Extended Variational Message Passing for Automated Approximate Bayesian Inference

entropy

MDPI

Article

Extended Variational Message Passing for Automated Approximate Bayesian Inference

Semih Akbayrak [1,*], Ivan Bocharov [1,†] and Bert de Vries [1,2]

[1] Department of Electrical Engineering, Eindhoven University of Technology, P.O. Box 513, 5600MB Eindhoven, The Netherlands; i.a.bocharov@tue.nl (I.B.); Bert.de.Vries@tue.nl (B.d.V.)
[2] GN Hearing BV, JF Kennedylaan 2, 5612AB Eindhoven, The Netherlands
* Correspondence: s.akbayrak@tue.nl
† Current address: Department of Radiology and Nuclear Medicine, Erasmus MC, P.O. Box 2040, 3000CA Rotterdam, The Netherlands.

Abstract: Variational Message Passing (VMP) provides an automatable and efficient algorithmic framework for approximating Bayesian inference in factorized probabilistic models that consist of conjugate exponential family distributions. The automation of Bayesian inference tasks is very important since many data processing problems can be formulated as inference tasks on a generative probabilistic model. However, accurate generative models may also contain deterministic and possibly nonlinear variable mappings and non-conjugate factor pairs that complicate the automatic execution of the VMP algorithm. In this paper, we show that executing VMP in complex models relies on the ability to compute the expectations of the statistics of hidden variables. We extend the applicability of VMP by approximating the required expectation quantities in appropriate cases by importance sampling and Laplace approximation. As a result, the proposed Extended VMP (EVMP) approach supports automated efficient inference for a very wide range of probabilistic model specifications. We implemented EVMP in the Julia language in the probabilistic programming package *ForneyLab.jl* and show by a number of examples that EVMP renders an almost universal inference engine for factorized probabilistic models.

Keywords: Bayesian inference; variational inference; factor graphs; variational message passing; probabilistic programming

Citation: Akbayrak, S.; Bocharov, I.; de Vries, B. Extended Variational Message Passing for Automated Approximate Bayesian Inference. *Entropy* **2021**, *23*, 815. https://doi.org/10.3390/e23070815

Academic Editors: Antonio Salmerón and Rafael Rumí

Received: 18 May 2021
Accepted: 23 June 2021
Published: 26 June 2021

1. Introduction

Probabilistic Programming Languages (PPL) and packages [1] have gained strong popularity over recent years since they support fast algorithm development through automating Bayesian inference in probabilistic models. Many of these PPLs [2–5] are based on numerical approximation methods, which leads to inexact inference results, even if the model comprises conjugate factor pairs and exact inference is achievable. Moreover, although a majority of popular PPLs scale well to processing large data sets due to their stochastic inference settings [6], they tend to execute very slowly for certain types of structured dynamic models, such as state space models. Alternatively, some PPLs that execute inference by message passing in a factor graph [7,8] provide efficient inference performance by exploiting factorization and conjugacy between exponential family-based distribution pairs in the model. In particular the Variational Message Passing (VMP) [9,10] algorithm has gained a good reputation, as it supports efficient inference for conjugate factor pairs in factorized probabilistic models. Unfortunately, non-conjugate factor pairs complicate the automated estimation of posterior distributions, due to intractability of the normalization constants. Likewise, non-linear deterministic relations between model variables often create non-conjugate pairings and thus obstruct the message-passing-based inference mechanism.

This paper proposes an Extended VMP (EVMP) algorithm to support automated efficient inference on a wide class of models that contain both non-conjugate relations between factor pairs and deterministic, possibly non-linear factor nodes. In our solution proposal, the regular VMP algorithm constructs the functional forms of the messages. These functional forms contain expectations of functions of hidden variables. In the case that these expectation quantities cannot be evaluated to a closed-form expression, we estimate them by Importance Sampling (IS) [11], which is a well-known Monte Carlo method that approximates intractable posteriors by a set of weighted samples and estimates expectations over this sample set. We also make use of Laplace approximation ([12], Section 4.4) with support by automatic differentiation tools (autodiff) [13] in appropriate cases to approximate posteriors by normal distributions, which allows us to calculate the expectations over the approximating normal distribution. Our proposal leads to an efficient, automatable message-passing framework that removes most model specification limitations.

