10$ runs. In grey-shaded regions, GaBP ran out of memory.

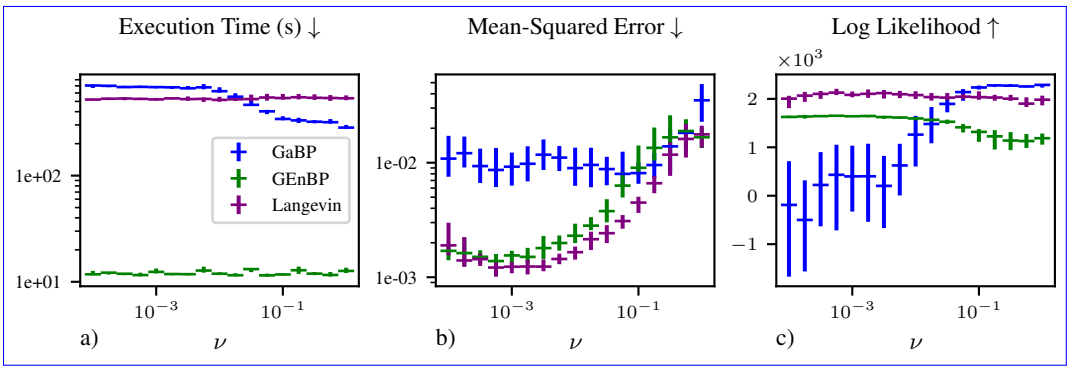


Figure 3: Influence of viscosity ν on a 32×32 2D fluid dynamics model. Error bars show empirical 50% intervals from $n = 40$ simulations.

To provide a comparison to a classic sampler which does not exploit belief propagation, we implement a Langevin Monte Carlo (LMC) (Roberts & Tweedie, 1996) algorithm as a baseline. We run this until it produces comparable MSE to GEnBP (~~5000~~ 2000 iterations after a 1000 iteration burn-in).

~~Results~~ Results in increasing dimension are shown in Figure 2. We note that the Langevin Monte Carlo algorithm, attains the best posterior ~~likelihood~~RMSE, but at a cost of approximately 10^3 times the compute cost of GEnBP. We observe the GEnBP attains superior mean-squared error and posterior likelihood to GaBP, while scaling to a far higher $D_{\mathcal{Q}}$. Note the GaBP experiments are truncated because experiments with $D_{\mathcal{Q}} > 1024$ failed to complete due to resource exhaustion.

We caution against interpreting the absolute run time of the algorithm too literally. Notably, the high-dimensional Jacobian calculation in the GaBP algorithm is not parallelised effectively by the Pytorch library, which penalises that algorithm by a constant multiplicative factor. The asymptotic $\mathcal{O}(D)$ scaling rate of such Jacobian calculations (Margossian, 2019) will ensure eventual dominance of GEnBP, however.

Figure 3 shows the influence of the viscosity parameter ν as it varies from laminar to turbulent flow regimes, producing diverse nonlinear behaviours (See Figure 6). GEnBP ~~still~~ strictly dominates GaBP in speed. It generally dominates in MSE, although there are ranges around $\eta \approx 1$ where the performance is indistinguishable or slightly worse. Log Regarding posterior log-likelihood of the ground-truth, GEnBP is superior in the low-ν (turbulent) regime, whereas GaBP is dominant in the high-ν (laminar) regime. Langevin Monte Carlo is not competitive in this the posterior log likelihood, although this could likely be improved by adjusting the algorithm's hyperparameters.

5 CONCLUSIONS

Gaussian Ensemble Belief Propagation (GEnBP) advances feasible inference in probabilistic graphical models by combining the strengths of the Ensemble Kalman Filter (EnKF) and Gaussian Belief Propagation (GaBP). It scales to higher factor dimensions than GaBP and achieves superior accuracy in complex, high-dimensional problems. By employing ensemble approximations, GEnBP accommodates ~~larger and~~ the larger, more intricate factors common in real-world applications. Unlike EnKF, GEnBP handles complex dependencies without requiring gradients and often surpasses the performance of GaBP in both accuracy and speed.

While GEnBP introduces additional tuning parameters and requires more model executions during sampling, these costs are often offset by avoiding high-dimensional Jacobian calculations needed in GaBP. Its effectiveness is best in systems where the dynamics are well-approximated in a low-dimensional subspace (DLR). Like GaBP and EnKF, GEnBP relies on Gaussian approximations and is constrained to unimodal distributions; convergence analysis remains an open question. Comparing overall computational cost of the algorithms is complicated and depends upon the graph structure. Our empirical results demonstrate the existence of problems where GEnBP is much more efficient than GaBP, but we have not characterised the full range of problems where this is the case.

Despite these limitations, GEnBP's scalability, flexibility, and ease of use make it a promising tool for a broad range of applications, notably in geospatial predictions and high-dimensional data assimilation. Future work includes integrating ~~improvements from~~ the diverse tweaks used in practice in applied GaBP and EnKF ~~literature, such as graph pruning for computational efficiency and robust distributions to handle outliers.~~ methods. We suspect that the same tricks that are used in EnKF would be useful in GEnBP, and moreover many of the same tricks that are used in GaBP would be useful in GEnBP. Ensemble or DLR equivalents of covariance localisation, covariance inflation and adaptive ensemble selection from EnKF seem relatively simple to transfer to GEnBP, and we suspect that further hybridizations would be useful in practice. Similarly, from GaBP, the use of message damping, incremental updating, robust covariance scaling and truncation of graphs would likely extend the range of applicability of the method by providing increased stability and convergence at lower cost if adequately tuned. Exploring connections with other message-passing algorithms, like particle-based or Stein variational methods, may further enhance GEnBP's capabilities.

REFERENCES

Elias Bareinboim and Judea Pearl. A General Algorithm for Deciding Transportability of Experimental Results. *Journal of Causal Inference*, 1(1):107–134, May 2013. doi: 10.1515/jci-2012-0004.

Danny Bickson. *Gaussian Belief Propagation: Theory and Application*. PhD thesis, July 2009.

Yuming Chen, Daniel Sanz-Alonso, and Rebecca Willett. Reduced-order autodifferentiable ensemble Kalman filters. *Inverse Problems*, 39(12):124001, October 2023. doi: 10.1088/1361-6420/acff14.

Andrew J. Davison and Joseph Ortiz. FutureMapping 2: Gaussian Belief Propagation for Spatial AI. *arXiv:1910.14139 [cs]*, October 2019.

Bert de Vries and Karl J. Friston. A Factor Graph Description of Deep Temporal Active Inference. *Frontiers in Computational Neuroscience*, 11, 2017.

Frank Dellaert and Michael Kaess. Factor Graphs for Robot Perception. *Foundations and Trends® in Robotics*, 6(1-2):1–139, August 2017. doi: 10.1561/2300000043.

R. A. Dorfman. A note on the δ-method for finding variance formulae. *Biometric Bulletin*, 1938.

Arnaud Doucet. A Note on Efficient Conditional Simulation of Gaussian Distributions. Technical report, University of British Columbia, 2010.

Ryan M. Eustice, Hanumant Singh, and John J. Leonard. Exactly Sparse Delayed-State Filters for View-Based SLAM. *IEEE Transactions on Robotics*, 22(6):1100–1114, December 2006. doi: 10.1109/TRO.2006.886264.

Geir Evensen. The Ensemble Kalman Filter: Theoretical formulation and practical implementation. *Ocean Dynamics*, 53(4):343–367, November 2003. doi: 10.1007/s10236-003-0036-9.

Geir Evensen. *Data Assimilation - The Ensemble Kalman Filter*. Springer, Berlin; Heidelberg, 2009. ISBN 978-3-642-03711-5 3-642-03711-9. doi: 10.1007/978-3-642-03711-5.

Paul Fearnhead and Hans R. Künsch. Particle Filters and Data Assimilation. *Annual Review of Statistics and Its Application*, 5(1):421–449, 2018. doi: 10.1146/annurev-statistics-031017-100232.

Joel H. Ferziger, Milovan Perić, and Robert L. Street. *Computational Methods for Fluid Dynamics*. Springer, Cham, 4th ed. 2020 edition edition, August 2019. ISBN 978-3-319-99691-2.

G.D. Forney. Codes on graphs: Normal realizations. *IEEE Transactions on Information Theory*, 47 (2):520–548, February 2001. doi: 10.1109/18.910573.

Brendan J. Frey, Frank R. Kschischang, Hans-Andrea Loeliger, and Niclas Wiberg. Factor graphs and algorithms. In *Proceedings of the Annual Allerton Conference on Communication Control and Computing*, volume 35, pp. 666–680. Citeseer, 1997.

R. Furrer and T. Bengtsson. Estimation of high-dimensional prior and posterior covariance matrices in Kalman filter variants. *Journal of Multivariate Analysis*, 98(2):227–255, February 2007. doi: 10.1016/j.jmva.2006.08.003.

Xiang Gao, Meera Sitharam, and Adrian E. Roitberg. Bounds on the Jensen Gap, and Implications for Mean-Concentrated Distributions, August 2020.

Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions, December 2010.

P. L. Houtekamer and Fuqing Zhang. Review of the Ensemble Kalman Filter for Atmospheric Data Assimilation. *Monthly Weather Review*, 144(12):4489–4532, December 2016. doi: 10.1175/MWR-D-15-0440.1.

D. T. B. Kelly, K. J. H. Law, and A. M. Stuart. Well-posedness and accuracy of the ensemble Kalman filter in discrete and continuous time. *Nonlinearity*, 27(10):2579, September 2014. doi: 10.1088/0951-7715/27/10/2579.

Daphne Koller and Nir Friedman. *Probabilistic Graphical Models : Principles and Techniques*. MIT Press, Cambridge, MA, 2009. ISBN 0-262-01319-3.

Frank R Kschischang, Brendan J Frey, and Hans-Andrea Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, February 2001. doi: 10.1109/18.910572.

Soeren Laue, Matthias Mitterreiter, and Joachim Giesen. Computing Higher Order Derivatives of Matrix and Tensor Expressions. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 2750–2759. Curran Associates, Inc., 2018.

Sören Laue, Matthias Mitterreiter, and Joachim Giesen. A simple and efficient tensor calculus. In *AAAI Conference on Artificial Intelligence, (AAAI)*. 2020.

François Le Gland, Valérie Monbet, and Vu-Duc Tran. *Large Sample Asymptotics for the Ensemble Kalman Filter*. Report, INRIA, 2009.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier Neural Operator for Parametric Partial Differential Equations. *arXiv:2010.08895 [cs, math]*, October 2020.

Finn Lindgren, Håvard Rue, and Johan Lindström. An explicit link between Gaussian fields and Gaussian Markov random fields: The stochastic partial differential equation approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(4):423–498, September 2011. doi: 10.1111/j.1467-9868.2011.00777.x.

David J. C. Mackay. A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation*, 4(3):448–472, May 1992. doi: 10.1162/neco.1992.4.3.448.

Charles C. Margossian. A Review of automatic differentiation and its efficient implementation. *WIREs Data Mining and Knowledge Discovery*, 9(4):e1305, July 2019. doi: 10.1002/WIDM.1305.

Kevin P. Murphy, Yair Weiss, and Michael I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, UAI'99, pp. 467–475, San Francisco, CA, USA, July 1999. Morgan Kaufmann Publishers Inc. ISBN 978-1-55860-614-2.

Terence J. O'Kane, Paul A. Sandery, Vassili Kitsios, Pavel Sakov, Matthew A. Chamberlain, Dougal T. Squire, Mark A. Collier, Christopher C. Chapman, Russell Fiedler, Dylan Harries, Thomas S. Moore, Doug Richardson, James S. Risbey, Benjamin J. E. Schroeter, Serena Schroeter, Bernadette M. Sloyan, Carly Tozer, Ian G. Watterson, Amanda Black, Courtney Quinn, and Richard J. Matear. CAFE60v1: A 60-year large ensemble climate reanalysis. Part II: Evaluation. *Journal of Climate*, pp. 1–62, March 2021. doi: 10.1175/JCLI-D-20-0518.1.

Joseph Ortiz, Talfan Evans, and Andrew J. Davison. A visual introduction to Gaussian Belief Propagation. *arXiv:2107.02308 [cs]*, July 2021.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. The Morgan Kaufmann Series in Representation and Reasoning. Kaufmann, San Francisco, Calif, rev. 2. print., 12. [dr.] edition, 2008. ISBN 978-1-55860-479-7.

Judea Pearl, Madelyn Glymour, and Nicholas P. Jewell. *Causal Inference in Statistics: A Primer*. Wiley, March 2016. ISBN 978-1-119-18684-7.

Kaare Brandt Petersen and Michael Syskind Pedersen. The Matrix Cookbook. Technical report, 2012.

Ananth Ranganathan, Michael Kaess, and Frank Dellaert. Loopy SAM. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pp. 2191–2196, San Francisco, CA, USA, January 2007. Morgan Kaufmann Publishers Inc.

Gareth O. Roberts and Richard L. Tweedie. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341–363, December 1996.

Håvard Rue, Sara Martino, and Nicolas Chopin. Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(2):319–392, 2009. doi: 10.1111/j.1467-9868.2008.00700.x.

Claudia Schillings and Andrew M. Stuart. Analysis of the Ensemble Kalman Filter for Inverse Problems. *SIAM Journal on Numerical Analysis*, 55(3):1264–1290, January 2017. doi: 10.1137/16M105959X.

Bharath K. Sriperumbudur, Arthur Gretton, Kenji Fukumizu, Bernhard Schölkopf, and Gert R. G. Lanckriet. Hilbert Space Embeddings and Metrics on Probability Measures. *Journal of Machine Learning Research*, 11:1517–1561, April 2010.

Aki Vehtari, Andrew Gelman, Tuomas Sivula, Pasi Jylänki, Dustin Tran, Swupnil Sahai, Paul Blomstedt, John P. Cunningham, David Schiminovich, and Christian Robert. Expectation propagation as a way of life: A framework for Bayesian inference on partitioned data. *arXiv:1412.4869 [stat]*, November 2019.

Martin J. Wainwright and Michael I. Jordan. *Graphical Models, Exponential Families, and Variational Inference*, volume 1 of *Foundations and Trends® in Machine Learning*. Now Publishers, 2008. doi: 10.1561/2200000001.

M. P. Wand. Fast Approximate Inference for Arbitrarily Large Semiparametric Regression Models via Message Passing. *Journal of the American Statistical Association*, 112(517):137–168, January 2017. doi: 10.1080/01621459.2016.1197833.

Yair Weiss and William T. Freeman. Correctness of Belief Propagation in Gaussian Graphical Models of Arbitrary Topology. *Neural Computation*, 13(10):2173–2200, October 2001. doi: 10.1162/089976601750541769.

James T Wilson, Viacheslav Borovitskiy, Alexander Terenin, Peter Mostowsky, and Marc Peter Deisenroth. Pathwise Conditioning of Gaussian Processes. *Journal of Machine Learning Research*, 22(105):1–47, 2021.

Sewall Wright. The Method of Path Coefficients. *The Annals of Mathematical Statistics*, 5(3): 161–215, September 1934. doi: 10.1214/aoms/1177732676.

Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Generalized belief propagation. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS'00, pp. 668–674, Cambridge, MA, USA, January 2000. MIT Press.

J.S. Yedidia, W.T. Freeman, and Y. Weiss. Constructing Free-Energy Approximations and Generalized Belief Propagation Algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, July 2005. doi: 10.1109/TIT.2005.850085.

A BENCHMARK PROBLEMS

To demonstrate the utility of our method, we select example problems of system identification type, where a latent parameter of interest must be inferred from its effects upon the dynamics of a hidden Markov model. As such it is closely related to, but more challenging than, state filtering.

Throughout, unless otherwise specified, the hyper parameters of the algorithms are $\gamma^2 = 0.01$ and $\sigma^2 = 0.001$ for both GaBP and GEnBP. GEnBP has additional hyperparameters $\eta^2 = 0.1$ and $N = 64$. We cap the number of iterations of message-propagation descent steps at 150. We relinearise or re-simulate after 10 message-propagation steps.

A.1 SYSTEM IDENTIFICATION

In the system identification problem our goal is to estimate the time-invariant latent system parameter $\mathbf{q}$. This parameter influences all states in the Hidden Markov Model, where $\mathbf{x}_0$ represents the unobserved initial state, and subsequent states $\mathbf{x}_1, \mathbf{x}_2, \ldots$ evolve over time. Each state $\mathbf{x}_i$ is associated with an observed state $\mathbf{y}_i$, as depicted in Figure 4.

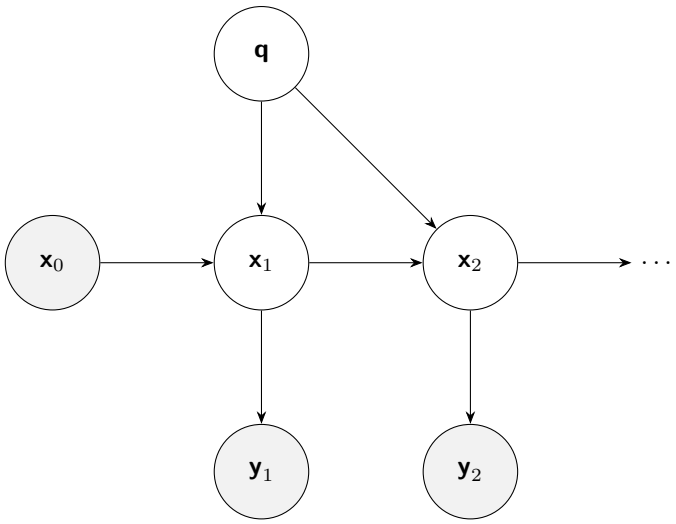


Figure 4: Generative model for the system identification problem, with latent system parameter $\mathbf{q}$, where all states depend on $\mathbf{q}$ and the previous state. Observed states are shaded.

A.2 ONE-DIMENSIONAL TRANSPORT PROBLEM

We consider a one-dimensional state-space model where the state at time t, $\mathbf{x}_t \in \mathbb{R}^d$, evolves according to

$$\mathcal{P}_t^x : \mathbf{x}_t \to \mathbf{x}_{t+1} \tag{22}$$

$$= S_k \left(C * (\gamma \mathbf{x}_t + (1-\gamma)\mathbf{q}) \right) + \boldsymbol{\epsilon}_t, \tag{23}$$

$$\mathcal{P}_t^y : \mathbf{x}_t \to \mathbf{y}_t \tag{24}$$

$$= D_\ell \left(\mathbf{x}_t \right) + \boldsymbol{\eta}_t, s \tag{25}$$

where

- $\mathbf{x}_t$ is the state vector at time t.
- $\mathbf{q} \in \mathbb{R}^d$ is a fixed parameter vector to be estimated.
- $\gamma \in [0, 1]$ is a decay parameter.
- $C * \mathbf{z}$ denotes the convolution of $\mathbf{z}$ with kernel C.
- S_k is a circular shift operator that shifts the vector by k positions to the right.
- $\boldsymbol{\epsilon}_t \sim \mathcal{N}(0, \tau^2 I_d)$ is process noise.
- $\mathbf{y}_t$ is the observation vector at time t.

- D_ℓ is a downsampling operator that selects every ℓ-th element of its input vector.
- $\boldsymbol{\eta}_t \sim \mathcal{N}(0, \sigma^2 I_m)$ is observation noise.

The parameter $\mathbf{q}$ is drawn from a prior distribution that generates smooth periodic functions. Specifically, each element of $\mathbf{q}$ is defined as

$$q_k = A \cdot e^{\kappa \cos\left(\frac{2\pi k}{d} - \mu\right)}, \quad k = 0, 1, \ldots, d-1, \tag{26}$$

where

- $\mu \sim \text{Uniform}[0, 2\pi]$ is a random phase.
- $\kappa \sim \chi^2(\sqrt{2\pi})$ controls the concentration (smoothness) of the function.
- A is a scaling constant.

Operator Definitions

- **Convolution** ($C * \mathbf{z}$): The convolution operator applies a blur kernel C to the vector $\mathbf{z}$, modeling diffusion.
- **Circular Shift** (S_k): The operator shifts the vector $\mathbf{z}$ by k positions to the right in a circular manner (elements shifted beyond the last position re-enter at the first position), modeling advection.
- **Downsampling** (D_ℓ): The operator selects every ℓ-th element from the input vector $\mathbf{z}$, reducing spatial resolution.

Intuitive Interpretation The state update (Eq. 23) models a transport process where the state $\mathbf{x}_t$ evolves through

1. *Decay towards a background field*: The term $\gamma \mathbf{x}_t + (1-\gamma)\mathbf{q}$ represents a weighted combination of the current state and the background field $\mathbf{q}$.
2. *Diffusion*: Convolution with kernel C introduces spatial smoothing, simulating diffusion.
3. *Advection*: The circular shift S_k models the transport of the field in a particular direction.
4. *Process Noise*: $\boldsymbol{\epsilon}_t$ accounts for uncertainties in the evolution.

The observation equation (Eq. 25) represents measurements of the state at reduced spatial resolution due to downsampling, with additive observation noise $\boldsymbol{\eta}_t$.

Inference Problem Our goal is to estimate the parameter $\mathbf{q}$ given the sequence of observations $\{\mathbf{y}_t\}$ and the initial state $\mathbf{x}_0$. This involves recovering the underlying background field influencing the state evolution from noisy, downsampled observations over time.

The number of timesteps is $T = 10$.

Although this problem is not realistic, it is useful for visualisation and understanding the behaviour of the algorithms, since the dynamics remain well-defined as the resolution is lowered.

A.2.1 Results

Figure 5 plots the influence of dimension upon various measures of inference quality. GEnBP estimates are substantially more accurate than GaBP, in terms of both posterior likelihood and MSE, as can be seen in the tighter clustering of the posterior samples about ground-truth. In this relatively low-dimensional problem, the Laplace approximation is also available for comparison; We see that the Laplace approximation is superior to both GaBP and GEnBP in terms of execution time. Its MSE performance is intermediate in quality, but its posterior likelihood, while similar to GaBP, is even less stable, and both are inferior to GEnBP. At small dimension both Laplace and GEnBP produce similar log likelihood estimates, with GaBP somewhat worse than either. As the dimension increases beyond 100, however, the gradient-based covariances in both GaBP and Laplace methods are increasingly likely not to be positive-definite, and thus the posterior log-likelihood may fail to be defined. GEnBP by contrast decays more slowly in log likelihood, and is always defined.

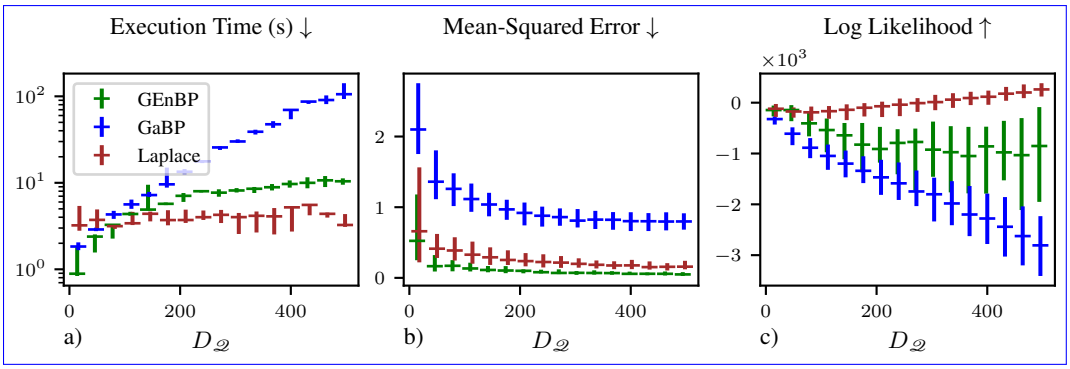
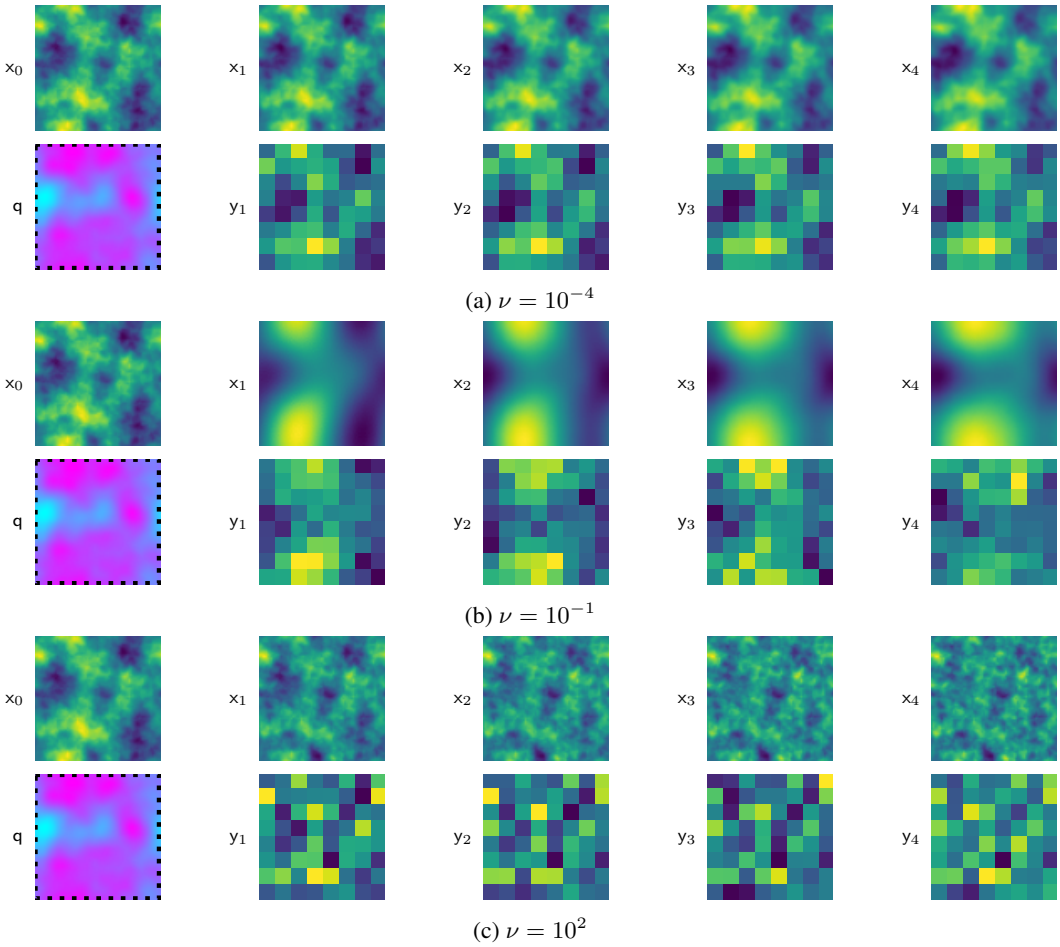


Figure 5: Influence of dimension $D_{\mathcal{Q}}$ in the transport example. Error bars are empirical 90% intervals from $n = 40$ runs. Missing log likelihood values denote undefined values arising from non-positive definite covariance estimates.



(a) $\nu = 10^{-4}$

(b) $\nu = 10^{-1}$

(c) $\nu = 10^{2}$

Figure 6: Three different Navier-Stokes simulations, with $\Delta t = 0.2$ and varying value of viscosity ν. The simulation is run on a 64×64 grid; observations are 8×8 and corrupted by white Gaussian noise with sd $\sigma = 0.1$ relative to a normalized unit scale.

A.3 NAVIER-STOKES SYSTEM

The Navier-Stokes equation defines a classic problem in fluid modeling, whose solution is of interest in many engineering applications. A full introduction to the Navier-Stokes equation is beyond the scope of this paper; see Ferziger et al. (2019) or one of the many other introductions.

Our implementation here is a 2D incompressible flow, solved using a spectral method with pytorch implementation from a simulator Li et al. (2020). Defining vorticity $\omega = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}$ and streamfunction ψ which generates the velocity field by

$$u = \frac{\partial \psi}{\partial y}$$
$$v = -\frac{\partial \psi}{\partial x}$$

the Navier Stokes equations are

1. Poisson equation $\nabla^2 \psi = -\omega$
2. Vorticity equation $\frac{\partial \omega}{\partial t} + u\frac{\partial \omega}{\partial x} + v\frac{\partial \omega}{\partial y} = \nu \nabla^2 \omega$

The discretisation of the equation is onto an $d \times d$ finite element basis for the domain, where each point summarises the vorticity field at at hat point. At each discretised time-step we inject additive mean-zero (d^2)-dimensional Gaussian white noise $\boldsymbol{\nu}_t \sim \mathcal{N}(\mathbf{0}, \sigma_{\boldsymbol{\eta}}^2 \mathrm{I})$ and static forcing term $\mathbf{q}$ to the velocity term before solving the equation.

2d matrices are represented as 2d vectors by stacking and unstacking as needed, so $\mathbf{x}_t = \mathrm{vec}(\psi)$. $\mathbf{y} = \mathrm{downsample}_\ell(\mathbf{x}_t) + \boldsymbol{\eta}$, where $\boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \sigma_{\boldsymbol{\eta}}^2 \mathrm{I})$ is the observation noise.

Initial state and forcings are sampled from a discrete periodic Gaussian random field with

$$PSD(k) = \sigma^2 \left(4\pi^2 \|k\|^2 + \tau^2\right)^{-\alpha}. \tag{27}$$

B EXPERIMENTAL DETAILS AND ADDITIONAL RESULTS

B.1 HARDWARE CONFIGURATION

Timings are on conducted on Dell PowerEdge C6525 Server AMD EPYC 7543 32-Core Processors running at 2.8 GHz (3.7 GHz turbo) with 256 MB cache. Float precision is set to 64 bits for all calculations. Memory limit is capped to 32 GB. Execution time is capped to 119 minutes.

B.2 ENSEMBLE SIZE

In the main text, we have left the matter of ensemble size N open. On one hand, ensembles of order $N \approx 10^2$ seem to be ample for the problems we have considered. We might be concerned that the trade-off of increasing ensemble size is unclear; on one hand many computational costs scale relatively simply as $\mathcal{O}(D^N 2 + N^3)$ (Section 3.4). On the other, we do not know how much extra precision we gain in general by increasing N. In Figure 7 we show a small experiment to examine this trade-off, where the ensemble size is increased from $N = 16$ to $N = 512$ in steps of size 16. The problem shows a rapid improvement as the ensemble increases to $N = 64$, but diminishing marginal returns. There is no clear cut-off as such, but we specify $N = 64$ as a default value for the experiments in the main text.