In Section 2, we start with a review of factor graphs and the VMP algorithm. Next, we specify the proposed Extended VMP algorithm in Section 3. In order to keep the paper readable, both for the advanced researcher and someone who just needs the results, we defer detailed discussions and derivations of the key equations in EVMP to Appendices A and B. We implemented EVMP in the Julia package *ForneyLab.jl* [8,14]. In Section 4 we present several comparative experiments of EVMP in ForneyLab vs. *Turing.jl*, which is an alternative state-of-the-art Julia-based PPL that focuses on Monte Carlo methods for inference. We show that EVMP transforms ForneyLab into an almost universally applicable inference engine, while retaining computational efficiency, due to its library of closed-form message passing rules. An extensive comparison to related work is presented in Section 5.

2. Problem Statement

Variational Message Passing on Forney-Style Factor Graphs

We assume a probabilistic model $p(y, z)$ with a given set of observations $y = y_{1:N} = \{y_1, \ldots, y_N\}$ and a set of latent variables $z = z_{1:M} = \{z_1, \ldots, z_M\}$. Bayesian inference in this model relates to evaluating the following posterior:

$$p(z|y) = \frac{p(y,z)}{\int p(y,z)\mathrm{d}z},$$

which relies on evaluating the model evidence $\int p(y,z)\mathrm{d}z$. Unfortunately, due to the computational complexity of evaluating the integral for the evidence, exact Bayesian inference is achievable only for a limited set of probabilistic models. Alternatively, inference can be executed by minimization of a variational objective called free energy.

$$\begin{aligned}\mathcal{F}[q] &= \mathbb{E}_q\left[\log \frac{q(z)}{p(y,z)}\right] \\ &= D_{KL}[q(z)||p(z|y)] - \log p(y),\end{aligned} \tag{1}$$

where $\mathbb{E}_q[\cdot]$ stands for the expectation with respect to $q(z)$ and $D_{KL}[\cdot||\cdot]$ denotes a Kullback–Leibler divergence. (In this paper, we denote the expected value $\int q(x)f(x)\mathrm{d}x$ of function f with respect to distribution q both by $E_q[f(x)]$ and $\langle f(x)\rangle_q$.) The KL divergence is greater or equal to zero for any distribution $q(z)$ and $D_{KL}[q(z)||p(z|y)] = 0$ if and only if $q(z) = p(z|y)$. As a result, minimizing the free energy with respect to q leads both to an approximate posterior

$$q^*(z) = \arg\max_q \mathcal{F}[q] \approx p(z|y)$$

and an upper bound $\mathcal{F}[q^*]$ on the negative log-evidence. In practice, minimization of $\mathcal{F}$ is often greatly alleviated by assuming a mean-field constraint, i.e., a fully factorized posterior $q(z_{1:M}) = \prod_{i=1}^{M} q(z_i)$.

Variational inference on factorized models $p(y, z)$ with the mean-field assumption for q leads to an automatable algorithm called Variational Message Passing (VMP) [9,10]. VMP can be visualized by representing the model as a graph and interpreting the VMP update equations as messages.