B.3 EMULATION USING GENBP

In Section 4 we demonstrated the utility of GEnBP in system identification problems. These are pedagogically useful, since they are well understood and require relatively little complexity in implementation. By the same token, however, they may fail to demonstrate the full range of capabilities of belief propagation methods, in that belief propagation is a general method for inference in graphical models, and not limited to system identification problems. We can apply GEnBP to a wide range of problems. However, a method which claims to scale to high dimensional

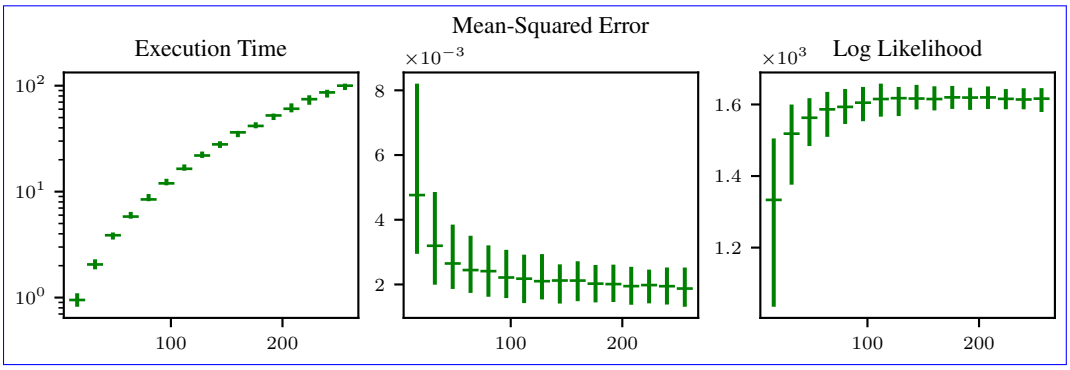


Figure 7: Performance of GEnBP with varying ensemble size N for the 1d transport system identification problem (Appendix A.2). Confidence intervals are empirical 95% intervals based on $n = 80$ independent runs.

problems naturally entails many complexities in practice; Large scale data simulation methods in climate, for example, are understood to require specialized clusters, custom software, and many intricate tweaks to the basic algorithms to achieve practical performance (e.g. O'Kane et al., 2021), which means that in practice they are neither pedagogic, not practical for a conference paper.

However, by setting up a reasonably tractable problem, we can demonstrate a proof-of-concept for the utility of GEnBP in a more complex setting. Noting that the Ensemble Kalman filter can be used for training Bayesian neural networks (**?**Schillings & Stuart, 2017) it seems that our chances of training a Bayesian neural network using GEnBP are good.

We devise an experiment which exploits this connection in the Gaussian Ensemble Belief propagation framework, and use the GEnBP algorithm to perform few-shot domain adaptation of a neural network emulator on unobserved states. The graphical model structure is shown in Figure 10.

The interpretation is as follows: We have a forward model $\mathcal{P} : \mathbf{x}_{t-1}, \mathbf{u} \mapsto \mathbf{x}_t$ which is a physical simulator, which could reproduce the state of the system exactly if we knew the unknown parameter $\mathbf{u}$, which is unknown. We have noisy observations $\mathcal{H} : \mathbf{x}_t \mapsto \mathbf{y}_t$ of the state of the system subject to a known observation operator, and we have a neural network $\mathcal{Q} : \mathbf{x}|\mathbf{u}$ which is trained to predict the state of the system given the unknown parameter $\mathbf{u}$. In the system identification problem we would stop there; However, in the few shot domain adaptation problem, we have a Bayesian neural network $\mathcal{N} : \hat{\mathbf{x}}_{t-1}, \mathbf{w} \mapsto \mathbf{x}_t$ which can to predict forward the state of the system given previous state $\hat{\mathbf{x}}_{t-1}$ and weights $\mathbf{w}$. We assume that the weights of the neural network have been trained on some possibly synthetic data on a related problem; we are interested in seeing whether we can adapt the neural network to the new, target problem by training it on our noisy observations, without access to the ground truth. This problem reduces to the problem of estimating a posterior distribution $p(\mathbf{w}|\mathbf{y})$, which is intractable in general, but can be approximated by GEnBP.

Fixing this structure, we select architecture. For the ground truth model we use the CFD model from Section 4, and for the neural network we use the Fourier Neural Operator (FNO) (Li et al., 2020) architecture, which is a neural network architecture designed for solving partial differential equations, with the useful property that it can produce high fidelity results with fewer than 10^6 parameters, putting it within the reach of GEnBP without intricate additional mathematical or computational machinery.

The final task is, with a small number of steps, train a model to estimate the random weights in a Bayesian neural network which can predict the state of the system given the previous state, which it observes only through a noisy observation operator. This class of problems represents a class immense practical interest, which is training emulators of real physical systems from data rather than from ground truth observations which are in practice unobtainable for most systems of interest. In this demonstration we have selected hyperparameters for ease rather than stringency, however, and we make no claims about the general application.

We simulate 10 time steps of a 32x32 grid of the CFD model, and apply GEnBP to the data generated by the first 5 timesteps, then seeing how well we are able to propagate the FNO model on the next 5 timesteps, to see if we have successfully capture the influence of the unobserved $\mathbf{q}$ in our updated operator. The prior distribution of the weights is a Gaussian with mean given by the optimal distribution of the network trained upon Navier-stokes simulation problems with no forcing term, and a posterior covariance estimated by the Adam optimizer 2nd moment estimate.

The experiment is configured on a 32×32 grid ($d = 32$) with a viscosity coefficient $\nu = 0.001$ and a temporal resolution of $\Delta t = 0.01$. We simulate a total of 10 time steps, applying GEnBP to the first 5 time steps to perform few-shot domain adaptation. Observations are recorded every 10 intervals, resulting in a downsampling factor of 5 (`downsample` $= 5$), and the observation noise is modeled as Gaussian with variance $\sigma^2_{\text{obs}} = 0.01$. In the inference process, we utilize an ensemble size of $N = 30$. For the neural network architecture, we employ a Fourier Neural Operator (FNO) with 12 Fourier modes, 32 hidden channels, and 4 layers, utilizing the Tensor Fourier Neural Operator variant, with a 5% rank factor, for a total of 1590030 weights. Results are shown in Figure 8. We note that the adapted emulator is capable of hewing close to the target density over the subsequent timesteps, while the unadapted emulator rapidly diverges.

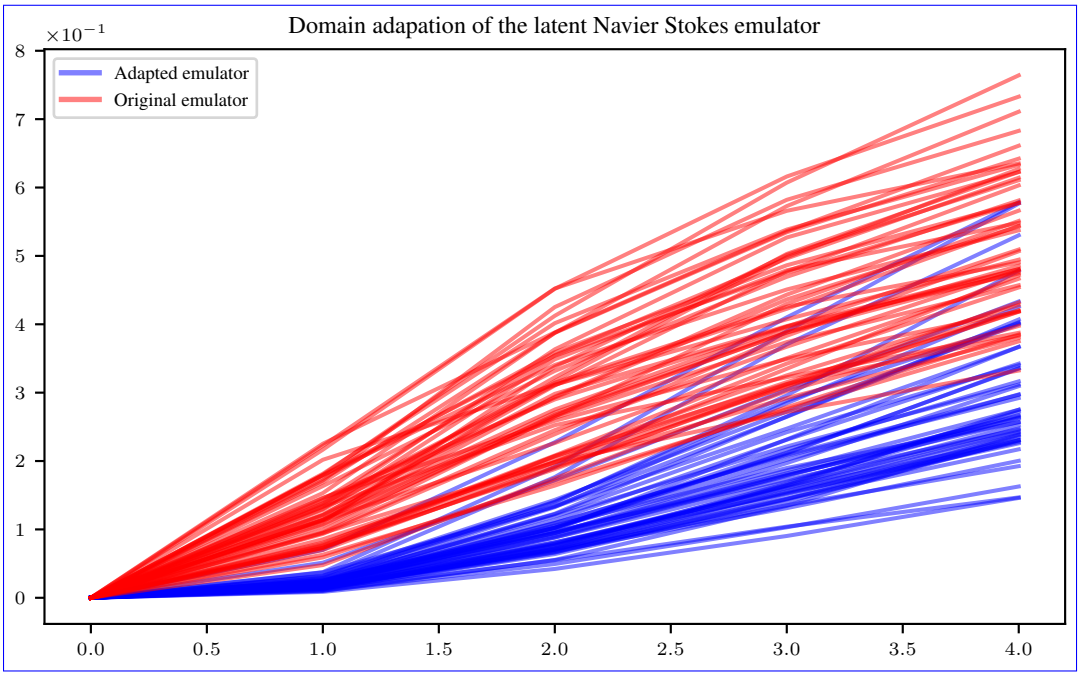


Figure 8: Performance of GEnBP in the emulator domain adaptation problem. Graph shows the maximum mean discrepancy between the observation-conditional state distribution, the un-adapted emulator, and the domain adapted emulator in terms of Maximum Mean Discrepancy with an isotropic Gaussian kernel of width 0.05.

C Minimum viable introduction to probabilistic graphical models

The field of probabilistic graphical models is mature and vast. We refer the reader to (e.g. Koller & Friedman, 2009; Wainwright & Jordan, 2008) for a thorough introduction. A miniature introduction sufficient for this paper may be found here.

For the purposes of the current paper, it is sufficient to understand that this is the framework that we use to enable inference in complicated hierarchical systems with partial observations. Consider, for example, the hierarchical process in Figure 9, which is a simplified representation of a weather model. An oceanic physics simulator and an atmospheric physics simulator provide forward predictions of the state of the ocean and atmosphere, and influence each other through physical coupling. Each of these dynamic systems is observed on a different time schedule by some form of telemetry (thermal

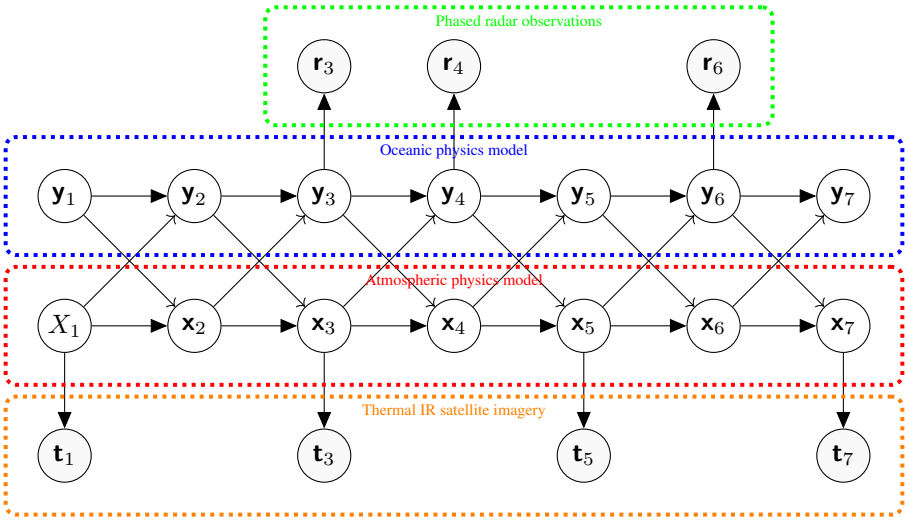


Figure 9: A complicated graphical model structure arising from a weather prediction problem. The model includes an oceanic physics simulator, an atmospheric physics simulator, phased radar observations, and thermal IR satellite imagery. The model is high-dimensional, noisy, and governed by nonlinear partial differential equations.

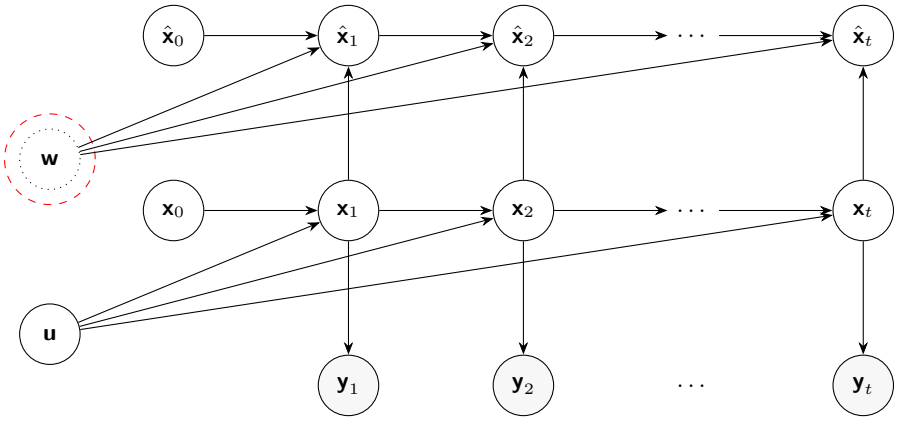


Figure 10: Directed graph of the generative model $\mathcal{M}_{\text{emu}}$ in an emulation problem, showing dependencies over time steps $t = 0, 1, 2, \ldots, T$. Observed variables $\mathbf{y}_t$ are shaded. The global parameter $\mathbf{w}$ is the marginal of interest.

satellite imagery for the oceans, phase radar for the atmosphere). All observations and state are high-dimensional, noisy and governed by nonlinear partial differential equations, including the observation models. Our primary target of inference is the 1-step ahead forward prediction for the atmosphere model. We could imagine many extensions of this, for example, to simultaneously learn the parameters of the observation or physics models as we learn their states.

Figure 10 by contrast shows a model closer to the system identification problem we consider in the experiments of Section 4. In this model we have supplement the system identification problem which tried to recover $\mathbf{u}$ with an additional problem that attempts to recover an emulator (for example a neural network) which can predict an approximation $\hat{\mathbf{x}}_t$ of the system at time t from the previous emulated state $\hat{\mathbf{x}}_{t-1}$. That is, we aim to predict a map from uncertain latent states to uncertain states by recovering a posterior distribution for neural network weights $\mathbf{w}$ to maximise the quality of this prediction, without access to the true states $\mathbf{x}_t$ except through noisy observations $\mathbf{y}_t$.

Graphical model formalisms provide a means of estimating states and parameters in these systems; the essential idea is that if we can find a set of rules that work for inference for arbitrary models,

then we can apply them to complicated models like this. This paper introduces such a generic set of rules, which in-principle applies ot arbitrarily complex graphical models, although scaling up to the complexity of the model in Figure 9 is out of scope for a short methodological paper such as this.

C.1 DIRECTED GRAPHS AND FACTOR GRAPHS

We begin with the structural equations defining the model. By sampling from the distribution over these ancestral variables then iteratively applying the generative model (Eq. 28), we obtain samples from the joint prior distribution of all random variates from the model. The state is jointly

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_{\mathscr{O}_1} \\ \vdots \\ \mathbf{x}_{\mathscr{O}_J} \end{bmatrix} = \begin{bmatrix} \mathcal{P}_1(\mathbf{x}_{\mathscr{I}_1}) \\ \vdots \\ \mathcal{P}_J(\mathbf{x}_{\mathscr{I}_J}) \end{bmatrix}. \tag{28}$$

The following conditions must hold for this to define a valid structural equation model

1. $\forall j, \mathscr{O}_j \cap \mathscr{I}_j = \emptyset$ (no self-loop in a step).
2. If $v \in \mathscr{O}_j$, then for $k > j$, v appears only in $\mathscr{I}_k$.

This defines a directed acyclic graph (DAG) over the variables, where the nodes are the variables and the edges are the generating equations. Each generating equation j induces an associated density $p(\boldsymbol{x}_{\mathscr{O}_j}|\boldsymbol{x}_{\mathscr{I}_j})$. The associated joint density is

$$p(\boldsymbol{x}) = \prod_j p\left(\boldsymbol{x}_{\mathscr{O}_j}|\boldsymbol{x}_{\mathscr{I}_j}\right). \tag{29}$$

Under some mild technical conditions (the local Markov condition, faithfulness and consistence - see Koller & Friedman (2009)), the representation given by the graph and the structural equations is equivalent to the joint distribution over the variables. We assume these conditions throughout.

Hereafter we use the following specific generative model as a running example:

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \mathbf{x}_4 \\ \mathbf{x}_5 \\ \mathbf{x}_6 \end{bmatrix} = \begin{bmatrix} \mathcal{P}_1() \\ \mathcal{P}_{12}(\mathbf{x}_1) \\ \mathcal{P}_{13}(\mathbf{x}_1) \\ \mathcal{P}_{234}(\mathbf{x}_2, \mathbf{x}_3) \\ \mathcal{P}_{25}(\mathbf{x}_2) \\ \mathcal{P}_{456}(\mathbf{x}_4, \mathbf{x}_5) \end{bmatrix}. \tag{30}$$

It is diagrammed in Figure 11 as a running example of the formalism. Its associated density can be factorised as

$$p(\boldsymbol{x}) = p_1(\boldsymbol{x}_1)p_{12}(\boldsymbol{x}_2|\boldsymbol{x}_1)p_{13}(\boldsymbol{x}_3|\boldsymbol{x}_1)p_{234}(\boldsymbol{x}_4|\boldsymbol{x}_2, \boldsymbol{x}_3)p_{25}(\boldsymbol{x}_5|\boldsymbol{x}_2)p_{456}(\boldsymbol{x}_6|\boldsymbol{x}_4, \boldsymbol{x}_5) \tag{31}$$

The directed generative model $\mathcal{M}$, and equivalently, conditional density, corresponds to Figure 11a.

The directed graphical model has the advantage of being intuitive, but it is not the most convenient for inference.

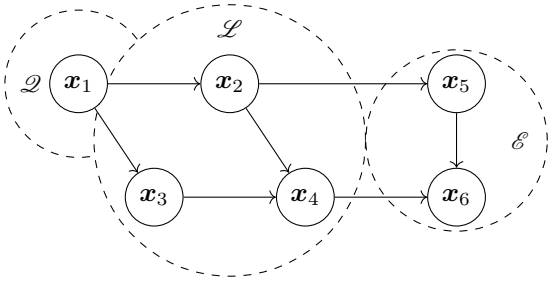
The factor graph form (Frey et al., 1997; Kschischang et al., 2001) discards the directions of the arrows induced by the generative models and works only with generic factors, transforming

$$p(\boldsymbol{x}_1|\boldsymbol{x}_2) \to f(\boldsymbol{x}_1, \boldsymbol{x}_2). \tag{32}$$

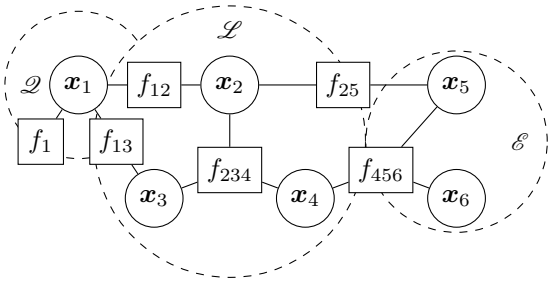
Our running example density (Eq. 31) becomes

$$p(\boldsymbol{x}) = f_1(\boldsymbol{x}_1)f_{12}(\boldsymbol{x}_1, \boldsymbol{x}_2)f_{13}(\boldsymbol{x}_1, \boldsymbol{x}_3)f_{234}(\boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4)f_{25}(\boldsymbol{x}_2, \boldsymbol{x}_5)f_{456}(\boldsymbol{x}_4, \boldsymbol{x}_5, \boldsymbol{x}_6). \tag{33}$$

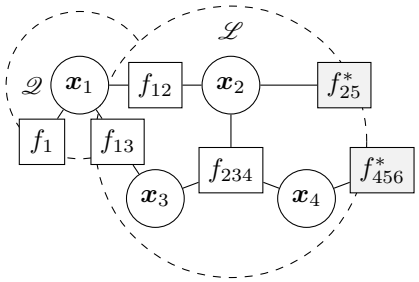
The corresponding factor graph is shown in Figure 11b. The factor graph is a bipartite graph with nodes corresponding to both variables and factors. This representation has the advantage that the associated approximate belief-updating message-passing rules are simple and local. These rules we introduce in Section C.2.



(a) Prior directed graph of generative model $\mathcal{M}$,
$p(\boldsymbol{x}) = p_1(\boldsymbol{x}_1)p_{12}(\boldsymbol{x}_2|\boldsymbol{x}_1)p_{13}(\boldsymbol{x}_3|\boldsymbol{x}_1)p_{25}(\boldsymbol{x}_5|\boldsymbol{x}_2)p_{234}(\boldsymbol{x}_4|\boldsymbol{x}_2,\boldsymbol{x}_3)p_{456}(\boldsymbol{x}_6|\boldsymbol{x}_4,\boldsymbol{x}_5)$.



(b) Prior factor graph $\mathcal{G}$ for generative model $\mathcal{M}$,
$p(\boldsymbol{x}) = f_1(\boldsymbol{x}_1)f_{12}(\boldsymbol{x}_1,\boldsymbol{x}_2)f_{13}(\boldsymbol{x}_1,\boldsymbol{x}_3)f_{234}(\boldsymbol{x}_2,\boldsymbol{x}_3,\boldsymbol{x}_4)f_{25}(\boldsymbol{x}_2,\boldsymbol{x}_5)f_{456}(\boldsymbol{x}_4,\boldsymbol{x}_5,\boldsymbol{x}_6)$.



(c) Posterior factor graph $\mathcal{G}^*$ of generative model $\mathcal{M}$ after observing $\boldsymbol{x}_5$ and $\boldsymbol{x}_6$,
$p(\boldsymbol{x}_1,\boldsymbol{x}_2,\boldsymbol{x}_3,\boldsymbol{x}_4|\boldsymbol{x}_5,\boldsymbol{x}_6) \propto$
$f_1(\boldsymbol{x}_1)f_{12}(\boldsymbol{x}_1,\boldsymbol{x}_2)f_{13}(\boldsymbol{x}_1,\boldsymbol{x}_3)f_{234}(\boldsymbol{x}_2,\boldsymbol{x}_3,\boldsymbol{x}_4)f^*_{25}(\boldsymbol{x}_2)f^*_{456}(\boldsymbol{x}_4)$

Figure 11: The graphical model variants used in this paper for example model Figure 11: (a) prior generative graph, (b) prior factor graph, and (c) posterior factor graph; each with query nodes $\mathscr{Q} = \{1\}$, latent nodes $\mathscr{L} = \{2, 3, 4\}$, and observed nodes $\mathscr{E} = \{5, 6\}$.

We introduce one final graph transformation of use in this paper, which produces the conditional graph. We recall that in practice we are interested in the evidence-conditional, posterior distribution, $\mathbf{x}_{\mathscr{Q}}$ given the evidence variables $\mathbf{x}_{\mathscr{E}} = \boldsymbol{x}^*_{\mathscr{E}}$, (Eq. 2) i.e.

$$p(\boldsymbol{x}_{\mathscr{Q}}|\mathbf{x}_{\mathscr{E}}{=}\boldsymbol{x}^*_{\mathscr{E}}) = \frac{p(\boldsymbol{x}_{\mathscr{Q}}, \boldsymbol{x}_{\mathscr{E}}{=}\boldsymbol{x}^*_{\mathscr{E}})}{p(\boldsymbol{x}_{\mathscr{E}}{=}\boldsymbol{x}^*_{\mathscr{E}})} \tag{34}$$

$$= \frac{\int p(\boldsymbol{x}_{\mathscr{Q}}, \boldsymbol{x}_{\mathscr{L}}, \boldsymbol{x}_{\mathscr{E}}{=}\boldsymbol{x}^*_{\mathscr{E}})\mathrm{d}\boldsymbol{x}_{\mathscr{L}}}{\int p(\boldsymbol{x}_{\mathscr{Q}}, \boldsymbol{x}_{\mathscr{L}}, \boldsymbol{x}_{\mathscr{E}}{=}\boldsymbol{x}^*_{\mathscr{E}})\mathrm{d}\boldsymbol{x}_{\mathscr{Q}}\mathrm{d}\boldsymbol{x}_{\mathscr{L}}}. \tag{35}$$

This can also be represented as a factor graph, as shown in Figure 11c. The rule in this case is simply to use the factor graph corresponding to the factorisation induced by the posterior, which has a slightly different set of factors, since the observed nodes are no longer random, e.g.

$$p(\boldsymbol{x}_{\mathscr{Q}}|\boldsymbol{x}_{\mathscr{E}}{=}\boldsymbol{x}^*_{\mathscr{E}}) \propto p_1(\boldsymbol{x}_1)p_{12}(\boldsymbol{x}_2|\boldsymbol{x}_1)p_{13}(\boldsymbol{x}_3|\boldsymbol{x}_1)p_{234}(\boldsymbol{x}_4|\boldsymbol{x}_2,\boldsymbol{x}_3)p_{25}(\boldsymbol{x}_2|\boldsymbol{x}_5{=}\boldsymbol{x}^*_5)p_{456}(\boldsymbol{x}_4|\boldsymbol{x}_5{=}\boldsymbol{x}^*_5,\boldsymbol{x}_6{=}\boldsymbol{x}^*_6) \tag{36}$$

$$\propto f_1(\boldsymbol{x}_1)f_{12}(\boldsymbol{x}_1,\boldsymbol{x}_2)f_{13}(\boldsymbol{x}_1,\boldsymbol{x}_3)f_{234}(\boldsymbol{x}_2,\boldsymbol{x}_3,\boldsymbol{x}_4)f^*_{25}(\boldsymbol{x}_2)f^*_{456}(\boldsymbol{x}_4). \tag{37}$$

C.2 Factor Graph Gaussian Belief Propagation

Basic loopy belief propagation (Murphy et al., 1999) is a simple and efficient algorithm for approximate inference of marginals over certain variables in factor graphs. There is a rich literature on when and how this algorithm converges to the true marginal, (Wainwright & Jordan, 2008; Yedidia et al., 2005). We do not concern ourselves with those details in this work, but simply follow industrial practice (e.g. Dellaert & Kaess, 2017) in treating it as a good enough approximation for our purposes. Following the terminology of Ortiz et al. (2021).

Proposition 1 (Belief Propagation on Factor Graphs)**.** *By iteratively and synchronously propagating the following messages between all nodes in the factor graph,*

$$m_{f_j \to \mathbf{x}_k} = \int \left(f_j\left(\boldsymbol{x}_{\mathcal{N}_j}\right) \prod_{i \in \mathcal{N}_j \setminus k} m_{\mathbf{x}_i \to f_j} \right) \mathrm{d}\boldsymbol{x}_{\mathcal{N}_j \setminus k}, \tag{5}$$

$$m_{\mathbf{x}_k \to f_j} = \prod_{s \in \mathcal{N}_k \setminus j} m_{f_s \to \mathbf{x}_k}. \tag{6}$$

BP approximates the marginals for each variable by the product of incoming messages,

$$b_{\mathcal{G}}\left(\boldsymbol{x}_k\right) = \prod_{s \in \mathcal{N}_k} m_{f_s \to \mathbf{x}_k} \approx \int p(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}_{\setminus k}. \tag{7}$$

The factor graph belief propagation algorithm for generic factor graphs using these messages is presented in Algorithm 1.

Algorithm 1 Loopy Low-rank Belief Propagation over Factor Graph $\mathcal{G}$

Require: Factor graph $\mathcal{G}$ with variable nodes $\{\mathbf{x}_k\}_k$ and factor nodes $\{f_j\}_j$
Require: Initial messages $\mathbf{x}_k \to f_j$ and $f_j \to \mathbf{x}_k$
Ensure: Approximate marginal beliefs $b(\mathbf{x}_k)$ for all $\mathbf{x}_k \in \mathcal{G}$

```
Initialize message queues Q_{x_k → f_j} and Q_{f_j → x_k} to be empty
while not converged do
    for each factor f_j ∈ G do
        for each variable x_k ∈ N_{f_j} do
            Send factor to variable message (Eq. 5) to each neighbour
        end for
    end for
    for each variable x_k ∈ G do
        for each factor f_j ∈ N_{x_k} do
            Send variable to factor message (Eq. 6) to each neighbour
        end for
        Update belief b(x_k) (Eq. 7)
    end for
    Check for convergence criteria
end while
Return b(x_k) for all x_k ∈ G
```

C.3 Gaussian Factor updates

Finally we note that the factor graph formalism is particularly convenient for Gaussian models, as the Gaussian density is closed under multiplication, conditioning and marginalisation. We introduced this in Section 2.2.3, but write it out in full here.

As an exponential family distribution (Wainwright & Jordan, 2008), it become convenient to work with Gaussians in canonical form (Eustice et al., 2006). We write $\phi_M\left(\boldsymbol{x}; \boldsymbol{m}, \mathrm{K}\right)$ for ϕ_M the Gaussian density with moment parameters mean $\boldsymbol{m}$ and covariance K.

Proposition 5. *As in Definition 1, we partition the random vector*

$$\mathbf{x}_j \sim \phi_M\left(\begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{x}_\ell \end{bmatrix}; \begin{bmatrix} \boldsymbol{m}_k \\ \boldsymbol{m}_\ell \end{bmatrix}, \begin{bmatrix} \mathrm{K}_{kk} & \mathrm{K}_{k\ell} \\ \mathrm{K}_{\ell k} & \mathrm{K}_{\ell\ell} \end{bmatrix} \right). \tag{38}$$

We define $\hat{\mathrm{K}} \coloneqq (\mathrm{K}_{jj}^{-1} + {\mathrm{K}'_{jj}}^{-1})^{-1}$. *Where the node and factors are Gaussian, the operations of Definition 1 have the following form,*

Conditioning: $$\phi_M\left(\boldsymbol{x}_j; \boldsymbol{m}_j, \mathrm{K}_{jj}\right), \boldsymbol{x}_k^* \mapsto \phi_M(\boldsymbol{x}_\ell; \boldsymbol{m}_\ell + \mathrm{K}_{\ell k}\mathrm{K}_{kk}^{-1}(\boldsymbol{x}_k^* - \boldsymbol{m}_k), \mathrm{K}_{\ell\ell} - \mathrm{K}_{\ell k}\mathrm{K}_{kk}^{-1}\mathrm{K}_{k\ell}); \tag{39}$$

Marginalisation: $$\phi_M\left(\boldsymbol{x}_j; \boldsymbol{m}_j, \mathrm{K}_{jj}\right) \mapsto \phi_M\left(\boldsymbol{x}_k; \boldsymbol{m}_k, \mathrm{K}_{kk}\right); \tag{40}$$

Multiplication: $$\phi'_M\left(\boldsymbol{x}_j; \boldsymbol{m}'_j, \mathrm{K}'_{jj}\right), \phi_M\left(\boldsymbol{x}_k; \boldsymbol{m}_k, \mathrm{K}_{kk}\right) \mapsto \phi_M\left(\boldsymbol{x}_j; \hat{\mathrm{K}}(\mathrm{K}_{jj}^{-1}\boldsymbol{m}_j + {\mathrm{K}'_{kk}}^{-1}\boldsymbol{m}'_j), \hat{\mathrm{K}}\right); \tag{41}$$

$$\phi'_C\left(\boldsymbol{x}_j; \boldsymbol{n}'_j, \mathrm{P}'_{jj}\right), \phi_C\left(\boldsymbol{x}_k; \boldsymbol{n}_k, \mathrm{P}_{kk}\right) \mapsto \phi_C\left(\boldsymbol{x}_j; \boldsymbol{n}'_j + \boldsymbol{n}_j, \mathrm{P}' + \mathrm{P}\right)$$

Proof: Bickson (2009). □

The classic GaBP algorithm includes many more details not captured in these minimal rules. Most importantly, since GaBP is frequently applied to factors that are not truly Gaussian, it needs a rule for finding a Gaussian distribution which approximates some factor potential.

C.4 Gaussian approximation of non-Gaussian factor potentials

Throughout this section we assume without loss of generality that all factor potentials are bivariate with $\mathbf{x}_2 = \mathcal{P}(\mathbf{x}_1)$. If this is not the case, we can stack and relabel the variates to make it bivariate.