In this paper, we favor a *Forney-style* Factor Graph (FFG) representation to visualize the factorization properties of probabilistic models and inference by message passing [15]. FFGs are undirected graph representations of factorized probabilistic models in which the conditional distributions are represented by nodes and the variables are associated with edges that connect the nodes. Besides visualizing the factorization properties of probabilistic models, FFGs also provide a formal framework for message passing-based inference in probabilistic models. In the FFGs that we discuss here, we distinguish three types of factors (nodes): soft factors, deterministic factors and equality factors. Throughout the paper, a soft factor represents an Exponential Family (EF) distribution (see (6) for definition) such as a Gaussian, Bernoulli, Gamma or Categorical distribution. Deterministic factors hold deterministic mappings of variables; in particular, we will use the relation $f_\delta(x, z) = p(x|z) = \delta(x - g(z))$, where $g(\cdot)$ is a deterministic function. Lastly, equality factors are used to circumvent the constraint that an edge (representing a variable z) can only be connected to maximally two factors. In an FFG representation, this problem is resolved by adding variable copies z' and z'' and constraining the beliefs over these copy variables through an equality factor $f_=(z, z', z'') = \delta(z - z')\delta(z - z'')$. As an example, the FFG for one time step for a hierarchical state-space model is visualized in Figure 1. In this graph, the factors $f_a(z'_t, z_{t-1}) = p(z'_t|z_{t-1})$, $f_b(x'_t, x_{t-1}, w_t) = p(x'_t|x_{t-1}, w_t)$, $f_c(y_t, x''_t) = p(y_t|x''_t)$ are encoded by EF distributions. The factor $f_\delta(w_t, z''_t) = p(w_t|z''_t) = \delta(w_t - g(z''_t))$, for a given function $g(\cdot)$, represents a non-linear deterministic relation. An interesting property of FFGs is the hierarchical composition: we can create new "higher level" nodes by enclosing a set of connected nodes in a box and integrating out the internal variables in the box. For instance, the *composite* node f_d can be created through the following:

$$\begin{aligned} f_d(x'_t, x_{t-1}, z''_t) &= p(x'_t|x_{t-1}, z''_t) \\ &= \int p(x'_t|x_{t-1}, w_t)\delta(w_t - g(z''_t))\mathrm{d}w_t \\ &= \int f_b(x'_t, x_{t-1}, w_t) f_\delta(w_t, z''_t)\mathrm{d}w_t \end{aligned}$$

For a more detailed introduction to FFGs, we refer to [15,16].

Aside from visualization, FFGs also serve to formalize message-passing-based inference in probabilistic models, and VMP on FFGs realizes coordinate-descent optimization of the free energy functional (1). Coordinate-descent optimization of the free energy refers to iterative updates of the variational factors one at a time while keeping the other factors fixed [12,17]. To illustrate, let us optimize $\mathcal{F}$ with respect to $q(z_k)$ for the system $f_a(z_{1:k})f_b(z_{k:K})$ (see Figure 2). First, we decompose the free energy as follows:

$$\mathcal{F} = \tilde{\mathcal{F}} + \underbrace{\mathbb{E}_{q(z_{1:K})}\left[\log \frac{q(z_k)}{f_a(z_{1:k})f_b(z_{k:K})}\right]}_{\mathcal{F}_k},$$

where $\tilde{\mathcal{F}}$ holds terms that are not a function of variable z_k. The term $\mathcal{F}_k$ can be re-arranged as follows:

$$\mathcal{F}_k = \mathbb{E}_{q(z_k)}\left[\log \frac{q(z_k)}{\exp\left(\mathbb{E}_{q(z_{1:k-1})}[\log f_a(z_{1:k})]\right)\exp\left(\mathbb{E}_{q(z_{k+1:K})}[\log f_b(z_{k:K})]\right)}\right], \tag{2}$$

so it follows that $\mathcal{F}_k$ is minimized when $q(z_k)$ is set proportional to the denominator in (2) [12,17]. In addition, notice that the terms with the local factors f_a and f_b are uncoupled, which paves the way for a message-passing interpretation of coordinate-descent variational

inference. As a result, provided that f_a and f_b are not deterministic factors, the VMP algorithm proceeds by repeating the following four steps until convergence [10]:

1. Choose a variable z_k from the set $z_{1:K}$.
2. Compute the incoming messages.

$$\begin{aligned} \overrightarrow{m}_{z_k}(z_k) &\propto \exp\Big(\langle \log f_a(z_{1:k}) \rangle_{q(z_{1:k-1})}\Big) \\ \overleftarrow{m}_{z_k}(z_k) &\propto \exp\Big(\langle \log f_b(z_{k:K}) \rangle_{q(z_{k+1:K})}\Big) \end{aligned} \tag{3}$$