The classic choice for approximating the factor potential generated by a nonlinear process is *propagation-of-errors* a.k.a. the *δ-method* (Dorfman, 1938). The δ-method relies upon the approximation, justified by Taylor expansion, that for a function f and a random variable $\mathbf{x}_1$,

$$\mathbb{E}f(\mathbf{x}_1) \approx f(\mathbb{E}\mathbf{x}_1). \tag{42}$$

When we estimate, e.g. the joint covariance of the factor potential, we choose

$$f : [\boldsymbol{x}] \mapsto \begin{bmatrix} \boldsymbol{x} \\ \mathcal{P}(\boldsymbol{x}) \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \mathcal{P}(\boldsymbol{x}) \end{bmatrix}^\top - \mathbb{E}\begin{bmatrix} \boldsymbol{x} \\ \mathcal{P}(\boldsymbol{x}) \end{bmatrix} \mathbb{E}\begin{bmatrix} \boldsymbol{x} \\ \mathcal{P}(\boldsymbol{x}) \end{bmatrix}^\top \tag{43}$$

The difference

$$\mathcal{E}_{\text{Jensen}}(\mathbf{x}_1, f) \coloneqq \mathbb{E}f(\mathbf{x}_1) - f(\mathbb{E}\mathbf{x}_1) \tag{44}$$

is called the *Jensen Gap* and is one source of error in the linearisation. The Jensen gap may sometimes be bounded, e.g. in terms of the coefficient of Hölder continuity of f and the moments of $\mathbf{x}_1$ (Gao et al., 2020). In practice, this seems to be rarely done.

Further, $\mathcal{P}$ and thus f, for problems of interest is still intractable in even this simplified calculation, and thus the δ-method further approximates it using a surrogate given by first-order Taylor expansion about the mean,

$$\hat{f}(\boldsymbol{x}) \approx f(\mathbb{E}\mathbf{x}_1) + \nabla_{\boldsymbol{x}'} f(\boldsymbol{x}')|_{\boldsymbol{x}'=\mathbb{E}\mathbf{x}_1}(\boldsymbol{x} - \mathbb{E}\mathbf{x}_1) \tag{45}$$

The error for evaluating the expectation of some Taylor approximation $\hat{f}$ of f is

$$\mathcal{E}_{\text{Taylor}}(\mathbf{x}_1, f, \hat{f}) \coloneqq \mathbb{E}f(\mathbf{x}_1) - \mathbb{E}\hat{f}(\mathbf{x}_1). \tag{46}$$

The ultimate approximation of the covariance by the δ method will include both the Jensen gap and the Taylor error, and additionally the inputs may not even be Gaussian. We can say little generally about the contribution of all such errors; there do not seem to be any results justifying its optimality in terms of e.g. a variational bound. In practice we may choose alternative methods based on practical effectiveness on the problem of interest.

While the δ-method is simple and easy, it is not the only choice, and is empirically not necessarily favourable in either compute or accuracy, which deficiency we address in the sequel.

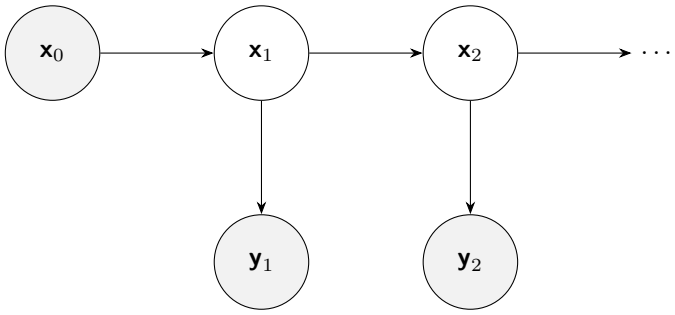


Figure 12: Generative model for state filtering problem, hidden states $\mathbf{x}_t$ and observation $\mathbf{y}_t$ for $t = 1, 2, \ldots$.

D MINIMUM VIABLE INTRODUCTION TO THE ENSEMBLE KALMAN FILTER

The field of Ensemble Kalman methods is mature and vast. We refer the reader to e.g. Evensen (2009) for a thorough introduction. A miniature introduction sufficient for this paper may be found here.

The central idea is the following: given the state-space model (Figure 12) with hidden states $\mathbf{x}_t$ and observations $\mathbf{y}_t$, we wish to estimate the hidden states given the observations. The joint prior density of the state and observations is

$$p(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T, \boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_T) =: p(\boldsymbol{x}_{0:T}, \boldsymbol{y}_{1:T}) \tag{47}$$

$$= \prod_{t=1}^{T} p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) p(\boldsymbol{y}_t|\boldsymbol{x}_t). \tag{48}$$

Filtering leverages the fact that

$$p(\boldsymbol{x}_{0:T}, \boldsymbol{y}_{1:T}) = p(\boldsymbol{y}_T|\boldsymbol{x}_T) p(\boldsymbol{x}_T|\boldsymbol{x}_{T-1}) p(\boldsymbol{x}_{1:T-1}, \boldsymbol{y}_{1:T}) \tag{49}$$

so we can solve the problem by induction,

$$p(\boldsymbol{x}_{0:T}|\boldsymbol{y}_{1:T}) \propto p(\boldsymbol{y}_T|\boldsymbol{x}_T) p(\boldsymbol{x}_T|\boldsymbol{x}_{T-1}) p(\boldsymbol{x}_{1:T-1}, \boldsymbol{y}_{1:T-1}) \tag{50}$$

$$\propto p(\boldsymbol{y}_T|\boldsymbol{x}_T) p(\boldsymbol{x}_T|\boldsymbol{x}_{T-1}) p(\boldsymbol{x}_{T-1}|\boldsymbol{y}_{1:T-1}). \tag{51}$$

If we know $p(\boldsymbol{x}_{1:T-1}|\boldsymbol{y}_{1:T-1})$, to update to a new estimate of $p(\boldsymbol{x}_T|\boldsymbol{y}_{1:T})$ we only need to be able to multiply its density by $p(\boldsymbol{x}_T|\boldsymbol{x}_{T-1}) p(\boldsymbol{y}_T|\boldsymbol{x}_T)$ and normalize; thus we can always calculate an estimate of the new state given the previous state estimate and the current observations.

Under Gaussianity of all distributions and linearity of all processes (e.g. Petersen & Pedersen (2012)) we can represent this density update in terms of the distribution parameters. For some $\boldsymbol{m}_{T-1}, \mathrm{K}_{T-1}$,

$$p(\boldsymbol{x}_{T-1}|\boldsymbol{y}_{1:T-1}) = \phi_M(\boldsymbol{x}_{T-1}; \boldsymbol{m}_{T-1}, \mathrm{K}_{T-1}) \tag{52}$$

Linearity implies that there exist $\boldsymbol{d}_{T-1}$, F_{T-1},Q_T the forward propagation operator $\boldsymbol{x}_T = \mathcal{P}_{T-1}(\boldsymbol{x}_{T-1})$ of $\boldsymbol{x}_T$ is

$$\boldsymbol{x}_T|\boldsymbol{x}_{T-1} = \mathcal{P}(\boldsymbol{x}_{T-1}) \tag{53}$$

$$= \mathrm{F}_{T-1}\boldsymbol{x}_{T-1} + \boldsymbol{d}_{T-1} + \boldsymbol{\epsilon}_T \tag{54}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathrm{Q}_T)$ so that

$$p\left(\begin{bmatrix}\boldsymbol{x}_{T-1}\\ \boldsymbol{x}_T\end{bmatrix}\middle|\, \boldsymbol{y}_{1:T-1}\right) \tag{55}$$

$$= \phi_M\left(\begin{bmatrix}\boldsymbol{x}_{T-1}\\ \boldsymbol{x}_T\end{bmatrix}; \begin{bmatrix}\boldsymbol{m}_{T-1}\\ \boldsymbol{d}_{T-1} + \mathrm{F}_{T-1}\boldsymbol{m}_{T-1},\end{bmatrix}, \begin{bmatrix}\mathrm{K}_{T-1} & \mathrm{K}_{T-1}\mathrm{F}_{T-1}^\top\\ \mathrm{F}_{T-1}\mathrm{K}_{T-1} & \mathrm{F}_{T-1}\mathrm{K}_{T-1}\mathrm{F}_{T-1}^\top + \mathrm{Q}_T\end{bmatrix}\right). \tag{56}$$

The $\boldsymbol{x}_T$ marginal is again Gaussian,

$$p(\boldsymbol{x}_T|\boldsymbol{x}_{T-1}, \boldsymbol{y}_{1:T-1}) = \phi_M(\boldsymbol{x}_T; \tilde{\boldsymbol{m}}_T, \tilde{\mathrm{K}}_T) \tag{57}$$

for $\tilde{\boldsymbol{m}}_T = \mathrm{F}_{T-1}\boldsymbol{m}_{T-1} + \boldsymbol{d}_{T-1}$ and $\tilde{\mathrm{K}}_T = \mathrm{F}_{T-1}\mathrm{K}_{T-1}\mathrm{F}_{T-1}^\top + \mathrm{Q}_T$.

It remains to condition on the observation $\boldsymbol{y}_T$. We suppose that the measurement model $\boldsymbol{y}_T = \mathcal{H}_T(\boldsymbol{x}_T)$ is linear and Gaussian, so that

$$\boldsymbol{y}_T|\boldsymbol{x}_T = \mathcal{H}(\boldsymbol{x}_T) \tag{58}$$

$$= \mathrm{H}_T\boldsymbol{x}_T + \boldsymbol{e}_T + \boldsymbol{\varepsilon}_T \tag{59}$$

where $\boldsymbol{\varepsilon}_T \sim \mathcal{N}(\mathbf{0}, \mathrm{R}_T)$. The joint density of $\boldsymbol{x}_T$ and $\boldsymbol{y}_T$ is

$$p\left(\begin{bmatrix}\boldsymbol{x}_T\\ \boldsymbol{y}_T\end{bmatrix}\middle|\,\boldsymbol{y}_{1:T-1}\right) = \phi_M\left(\begin{bmatrix}\boldsymbol{x}_T\\ \boldsymbol{y}_T\end{bmatrix}; \begin{bmatrix}\tilde{\boldsymbol{m}}_T\\ \mathrm{H}_T\tilde{\boldsymbol{m}}_T + \boldsymbol{e}_T\end{bmatrix}, \begin{bmatrix}\tilde{\mathrm{K}}_T & \tilde{\mathrm{K}}_T\mathrm{H}_T^\top\\ \mathrm{H}_T\tilde{\mathrm{K}}_T & \mathrm{H}_T\tilde{\mathrm{K}}_T\mathrm{H}_T^\top + \mathrm{R}_T\end{bmatrix}\right). \tag{60}$$

but we can use the standard formula for conditional Gaussian distributions to write

$$p(\boldsymbol{x}_T|\boldsymbol{y}_{1:T}) = \phi_M(\boldsymbol{x}_T; \boldsymbol{m}_T, \mathrm{K}_T) \tag{61}$$

where

$$\mathrm{K}_T = \tilde{\mathrm{K}}_T - \tilde{\mathrm{K}}_T\mathrm{H}_T(\mathrm{H}_T\tilde{\mathrm{K}}_T\mathrm{H}_T^\top + \mathrm{R}_T)^{-1}\mathrm{H}_T^\top\tilde{\mathrm{K}}_T^\top \tag{62}$$

$$\boldsymbol{m}_T = \tilde{\boldsymbol{m}}_T + \mathrm{K}_T\mathrm{H}_T^\top(\mathrm{H}_T\tilde{\mathrm{K}}_T\mathrm{H}_T^\top + \mathrm{R}_T)^{-1}\left(\boldsymbol{y}_T - \mathrm{H}_T\tilde{\boldsymbol{m}}_T - \boldsymbol{e}_T\right). \tag{63}$$

i.e. there is a closed-form update.

In practice, we may want to apply this method where at least one of the the state transition and observation models is nonlinear. The Extended Kalman Filter (EKF), uses a linearization of the state transition and observation models to approximate the update. We instead find a linear approximation to the state transition (Eq. 54) and observation (Eq. 59) models, by propagation of errors, setting

$$\boldsymbol{d}_{T-1} \approx \mathbb{E}[\mathcal{P}_{T-1}(\boldsymbol{m}_{T-1})] \tag{64}$$

$$\mathrm{F}_{T-1} \approx \nabla_{\boldsymbol{x}_{T-1}}\mathbb{E}[\mathcal{P}_{T-1}(\boldsymbol{m}_{T-1})] \tag{65}$$

$$\boldsymbol{e}_T \approx \mathbb{E}[\mathcal{H}_T(\tilde{\boldsymbol{m}}_T)] \tag{66}$$

$$\mathrm{H}_T \approx \nabla_{\boldsymbol{x}_T}\mathbb{E}[\mathcal{H}_T(\tilde{\boldsymbol{m}}_T)]. \tag{67}$$

and $\mathrm{Q}_T = \sigma^2\mathrm{I}$, $\mathrm{R}_T = \gamma^2\mathrm{I}$ are set to some diagonal covariance matrices. This type of linearised approximation is also used in the GaBP algorithm.

The EnKF takes a different approach to approximate inference, finding sample-based alternatives to the forward-propagation (Eq. 57) and the observation update (Eq. 61) Where the GaBP and EKF method summarises inference in terms of the parameters of a Gaussian distribution, the EnKF summarises the joint distribution in terms of an *ensemble* of Monte Carlo samples, i.e. a matrix of N samples, $\mathrm{X} \coloneqq \left[\boldsymbol{x}^{(n)}\right]_{1\le n\le N}$ drawn i.i.d. from the prior distribution, $\boldsymbol{x}^{(n)} \sim \phi_M(\boldsymbol{x}; \boldsymbol{m}, \mathrm{K})$. We reprise the definitions introduced in Section 2.3 by which the statistics of ensemble X define an implied Gaussian density, $\boldsymbol{x} \sim \phi_M(\boldsymbol{x}; \overline{\mathrm{X}}, \widehat{\mathrm{Var}}_{\mathrm{V}}\mathrm{X})$.

$$\mathrm{A}_N \coloneqq \begin{bmatrix}\frac{1}{N} & \cdots & \frac{1}{N}\end{bmatrix}^\top, \tag{68}$$

$$\mathrm{B}_N \coloneqq [1 \quad \cdots \quad 1] \tag{69}$$

$$\mathrm{C}_N \coloneqq \mathrm{I}_N - \mathrm{A}_N\mathrm{B}_N \tag{70}$$

$$\overline{\mathrm{X}} \coloneqq \mathrm{X}\mathrm{A}_N \qquad \text{Ensemble mean} \tag{71}$$

$$\breve{\mathrm{X}} \coloneqq \mathrm{X} - \mathrm{X}\mathrm{A}_N\mathrm{B}_N = \mathrm{X}\mathrm{C}_N. \qquad \text{Ensemble deviation} \tag{15}$$

$$\widehat{\mathbb{E}}\mathrm{X} = \overline{\mathrm{X}} \tag{72}$$

$$\widehat{\mathrm{Var}}_{\mathrm{V}}\mathrm{X} = \tfrac{1}{N-1}\breve{\mathrm{X}}\breve{\mathrm{X}}^\top + \mathrm{V} \tag{73}$$

$$\widehat{\mathrm{Cov}}(\mathrm{X}, \mathrm{Y}) = \tfrac{1}{N-1}\breve{\mathrm{X}}\breve{\mathrm{Y}}^\top \tag{74}$$

V is a diagonal matrix *nugget* term, typically set to $\sigma^2\mathrm{I}$.

The EnKF equivalent of the Kalman filter forward joint distribution (Eq. 56) is then

$$\begin{bmatrix}\mathrm{X}_{T-1}\\ \mathrm{X}_T\end{bmatrix}\middle|\,\boldsymbol{y}_{1:T-1} = \begin{bmatrix}\mathrm{X}_{T-1}|\boldsymbol{y}_{1:T-1}\\ \mathcal{P}(\mathrm{X}_{T-1}|\boldsymbol{y}_{1:T-1})\end{bmatrix}. \tag{75}$$

By assumption this means that if we wished to calculate the density of the joint state, it would be

$$p(\boldsymbol{x}_{T-1}, \boldsymbol{x}_T | \boldsymbol{y}_{1:T-1}) = \phi_M \left(\begin{bmatrix} \boldsymbol{x}_{T-1} \\ \boldsymbol{x}_T \end{bmatrix} ; \begin{bmatrix} \widehat{\mathbb{E}} \mathrm{X}_T \\ \widehat{\mathbb{E}}[\mathcal{P}(\mathrm{X}_T)] \end{bmatrix}, \begin{bmatrix} \widehat{\mathrm{Var}}_{\mathrm{V}} \mathrm{X}_T & \widehat{\mathrm{Cov}}(\mathrm{X}_T, \mathcal{P}(\mathrm{X}_T)) \\ \widehat{\mathrm{Cov}}(\mathcal{P}(\mathrm{X}_T), \mathrm{X}_T) & \widehat{\mathrm{Var}}_{\mathrm{U}} \mathcal{P}(\mathrm{X}_T) \end{bmatrix} \right). \tag{76}$$

The marginal (Eq. 57) is simply the truncation of the above.