3. Update the posterior.

$$q(z_k) = \frac{\overrightarrow{m}_{z_k}(z_k)\overleftarrow{m}_{z_k}(z_k)}{\int \overrightarrow{m}_{z_k}(z_k)\overleftarrow{m}_{z_k}(z_k)\mathrm{d}z_k}. \tag{4}$$

4. Update the local free energy (for performance tracking), i.e., update all terms in $\mathcal{F}$ that are affected by the update (4):

$$\mathcal{F}_k = \left\langle \log \frac{q(z_k)}{f_a(z_{1:k})f_b(z_{k:K})} \right\rangle_{q(z_{1:K})}. \tag{5}$$

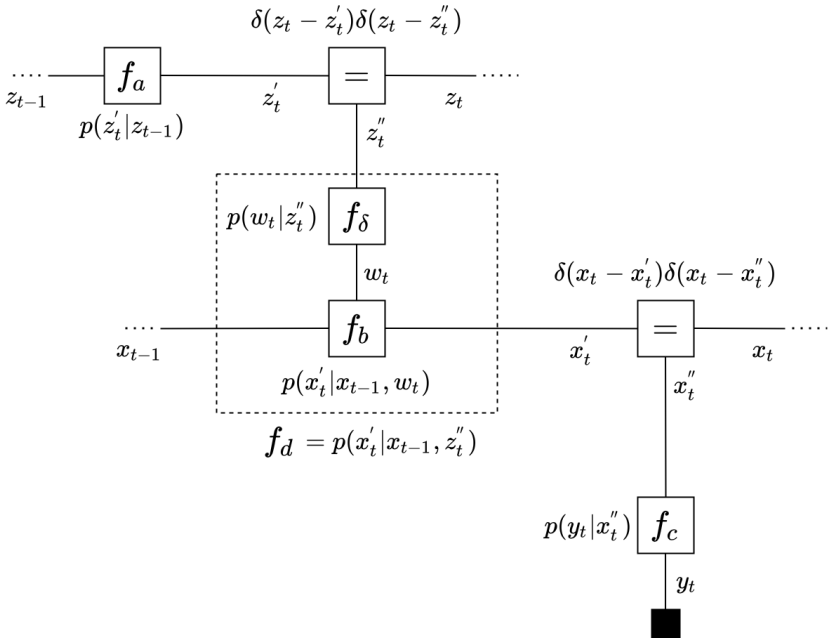


Figure 1. An FFG representation of one time step of a state space model. In FFGs, factors represent (conditional) distributions. Here, f_a, f_b and f_c are soft factors that each represent an exponential family distribution. On the other hand, $f_\delta = \delta(w_t - g(z_t))$ represents a deterministic factor, where $g(\cdot)$ is a deterministic function. It is possible to compose factors and consider them as a single unit. In this example, f_d, visualized by a dashed box, stands for the composition of f_δ and f_b. It is a notational convention to visualize observed values (y_t) by a small black node.

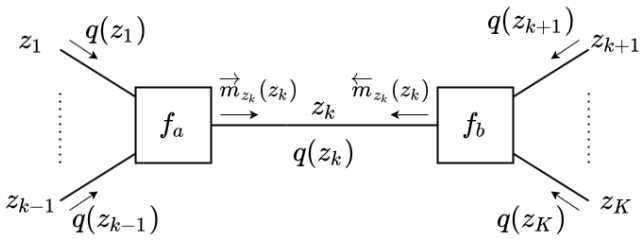


Figure 2. FFG representation for edge z_k with connected nodes f_a and f_b.

As we see in (3), messages flow on edges in both directions. It is common parlance to call one of the messages the forward message (denoted by $\overrightarrow{m}_{z_k}(z_k)$) and the other the backward message ($\overleftarrow{m}_{z_k}(z_k)$).

In this paper, the central problem is *how to execute the VMP update Equation* (3) *through* (5) *for a wide range of specifications for the factors* f_a *and* f_b. In the next section, we specify the proposed Extended VMP (EVMP) solution. A more detailed derivation of the key equations of EVMP is provided in Appendix B.