It turns out that we never need to explicitly evaluate such a density, because of the following result:

Proposition 2. *Partition* $\mathbf{x}^\top = \begin{bmatrix} \mathbf{x}_k^\top & \mathbf{x}_\ell^\top \end{bmatrix}$ *such that* $\mathrm{X}^\top = \begin{bmatrix} \mathrm{X}_k^\top & \mathrm{X}_\ell^\top \end{bmatrix}$. *Assume the ensemble* X *follows the Gaussian distribution:*

$$\mathrm{X} \sim \phi_M \left(\begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{x}_\ell \end{bmatrix} ; \begin{bmatrix} \overline{\mathrm{X}}_k \\ \overline{\mathrm{X}}_\ell \end{bmatrix}, \begin{bmatrix} \widehat{\mathrm{Var}}_{\mathrm{V}} \mathrm{X}_k & \widehat{\mathrm{Cov}}(\mathrm{X}_\ell, \mathrm{X}_k) \\ \widehat{\mathrm{Cov}}(\mathrm{X}_k, \mathrm{X}_\ell) & \widehat{\mathrm{Var}}_{\mathrm{V}}(\mathrm{X}_\ell) \end{bmatrix} \right). \tag{16}$$

In ensemble form, conditioning (Eq. 9) is performed as:

$$\mathrm{X}, \boldsymbol{x}_k^* \mapsto \mathrm{X}_\ell + \widehat{\mathrm{Cov}}(\mathrm{X}_\ell, \mathrm{X}_k) \widehat{\mathrm{Var}}_{\mathrm{V}}^{-1}(\mathrm{X}_k)(\boldsymbol{x}_k^* \mathrm{B} - \mathrm{X}_k) \tag{17}$$

The computational cost of solving Equation 17 is $\mathcal{O}(N^3 + N^2 D_{\mathbf{x}_k})$. *Marginalization (Eq. 10) is simply truncation, i.e.,* $\mathrm{X} \mapsto \mathrm{X}_k$.

Proof: The equality Equation 17 follows from 6 with the substitution of Equation 16.

Notably we do not need to construct $\widehat{\mathrm{Var}}_{\mathrm{V}}(\mathrm{X}_\ell)$, and can use Woodbury formula to efficiently solve the linear system involving $\widehat{\mathrm{Var}}_{\mathrm{V}}^{-1}(\mathrm{X}_k)$. □

We can use this to calculate an observation-conditional update, since

$$\begin{bmatrix} \mathrm{X}_T \\ \mathrm{Y}_T \end{bmatrix} \Bigg| \boldsymbol{y}_{1:T-1} = \begin{bmatrix} \mathrm{X}_T | \boldsymbol{y}_{1:T-1} \\ \mathcal{H}(\mathrm{X}_T | \boldsymbol{y}_{1:T-1}) \end{bmatrix}. \tag{77}$$

We can use Equation 17 to calculate $\mathrm{X}_T | \boldsymbol{y}_{1:T}$ by using $\boldsymbol{y}_T$ as the observations $\boldsymbol{x}_k^*$ in the update. We have thus obtained sampled from the filtered distribution without ever evaluating its density.

D.1 Gaussian approximation of non-Gaussian factor potentials

As with the the GaBP, the EnKF is frequently applied to non-Gaussian distributions. Unlike the GaBP, we never use the δ-method. Instead, the *empirical* joint (Eq. 77) already finds the moments of an approximating distribution. The price we pay that as a stochastic approximation, we have now introduced aleatoric noise to the estimate. Despite this, the EnKF approximation is often more accurate than the GaBP, a fact intensely studied in the literature (Furrer & Bengtsson, 2007, e.g.), although once again few actionable analytic results are available Le Gland et al. (2009); Kelly et al. (2014). Empirically, EnKF is nonetheless frequently SOTA.

Heuristically, we argue that this is because the EnKF samples from the full joint distribution. This avoids accruing error via the Jensen gap (Eq. 43) or the Taylor expansion approximation (Eq. 45) used in the GaBP, which samples only the mean. Explicitly, the propagation of errors in GaBP produces approximations like the following

$$\widehat{\mathrm{Var}}^{\mathrm{GaBP}} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} \simeq \begin{bmatrix} \mathrm{Var}\, \mathbf{x}_1 & J(\boldsymbol{m}_1) \mathrm{Var}\, \mathbf{x}_1 \\ \mathrm{Var}\, \mathbf{x}_1 J(\boldsymbol{m}_1)^\top & J(\boldsymbol{m}_1) \mathrm{Var}\, \mathbf{x}_1 J^\top(\boldsymbol{m}_1) \end{bmatrix} \tag{78}$$

where $J(x)$ is the Jacobian matrix of $\mathcal{P}$ at x. In the EnKF, the joint variance is modeled by empirical samples,

$$\widehat{\mathrm{Var}}^{\mathrm{EnKF}} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} \simeq \widehat{\mathrm{Var}}_{\sigma^2 \mathrm{I}} \begin{bmatrix} x^{(1)} & \cdots & x^{(N)} \\ \mathcal{P}(x^{(1)}) & \cdots & \mathcal{P}\left(x^{(N)}\right) \end{bmatrix} \tag{79}$$

$$= \sigma^2 \mathrm{I} + \begin{bmatrix} x^{(1)} & \cdots & x^{(N)} \\ \mathcal{P}(x^{(1)}) & \cdots & \mathcal{P}\left(x^{(N)}\right) \end{bmatrix} \begin{bmatrix} x^{(1)} & \cdots & x^{(N)} \\ \mathcal{P}(x^{(1)}) & \cdots & \mathcal{P}\left(x^{(N)}\right) \end{bmatrix}^\top. \tag{80}$$

The relative quality of each of these updates is nontrivial to discover. Noting, however, that the EnKF estimates are frequently better in practice, and, as demonstrated at length in this paper, can be made computationally more efficient in high dimensions, we find out motivation for exploring the ~~generallisation~~ generalisation of the EnKF that drives the GEnBP algorithm.

E MATHERON UPDATES FOR GAUSSIAN VARIATES

The pathwise Gaussian process update (*Matheron update*), credited to Matheron by Wilson et al. (2021); Doucet (2010) is a method of simulating from a conditional of some jointly Gaussian variate. If

$$\begin{bmatrix}\mathbf{y}\\ \mathbf{w}\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}m_{\mathbf{y}}\\ m_{\mathbf{w}}\end{bmatrix}, \begin{bmatrix}\mathrm{K}_{\mathbf{yy}} & \mathrm{K}_{\mathbf{yw}}\\ \mathrm{K}_{\mathbf{wy}} & \mathrm{K}_{\mathbf{ww}}\end{bmatrix}\right) \tag{81}$$

then the moment of the $\mathbf{w}$-conditional distribution are

$$\mathbb{E}[\mathbf{y} \mid \mathbf{w}{=}\boldsymbol{w}^*] = m_{\mathbf{y}} + \mathrm{K}_{\mathbf{wy}}\mathrm{K}_{\mathbf{ww}}^{-1}(\boldsymbol{w}^* - m_{\mathbf{w}}) \qquad \text{first moment} \tag{82}$$

$$\mathrm{Var}[\mathbf{y} \mid \mathbf{w}{=}\boldsymbol{w}^*] = \mathrm{K}_{\mathbf{yy}} - \mathrm{K}_{\mathbf{wy}}\mathrm{K}_{\mathbf{ww}}^{-1}\mathrm{K}_{\mathbf{yw}}. \qquad \text{second moment} \tag{83}$$

Proposition 6. *For* $\begin{bmatrix}\mathbf{y}\\ \mathbf{w}\end{bmatrix}$ *as in Equation 81, the variates in the following mapping*

$$\mathbf{y}, \mathbf{w}, \boldsymbol{w}^* \mapsto \mathbf{y} + \mathrm{K}_{\mathbf{wy}}\mathrm{K}_{\mathbf{ww}}^{-1}(\boldsymbol{w}^* - \mathbf{w}) \tag{84}$$

satisfy the moment conditions (Eq. 82) and (Eq. 83) and thus $(\mathbf{y}+\mathrm{K}_{\mathbf{wy}}\mathrm{K}_{\mathbf{ww}}^{-1}(\boldsymbol{w}^*-\mathbf{w})) \stackrel{\mathrm{d}}{=} (\mathbf{y} \mid \mathbf{w}{=}\boldsymbol{w}^*)$.

Proof: Taking moments of Equation 84

$$\mathbb{E}\left[\mathbf{y} + \mathrm{K}_{\mathbf{wy}}\mathrm{K}_{\mathbf{ww}}^{-1}(\boldsymbol{w}^* - \mathbf{w})\right] = m_{\mathbf{y}} + \mathrm{K}_{\mathbf{wy}}\mathrm{K}_{\mathbf{ww}}^{-1}(\boldsymbol{w}^* - m_{\mathbf{w}})$$

$$\begin{aligned}
&\mathrm{Var}\left[\mathbf{y} + \mathrm{K}_{\mathbf{wy}}\mathrm{K}_{\mathbf{ww}}^{-1}(\boldsymbol{w}^* - \mathbf{w})\right]\\
&= \mathrm{Var}[\mathbf{y}] + \mathrm{Var}[\mathrm{K}_{\mathbf{wy}}\mathrm{K}_{\mathbf{ww}}^{-1}(\boldsymbol{w}^* - \mathbf{w})] + \mathrm{Var}(\mathbf{y}, \mathrm{K}_{\mathbf{wy}}\mathrm{K}_{\mathbf{ww}}^{-1}(\boldsymbol{w}^* - \mathbf{w})) + \mathrm{Var}(\mathbf{y}, \mathrm{K}_{\mathbf{wy}}\mathrm{K}_{\mathbf{ww}}^{-1}(\boldsymbol{w}^* - \mathbf{w}))^\top\\
&= \mathrm{K}_{\mathbf{yy}} + \mathrm{K}_{\mathbf{wy}}\mathrm{K}_{\mathbf{ww}}^{-1}K_{\mathbf{ww}}\mathrm{K}_{\mathbf{ww}}^{-1}\mathrm{K}_{\mathbf{yw}} - 2\mathrm{K}_{\mathbf{yw}}\mathrm{K}_{\mathbf{ww}}^{-1}\mathrm{K}_{\mathbf{wy}}\\
&= \mathrm{K}_{\mathbf{yy}} - \mathrm{K}_{\mathbf{wy}}\mathrm{K}_{\mathbf{ww}}^{-1}\mathrm{K}_{\mathbf{yw}}
\end{aligned}$$

we see that both first and second moment conditions are satisfied. □

Note that this update does not require us to calculate $\mathrm{K}_{\mathbf{yy}}$ and further, may be conducted without needing to evaluate the density of the observation.

F DIAGONAL MATRICES WITH LOW-RANK PERTURBATION

Suppose the $\mathrm{K} = \mathrm{V} + s\mathrm{L}\mathrm{L}^\top$ where L is a $D \times N$ matrix, I is $D \times D$ and $\mathrm{V} = \mathrm{diag}(\boldsymbol{v}$ and $s \in \{-1, 1\})$ is the *sign* of the matrix. We are primarily interested in such matrices in the case that $N \ll D$, which case we have called *Diagonal Matrix with Low-rank perturbation*, when their computational properties are favourable for our purposes. This is in contrast to matrices with no particular exploitable structure, which we refer to as *dense*.

Throughout this section we assume that the matrices in question are positive definite; and that all operations are between conformable operations. We refer to the matrix L as the *component* of the DLR matrix, and the diagonal matrix V as the *nugget* term, by analogy with classic kriging.

If $N \geq D$ the name is misleading since they are not truly *low*-rank; the identities we write here still hold, but are not computationally expedient.

F.1 MULTIPLICATION BY A DENSE MATRIX

The matrix product of an arbitrary matrix with a DLR matrix may be calculated efficiently by grouping operations, since $\mathrm{K}\mathrm{X} = \mathrm{V}\mathrm{X} \pm \mathrm{U}(\mathrm{U}^\top\mathrm{X})$, which has a time cost of $\mathcal{O}(ND^2)$ and which may be calculated without forming the full matrix K. The result is not in general also a DLR matrix.

F.2 ADDITION

The matrix sum of two DLR matrices with the same sign is also a DLR matrix. Suppose $\mathrm{K} = \mathrm{V} + s\mathrm{L}\mathrm{L}^\top, \mathrm{K}' = \mathrm{V}' + s\mathrm{L}'\mathrm{L}'^\top$,

$$\mathrm{K} + \mathrm{K}' = \mathrm{V} + \mathrm{V}' + s\mathrm{L}\mathrm{L}^\top + s\mathrm{L}'\mathrm{L}'^\top \tag{85}$$

$$= \mathrm{V} + \mathrm{V}' + s\,[\mathrm{L} \quad \mathrm{L}']\,[\mathrm{L} \quad \mathrm{L}']^\top. \tag{86}$$

The new matrix, is also DLR with has nugget term $\mathrm{V} + \mathrm{V}$ and component $[\mathrm{L} \quad \mathrm{L}']$.

F.3 DLR INVERSES OF DLR MATRICES

Inverses of DLR matrices are once again DLR matrices, and may be found efficiently.

Proposition 7. *Choose DLR* $\mathrm{K} = \mathrm{V} + \mathrm{L}\mathrm{L}^\top$*, i.e. with sign* s *positive. Then its inverse*

$$\mathrm{K}^{-1} = \mathrm{V}^{-1} - \mathrm{R}\mathrm{R}^\top \tag{87}$$

is also DLR, of the same component dimension, but with a negative sign, with

$$\mathrm{R} = \mathrm{V}^{-1}\mathrm{L}\,\mathrm{chol}\left(\left(\mathrm{I} + \mathrm{L}^\top\mathrm{V}^{-1}\mathrm{L}\right)^{-1}\right). \tag{88}$$

where $\mathrm{chol}(\mathrm{A})$ *denotes a decomposition* $\mathrm{chol}(\mathrm{A})\,\mathrm{chol}(\mathrm{A})^\top = \mathrm{A}$.

Proof: Using the Woodbury identity,

$$\begin{aligned}\mathrm{K}^{-1} &= \mathrm{V}^{-1} - \mathrm{V}^{-1}\mathrm{L}\left(\mathrm{I} + \mathrm{L}^\top\mathrm{V}^{-1}\mathrm{L}\right)^{-1}\mathrm{L}^\top\mathrm{V}^{-1} \\ &= \mathrm{V}^{-1} - \mathrm{R}\mathrm{R}^\top.\end{aligned}$$

□

Proposition 8. *Choose DLR* $\mathrm{P} = \mathrm{U} - \mathrm{R}\mathrm{R}^\top$*, i.e. with sign* s *negative. Then its inverse*

$$\mathrm{P}^{-1} = \mathrm{U}^{-1} - \mathrm{L}\mathrm{L}^\top \tag{89}$$

is also DLR, of the same component dimension, but with a positive sign, with

$$\mathrm{L} = \mathrm{U}^{-1}\mathrm{R}\,\mathrm{chol}\left(\left(-\mathrm{I} + \mathrm{R}^\top\mathrm{U}^{-1}\mathrm{R}\right)^{-1}\right). \tag{90}$$

Proof: Using the Woodbury identity,

$$\mathrm{P}^{-1} = \mathrm{U}^{-1} - \mathrm{U}^{-1}\mathrm{R}\left(-\mathrm{I} + \mathrm{R}^\top\mathrm{U}^{-1}\mathrm{R}\right)^{-1}\mathrm{R}^\top\mathrm{U}^{-1} \tag{91}$$

$$= \mathrm{U}^{-1} + \mathrm{L}\mathrm{L}^\top \tag{92}$$

□

We have introduced $\mathrm{chol}(\mathrm{A})$ to construct Cholesky factors. The cost of the inversion is the same for both, $\mathcal{O}\left(N^2D + N^3\right)$ — $\mathcal{O}(N^3)$ for the construction of the Cholesky factor, and $\mathcal{O}(DN^2)$ for the requisite matrix multiplies. The space cost is $\mathcal{O}\left(ND\right)$.

For a given $\mathrm{K} = \mathrm{V} + s\mathrm{L}\mathrm{L}^\top$, the term $\left(s\mathrm{I} + \mathrm{L}^\top\mathrm{V}^{-1}\mathrm{L}\right)$ is referred to as the *capacitance* of the matrix by convention.

F.4 EXACT INVERSION WHEN THE COMPONENT IS HIGH-RANK

Suppose $\mathrm{P} = \mathrm{U} - \mathrm{R}\mathrm{R}^\top$ where the $D \times N$ components high rank, in the sense that $N > D$. Then the low rank inversion to calculate P^{-1} is no longer cheap, since the $\mathcal{O}(N^D)$ cost is greater than naive inversion of the dense matrix, at $\mathcal{O}(D^3)$. In this case, it is cheaper to recover a DLR inverse by an alternate method. Note that $(\mathrm{U} - \mathrm{P})^{-1} = \mathrm{U}^{-1} - \mathrm{U}^{-1}(\mathrm{U}^{-1} - \mathrm{P}^{-1})^{-1}\mathrm{U}^{-1}$. We find the eigendecomposition

$$\mathrm{P}^{-1} = \mathrm{U}^{-1} + \mathrm{L}\mathrm{L}^\top \tag{93}$$

for some L. Such an L is given by $\mathrm{L} = \mathrm{Q}\Lambda^{1/2}$ for $\mathrm{Q}\Lambda\mathrm{Q}^\top = \mathrm{P}^{-1} - \mathrm{U}^{-1}$ a spectral decomposition. In the case that the diagonal U is constant this may be found more efficiently.

If, instead, we wish to invert $\mathrm{K} = \mathrm{V} + \mathrm{L}\mathrm{L}^\top$ we need to find $(-\mathrm{V} + \mathrm{K})^{-1} = -\mathrm{V}^{-1} - \mathrm{V}^{-1}(\mathrm{K}^{-1} - \mathrm{V}^{-1})^{-1}\mathrm{V}^{-1}$, so we decompose instead $\mathrm{Q}\Lambda\mathrm{Q} = \mathrm{V}^{-1}(\mathrm{K}^{-1} - \mathrm{V}^{-1})^{-1}\mathrm{V}^{-1}$ and define $\mathrm{R} = \mathrm{Q}\Lambda^{-1/2}$. Then the DLR form of the inverse is $-\mathrm{V}^{-1} - \mathrm{R}\mathrm{R}^\top$.

F.5 REDUCING COMPONENT RANK

We use the SVD to efficiently reduce the rank of the components in the DLR matrix in the sense of finding a Frobenius-distance-minimising approximation.

Suppose $\mathrm{K} = \mathrm{V} + \mathrm{L}\mathrm{L}^\top$ where L is a $D \times M$ matrix, I is $D \times D$ and $\mathrm{V} = \mathrm{diag}(\boldsymbol{v})$. Let $\mathrm{Y}\mathrm{S}\mathrm{Z}^\top$ be the "thin" SVD of L, i.e. $\mathrm{Y} \in \mathbb{R}^{D\times M}, \mathrm{S} \in \mathbb{R}^{M\times M}, \mathrm{Z} \in \mathbb{R}^{M\times M}$ with $\mathrm{Y}^\top\mathrm{Y} = \mathrm{I}_N, \mathrm{Z}^\top\mathrm{Z} = \mathrm{I}_N$, and S diagonal with non-negative entries. Then

$$\mathrm{L}\mathrm{L}^\top = \mathrm{Y}\mathrm{S}\mathrm{Z}^\top\mathrm{Z}\mathrm{S}\mathrm{Y}^\top = \mathrm{Y}\mathrm{S}^2\mathrm{Y}^\top.$$

First we note that should any singular values in S be zero, we may remove the corresponding columns of Y and Z without changing the product, so their exclusion is exact. Next, we note that if we choose the largest S singular values of S, setting the rest to zero, we obtain a Frobenius approximation of $\mathrm{L}\mathrm{L}^\top$ of rank S.

An SVD that captures the top N singular values may be found by randomised methods (Halko et al., 2010, 6.1) at a cost of $\mathcal{O}(ND\log N + N^2(D+N))$.

G GAUSSIANS WITH DLR PARAMETERS

We recall the forms of the Gaussian density introduced in section Section 2.2.3, in *moments* form using the mean $\boldsymbol{m}$ and covariance matrix K,

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{m}, \mathrm{K}) = \mathcal{N}(\mathbb{E}[\mathbf{x}], \mathrm{Var}(\mathbf{x})).$$

and the *canonical* form,

$$\mathbf{x} \sim \mathcal{N}_C(\boldsymbol{n}, \mathrm{P}) = \mathcal{N}_C(\mathrm{Var}^{-1}(\mathbf{x})\mathbb{E}[\mathbf{x}], \mathrm{Var}^{-1}(\mathbf{x})).$$

which uses the information vector $\boldsymbol{n}$ and the *precision matrix*, P with $\boldsymbol{n} = \mathrm{K}^{-1}\boldsymbol{m}, \quad \mathrm{P} = \mathrm{K}^{-1}$. The (equivalent) densities induced by these parameterisations are

$$f(\boldsymbol{x}) \propto \frac{1}{2}(\boldsymbol{x}-\boldsymbol{m})^\top\mathrm{K}^{-1}(\boldsymbol{x}-\boldsymbol{m}) = \frac{1}{2}\boldsymbol{x}^\top\mathrm{P}\boldsymbol{x} - \boldsymbol{n}^\top\boldsymbol{x}$$

We associate a given Gaussian ensemble with a moments-form Gaussian in the natural way,

$$\mathrm{X}_{\sigma^2} \sim \mathcal{N}(\boldsymbol{m}, \mathrm{K}) = \mathcal{N}(\widehat{\mathbb{E}}[\mathrm{X}], \widehat{\mathrm{Var}}_{\sigma^2}(\mathrm{X}))$$

introducing a parameter σ^2 which we use to ensure invertibility of the covariance if needed.

We associate with each moments-form Gaussian a *canonical form* which may be cheaply calculated by using the by using the low-rank representation of the covariance by Proposition 7,

$$\begin{aligned}
\mathrm{X}_{\sigma^2} &\sim \mathcal{N}(\boldsymbol{m}, \mathrm{K}) \\
\boldsymbol{m} &= \widehat{\mathbb{E}}(\mathrm{X}) = \overline{\mathrm{X}} \\
\mathrm{K} &= \widehat{\mathrm{Var}}_{\sigma^2}(\mathrm{X}) = \sigma^2 + \breve{\mathrm{X}}\breve{\mathrm{X}}^\top \\
&\Leftrightarrow \\
\mathrm{X}_{\sigma^2} &\sim \mathcal{N}_C(\boldsymbol{n}, \mathrm{P}) \\
\boldsymbol{n} &= \widehat{\mathrm{Var}}_{\sigma^2}^{-1}(\mathrm{X})\widehat{\mathbb{E}}(\mathrm{X}) = \mathrm{P}\overline{\mathrm{X}} \\
\mathrm{P} &= \widehat{\mathrm{Var}}_{\sigma^2}^{-1}(\mathrm{X}) = \sigma^{-2}\mathrm{I} - \mathrm{R}\mathrm{R}^\top.
\end{aligned}$$

where we introduced $\mathrm{R} = \breve{\mathrm{X}}\,\mathrm{chol}\left(\left(\mathrm{I} + \sigma^{-2}\breve{\mathrm{X}}^\top\breve{\mathrm{X}}\right)^{-1}\right)$.

We may recover $\boldsymbol{m}$ and K from $\boldsymbol{n}$ and P by

$$\begin{aligned}
\mathrm{K} &= \mathrm{P}^{-1} \\
\boldsymbol{m} &= \mathrm{P}^{-1}\boldsymbol{n}.
\end{aligned}$$

this time using the alternative low rank inverse formula, Equation 8.

H GENBP DETAILS

H.1 GENBP ALGORITHM

Here we expand the steps of the GEnBP algorithm, itemising the matrix operations needed to conduct the Gaussian updates while mataining the DLR forms for the matrix parameters.

Algorithm 2: GEnBP

Require: Graph $\mathcal{G}$, set of generative processes $\{\mathcal{P}_j\}_j$, observations $\boldsymbol{x}_{\mathscr{E}}$, ancestral sample $\mathrm{X}_{\mathscr{A}}$.
Ensure: Observation-conditional sample $\mathrm{X}^*_{\mathscr{Q}}$.

1: **while** not converged **do**
2: Sample ensemble from the generative processes $\mathcal{P}_j$ on $\mathcal{G}$ using (Eq. 3)
3: Convert $\mathcal{G}$ into the conditional graph $\mathcal{G}^*$ by conditioning observed factors (Eq. 8) using (Eq. 17)
4: Convert variables and factors to DLR canonical form {Section 3.1}
5: Propagate DLR messages on $\mathcal{G}^*$ {Section H.2/Algorithm 3}
6: Conform ancestral nodes to belief $T(\mathrm{X}^*_{\mathscr{Q}}) \sim b_{\mathcal{G}^*}(\mathsf{x}_{\mathscr{A}})$ {Section 3.3}
7: **end while**
8: Return Approximate posterior sample $\mathrm{X}^*_{\mathscr{Q}}$.

H.2 GENBP FACTOR-TO-VARIABLE MESSAGE

We outline the steps in a GEnBP $f_j \to \mathsf{x}_k$ factor-to-variable message with a single incoming message $\mathsf{x}_\ell \to f_j$. The message is Gaussian with canonical parameters $\boldsymbol{n}, \mathrm{P}$ and the incoming message is Gaussian with canonical parameters $\boldsymbol{n}', \mathrm{P}'$. The message is updated to $\boldsymbol{n}'', \mathrm{P}''$ and passed to the variable node. Extending this to multiple incoming messages is straightforward by iteration.

Algorithm 3: GEnBP $f_j \to \mathsf{x}_\ell$ factor-to-variable message with a single incoming $\mathsf{x}_k \to f_j$

Require: Factor f_j canonical parameters:

- Information vector: $\boldsymbol{n} = \begin{bmatrix} \boldsymbol{n}_\ell \\ \boldsymbol{n}_k \end{bmatrix}$
- Precision matrix in DLR form: $\mathrm{P} = \mathrm{U} - \mathrm{R}\mathrm{R}^\top$, where:
  - $\mathrm{U} = \mathrm{diag}\left(\begin{bmatrix} \boldsymbol{u}_\ell \\ \boldsymbol{u}_k \end{bmatrix}\right)$
  - $\mathrm{R} = \begin{bmatrix} \mathrm{R}_\ell \\ \mathrm{R}_k \end{bmatrix}$
  - $\mathrm{R}_\ell \in \mathbb{R}^{D_\ell \times N}$, $\mathrm{R}_k \in \mathbb{R}^{D_k \times N}$

Require: Incoming message $\mathsf{x}_k \to f_j$ canonical parameters:

- Information vector: $\boldsymbol{n}'_k \in \mathbb{R}^{D_k}$
- Precision matrix in DLR form: $\mathrm{P}'_k = \mathrm{U}'_k - \mathrm{R}'_k {\mathrm{R}'_k}^\top$, where:
  - $\mathrm{U}'_k = \mathrm{diag}\left(\boldsymbol{u}'_k\right)$
  - $\mathrm{R}'_k \in \mathbb{R}^{D_k \times N'}$

Ensure: Outgoing message $f_j \to \mathsf{x}_\ell$ canonical parameters:

- Information vector: $\boldsymbol{n}'_\ell$
- Precision matrix in DLR form: $\mathrm{P}'_\ell = \mathrm{U}'_\ell - \mathrm{R}'_\ell {\mathrm{R}'_\ell}^\top$

1: Combine information vectors: $\tilde{\boldsymbol{n}}_k \leftarrow \boldsymbol{n}_k + \boldsymbol{n}'_k$
2: Combine precision diagonals: $\tilde{\boldsymbol{u}}_k \leftarrow \boldsymbol{u}_k + \boldsymbol{u}'_k$
3: Concatenate low-rank components: $\tilde{\mathrm{R}}_k \leftarrow [\mathrm{R}_k \quad \mathrm{R}'_k]$

4: Form joint information vector: $\boldsymbol{n}_\Pi \leftarrow \begin{bmatrix} \boldsymbol{n}_\ell \\ \tilde{\boldsymbol{n}}_k \end{bmatrix}$
5: Form joint precision diagonal: $\mathrm{U}_\Pi \leftarrow \mathrm{diag}\left(\begin{bmatrix} \boldsymbol{u}_\ell \\ \tilde{\boldsymbol{u}}_k \end{bmatrix}\right)$
6: Form joint low-rank component: $\mathrm{R}_\Pi \leftarrow \begin{bmatrix} \mathrm{R}_\ell & 0 \\ \mathrm{R}_k & \mathrm{R}'_k \end{bmatrix}$
7: Form joint precision matrix: $\mathrm{P}_\Pi \leftarrow \mathrm{U}_\Pi - \mathrm{R}_\Pi \mathrm{R}_\Pi^\top$
8: Compute joint covariance: $\mathrm{K}_\Pi \leftarrow \mathrm{P}_\Pi^{-1} = \mathrm{V}_\Pi + \mathrm{L}_\Pi \mathrm{L}_\Pi^\top$
9: Compute joint mean: $\boldsymbol{m}_\Pi \leftarrow \mathrm{K}_\Pi \boldsymbol{n}_\Pi$
10: Extract marginal mean for $\mathbf{x}_\ell$: $\boldsymbol{m}_\ell \leftarrow \boldsymbol{m}_\Pi[1{:}D_\ell]$
11: Extract marginal covariance for $\mathbf{x}_\ell$:
- Diagonal component: $\mathrm{V}_\ell \leftarrow \mathrm{V}_\Pi[1{:}D_\ell, 1{:}D_\ell]$
- Low-rank component: $\mathrm{L}_\ell \leftarrow \mathrm{L}_\Pi[1{:}D_\ell, :]$
- Marginal covariance: $\mathrm{K}_\ell \leftarrow \mathrm{V}_\ell + \mathrm{L}_\ell \mathrm{L}_\ell^\top$

12: Reduce rank of K_ℓ to N:
- SVD: $\mathrm{L}_\ell = \mathrm{ASB}^\top$
- Top N components $\mathrm{L}'_\ell = \mathrm{A}_{[:,1:N]}\mathrm{S}_{[1:N,1:N]}$
- Updated covariance: $\mathrm{K}_\ell \leftarrow \mathrm{V}_\ell + \mathrm{L}'_\ell \mathrm{L}'^\top_\ell$

13: Compute marginal precision: $\mathrm{P}'_\ell \leftarrow \mathrm{K}_\ell^{-1} = \mathrm{U}'_\ell - \mathrm{R}'_\ell \mathrm{R}'^\top_\ell$
14: Compute marginal information vector: $\boldsymbol{n}'_\ell \leftarrow \mathrm{P}'_\ell \boldsymbol{m}_\ell$
15: **return** $\boldsymbol{n}'_\ell, \mathrm{P}'_\ell$

H.3 Ensemble recovery

Suppose that after belief propagation we update our belief about a given query variable node to $b(\mathbf{x}_{\mathscr{Q}}) \sim \mathcal{N}_M(\boldsymbol{x}_{\mathscr{Q}}; \boldsymbol{m}, \mathrm{K}_{\mathscr{Q}})$. In the Ensemble message passing setting we have DLR $\mathrm{K}_{\mathscr{Q}} = \mathrm{V}_{\mathscr{Q}} + \mathrm{L}_{\mathscr{Q}}\mathrm{L}_{\mathscr{Q}}^\top$. In order to convert this belief into ensemble samples, we choose the transformation T which maps prior ensemble $\mathrm{X}_{\mathscr{Q}}$ from the previous iterate to an updated ensemble $\mathrm{X}'_{\mathscr{Q}} = T(\mathrm{X}_{\mathscr{Q}})$ such that the (empirical) ensemble distribution is as similar as possible, in some metric d to $\mathcal{N}_M(\boldsymbol{x}_{\mathscr{Q}}; \boldsymbol{m}, \mathrm{K}_{\mathscr{Q}})$, Hereafter we suppress the subscript Q for compactness, and because what follows is generic for ensemble moment matching. This amounts to choosing

$$T = \arg\min_{T^*} d(\mathcal{N}_M(\cdot, \widehat{\mathbb{E}}T(\mathrm{X}), \widehat{\mathrm{Var}}_{\sigma^2}T(\mathrm{X})), \mathcal{N}_M(\cdot; \boldsymbol{m}, \mathrm{K})), \tag{94}$$

We wish to do this *without forming* $\mathrm{K} \in \mathbb{R}^{D\times D}$, which may be prohibitively memory expensive, by exploiting its DLR decomposition. We restrict ourselves to the family of affine transformations $T_{\boldsymbol{\mu},\mathrm{T}} : \mathrm{X} \mapsto \boldsymbol{\mu}\mathrm{B} + \breve{\mathrm{X}}\mathrm{T}$ where the parameters $\boldsymbol{\mu} \in \mathbb{R}^D, \mathrm{T} \in \mathbb{R}^{N\times N}$ must be chosen. If we minimise between the belief and ensemble distributions empirical covariances, the solution is

$$\boldsymbol{\mu} = \widehat{\mathbb{E}}\mathrm{X} \tag{95}$$

$$\mathrm{T} = \arg\min_{\mathrm{T}} \left\| \widehat{\mathrm{Var}}_{\sigma^2}(\breve{\mathrm{X}}\mathrm{T}) - \mathrm{K} \right\|_F^2 \tag{96}$$

$$= \arg\min_{\mathrm{T}} \left\| \widehat{\mathrm{Var}}_{\sigma^2}(T(\mathrm{X})) - \mathrm{LL}^\top - \mathrm{V} \right\|_F^2 \tag{97}$$

$$= \arg\min_{\mathrm{T}} \left\| \frac{\breve{\mathrm{X}}\mathrm{TT}^\top\breve{\mathrm{X}}^\top}{N-1} - \mathrm{V}_{-\sigma^2} - \mathrm{LL}^\top \right\|_F^2, \tag{98}$$

Here we have introduced $\mathrm{V}_{-\sigma^2} \coloneqq \mathrm{V} - \sigma^2\mathrm{I}$. Note that minimisers of L with respect to T are not unique, because L depends only on $\mathrm{TT}^\top$. For example, we may take any orthogonal transformation U of T and obtain the same $(\mathrm{TU})(\mathrm{TU})^\top = \mathrm{TUU}^\top\mathrm{T}^\top = \mathrm{TT}^\top$. We instead optimise $\mathrm{M} = \mathrm{TT}^\top$ over the space of $N \times N$ PSD matrices $\mathbb{M}_+^N = \{\mathrm{M} \in \mathbb{R}^{N\times N} \mid \mathrm{M} = \mathrm{M}^\top, \mathrm{M} \succeq 0\}$. That is, we consider the problem

$$\mathrm{M}^* = \arg\min_{\mathrm{M}\in\mathbb{M}_+^N} L(\mathrm{M}) \tag{99}$$

where

$$L(\mathrm{M}) \coloneqq \arg\min_{\mathrm{M}} \left\| \frac{1}{N-1} \breve{\mathrm{X}} \mathrm{M} \breve{\mathrm{X}}^\top - \mathrm{L}\mathrm{L}^\top - \mathrm{V}_{-\sigma^2} \right\|_F^2. \tag{100}$$

This is a convex problem in M. The derivatives are given by (Laue et al., 2018; 2020)

$$\begin{aligned} \nabla_{\mathrm{M}} L &= \frac{2}{N-1} \breve{\mathrm{X}}^\top \left(\frac{\breve{\mathrm{X}} \mathrm{M} \breve{\mathrm{X}}^\top}{N-1} - \mathrm{L}\mathrm{L}^\top - \mathrm{V}_{-\sigma^2} \right) \breve{\mathrm{X}} \\ &= \frac{2}{N-1} \left(\frac{\breve{\mathrm{X}}^\top \breve{\mathrm{X}} \mathrm{M} \breve{\mathrm{X}}^\top \breve{\mathrm{X}}}{N-1} - (\breve{\mathrm{X}}^\top \mathrm{L})(\breve{\mathrm{X}}^\top \mathrm{L})^\top - \breve{\mathrm{X}}^\top \mathrm{V}_{-\sigma^2} \breve{\mathrm{X}} \right) \end{aligned}$$

An unconstrained solution in M' may be found by setting the gradient to zero,

$$\breve{\mathrm{X}}^\top \breve{\mathrm{X}} \mathrm{M}' \breve{\mathrm{X}}^\top \breve{\mathrm{X}} = (N-1) \left(\breve{\mathrm{X}}^\top \mathrm{L} (\breve{\mathrm{X}}^\top \mathrm{L})^\top + \breve{\mathrm{X}}^\top \mathrm{V}_{-\sigma^2} \breve{\mathrm{X}} \right).$$

This linear system may be solved at $\mathcal{O}(N^3)$ cost for the ensemble size N. As $\breve{\mathrm{X}}^\top \breve{\mathrm{X}}$ is Hermitian we also economise by using specialised methods such as pivoted LDL decomposition. Using the same decomposition we also find the required $\mathrm{M}^* = \mathrm{T}\mathrm{T}^\top$, choosing $\mathrm{T} = \mathrm{U}(\mathrm{S}^+)^{1/2}$. By careful ordering of operations we may calculate the RHS with cost $\mathcal{O}(DM^2)$ for a total cost of $\mathcal{O}(N^3 + DN^2 + DM^2)$.

I ALTERNATIVE LINEARISATIONS AND LOW RANK DECOMPOSITIONS

In response to a reader question, we investigate whether GEnBP is "just" a low-rank decomposition of the GaBP algorithm. We argue that it is not. Rather the relationship is that diagrammed by Figure 13. Although the GEnBP and GaBP algorithms both use Gaussian approximations, they do not use the

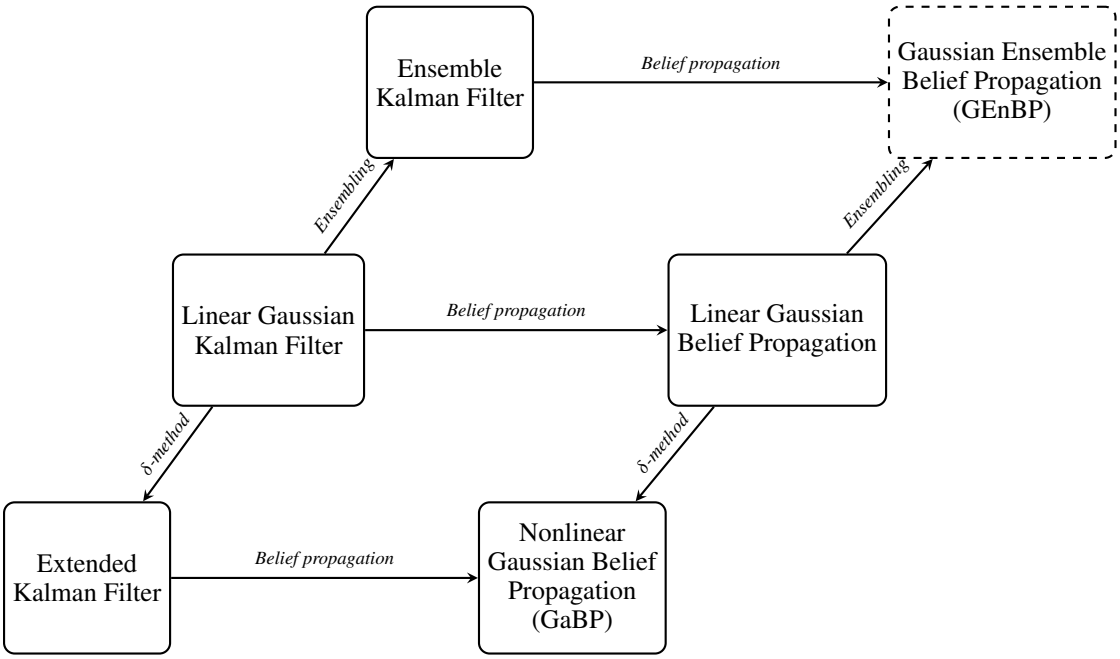


Figure 13: Relationship between GEnBP and GaBP.

same Gaussian approximations. This is why we observe in Section 4 that GEnBP is able to surpass GaBP in accuracy and not just speed. We noted in Appendix C.3 that GaBP include not only a choice of Gaussian density family, which it shares with GEnBP, but also a particular means of approximating nonlinear factors which it does not. That apparently minor difference has large implications. We expand upon some of the differences and implications in this section.

Consider how the joint density of a factor $\mathbf{x}_2 = \mathcal{P}(\mathbf{x}_1)$ to make the differences explicit. For the sake of simplicity, suppose $D_1 = D_2 = D$. Two alternatives for covariance estimation have been discussed in this work: GaBP uses propagation of errors,

$$\operatorname{Var}\begin{bmatrix}\mathbf{x}_1\\ \mathbf{x}_2\end{bmatrix} \simeq \begin{bmatrix}\operatorname{Var}\mathbf{x}_1 & J(\boldsymbol{m}_1)\operatorname{Var}\mathbf{x}_1\\ \operatorname{Var}\mathbf{x}_1 J(\boldsymbol{m}_1)^\top & J(\boldsymbol{m}_1)\operatorname{Var}\mathbf{x}_1 J^\top(\boldsymbol{m}_1)\end{bmatrix} \tag{101}$$

where $J(x)$ is the Jacobian matrix of $\mathcal{P}$ at x. GEnBP, by contrast, uses the empirical estimate

$$\operatorname{Var}\begin{bmatrix}\mathbf{x}_1\\ \mathbf{x}_2\end{bmatrix} \simeq \widehat{\operatorname{Var}}_{\sigma^2\mathrm{I}}\begin{bmatrix}x^{(1)} & \cdots & x^{(N)}\\ \mathcal{P}(x^{(1)}) & \cdots & \mathcal{P}\left(x^{(N)}\right)\end{bmatrix}. \tag{102}$$

As noted in Section 3.4, the costs of Equation 102 is greater than Equation 101 both in time and space. There are two components to this cost:

Firstly, the cost of generating the ensemble (in GenBP) and the Jacobians (in GaBP). In GEnBP, we need to generate N samples from the prior distribution, so the cost here scales as N in general. The cost of calculating the Jacobian for the GaBP depends on the function but in general is $\mathcal{O}(D)$.

Secondly, taking the matrix products in each of these covariance matrices. The matrix multiplication in Equation 101 is $\mathcal{O}(D^3)$, since it involved the product of three $D \times D$ matrices. The empirical covariance calculation in Equation 102 is $\mathcal{O}(DN)$. We note that it *would* be $\mathcal{O}(D^2N)$ if we were to calculate the full covariance matrix in GEnBP, but the central lesson of this paper is that we never need to calculate that product, and it suffices to calculate the deviance matrices.

It seems that at this stage, GEnBP dominates when $N < D$. We note that in the subsequent belief propagation the story is more complicated but that GEnBP has generally cheaper belief propagation steps (except where node degree is high). We might ask if GaBP can also benefit from these low-rank belief updates.

Suppose we wished to construct a 3rd option, a *low rank GaPB* (LRGaBP) which used a low-rank decomposition of the covariance matrix to conduct approximate GaBP. First, we would find rank N decomposition of the prior covariance $\operatorname{Var}\mathbf{x}_1 \approx \mathrm{L}\mathrm{L}^\top$, with $\mathrm{L} \in \mathbb{R}^{D\times N}$, say by eigendecomposition, which is naively a D^3 operation, or $\mathcal{O}(D^2 \log N + N^2 D + N^3)$ by the method of (Halko et al., 2010, 6.1). The joint variance arising from the propagation-of-errors/δ-method (Dorfman, 1938) is

$$\operatorname{Var}\begin{bmatrix}\mathbf{x}_1\\ \mathbf{x}_2\end{bmatrix} \simeq \begin{bmatrix}\operatorname{Var}\mathbf{x}_1 & J(\boldsymbol{m}_1)\operatorname{Var}\mathbf{x}_1\\ \operatorname{Var}\mathbf{x}_1 J(\boldsymbol{m}_1)^\top & J(\boldsymbol{m}_1)\operatorname{Var}\mathbf{x}_1 J^\top(\boldsymbol{m}_1)\end{bmatrix} \tag{103}$$

$$= \begin{bmatrix}\mathrm{L}\mathrm{L}^\top & J(\boldsymbol{m}_1)\mathrm{L}\mathrm{L}^\top\\ \mathrm{L}\mathrm{L}^\top J(\boldsymbol{m}_1)^\top & J(\boldsymbol{m}_1)\mathrm{L}\mathrm{L}^\top J^\top(\boldsymbol{m}_1)\end{bmatrix} \tag{104}$$

$$= \begin{bmatrix}\mathrm{L}\\ J(\boldsymbol{m}_1)\mathrm{L}\end{bmatrix}\begin{bmatrix}\mathrm{L}\\ J(\boldsymbol{m}_1)\mathrm{L}\end{bmatrix}^\top \tag{105}$$

This is indeed a low-rank decomposition, amenable to similar tricks as the other low-rank tricks outlined in Appendix F. However, it is not computationally competitive. J is $D \times D$ so the product costs $\mathcal{O}(D^2N)$, plus the $\mathcal{O}(D)$ cost of calculating the Jacobian. We summarise the costs of this hypothetical LRGaBP step in Table 3. Notably, while the precise relationship between the algorithms depends upon the exact problem structure, we should generally expect GEnBP to be more efficient than LRGaBP for high dimensional problems for a fixed N, since GEnBP is never worse than LRGaBP, and in many cases has a lower exponent in D.

Further, since LRGaBP is an approximation to GaBP, the accuracy of LRGaBP to be bounded above by the accuracy of GaBP. GEnBP, as a different approximation to the target estimand, has no such restriction.

Table 3: Computational Costs for Gaussian Belief Propagation, Ensemble Belief propagation, and the hypothetical Low Rank Gaussian Belief Propagation for node dimension D, ensemble size/component rank N.

Operation		GaBP	GEnBP	LRGaBP
Time	Simulation	$\mathcal{O}(1)$	$\mathcal{O}(N)$	$\mathcal{O}(1)$
	Error propagation	$\mathcal{O}(D^3)$	—	$\mathcal{O}(D^2N)$
	Jacobian calculation	$\mathcal{O}(D)$	—	$\mathcal{O}(D)$
	Covariance SVD	—	$\mathcal{O}(N^3 + N^2D)$	$\mathcal{O}(D^2 \log N + N^3 + N^2D)$
Space	Covariance Matrix	$\mathcal{O}(D^2)$	$\mathcal{O}(ND)$	$\mathcal{O}(ND + D^3)$
	Precision Matrix	$\mathcal{O}(D^2)$	$\mathcal{O}(ND)$	$\mathcal{O}(ND)$