3. Specification of EVMP Algorithm

Variational Message Passing is a fast, efficient and deterministic approximate inference algorithm. However, the applicability of VMP heavily relies on connected factors being conjugate pairs (see Appendix A). In contrast, Monte Carlo methods (see [18] for message-passing interpretation) are applicable to a wider range of models with non-conjugate factor pairs. Unfortunately, in comparison to VMP, Monte Carlo methods are considerably slower since they rely on stochastic simulations. As we elaborate in Section 5, the recent efforts to combine the best of Monte Carlo methods and variational inference predominantly focus on noisy gradient estimation of the free energy through Monte Carlo sampling and do not take the full advantage of deterministic message passing steps in inference.

In this section, we specify the EVMP algorithm, which combines the efficiency of VMP with the flexibility of the Laplace approximation and the universality of Monte Carlo methods. In the proposed EVMP algorithm, VMP constructs the functional forms of the messages while importance sampling and Laplace approximations are used to estimate the required expectations of statistical quantities if they are not available in closed form. We first specify the range of probability distribution types for factors, messages and posteriors. These different types are used to identify the specific calculation rules for updating the messages and posteriors in (3) and (4). We refer the interested reader to Appendix B for detailed derivations.

3.1. Distribution Types

We consider the following representation types for probability distributions in factors $p(z)$, where z holds a variable.

(1) The standard *Exponential Family* (EF) of distributions, i.e., the following:

$$p(z) = h(z) \exp\big(\phi(z)^\intercal \eta - A_\eta(\eta)\big), \tag{6}$$

where $h(z)$ is the base measure, $\phi(z)$ is the sufficient statistics vector, η is the natural parameters vector and $A_\eta(\eta)$ is the log-partition function.

(2) Distributions that are of the following exponential form:

$$p(z) \propto \exp(\phi(g(z))^\intercal \eta), \tag{7}$$

where $g(z)$ is a deterministic function. The key characteristic here is that $\phi(g(z))$ is not recognized as a sufficient statistics vector for any of the standard EF distributions. We call this distribution type a *Non-Standard Exponential Family* (NEF) distribution. As we show in Section 3.6, this distribution type arises only in backward message calculations.

(3) A *List of Weighted Samples* (LWS), i.e., the following:

$$p(z) := \left\{ \left(w^{(1)}, z^{(1)} \right), \ldots, \left(w^{(N)}, z^{(N)} \right) \right\}. \tag{8}$$

(4) Deterministic relations are represented by *delta distributions*, i.e., the following:

$$p(x|z) = \delta(x - g(z)). \tag{9}$$

Technically, the equality factor $f(x, y, z) = \delta(z - x)\delta(z - y)$ also specifies a deterministic relation between variables.

3.2. Factor Types

Factor types $f(z)$ are represented by EF and delta distributions.

In a VMP setting, as discussed in this and previous papers on VMP, conjugate soft factors from the exponential family enjoy some computational advantages. As an extension to VMP, the EVMP algorithm inherits the same computational advantages for conjugate factor pairs. In order to automate and generalize the inference to custom non-conjugate soft factors, we compose a generic soft factor by a delta distribution (to describe a non-linear deterministic function) and a standard EF distribution. This decomposition relieves us from manually deriving VMP messages for each different soft factor specification. For a given composite node (delta + standard EF), the EVMP algorithm uses the predefined VMP messages for the standard EF component to compute messages around the composite node. As we will see, this formulation yields an almost generic inference procedure.

3.3. Message Types

Forward messages carry either an EF or an LWS distribution. Backward messages carry either an EF or an NEF distribution. This is an arbitrary choice in the sense that we only make this assignment to indicate that in the EVMP algorithm, two colliding messages in posterior calculations are not both of the LWS type nor both of the NEF type.

3.4. Posterior Types

The posteriors $q(z)$ are represented by either the EF or LWS representations.

To summarize the terminology so far, we defined four distribution types: Standard EF (EF), Non-Standard EF (NEF), List of Weighted Samples (LWS) and delta distributions. The end user of our algorithm can design a model by using EF and delta distributions. Under the hood, messages may carry EF, NEF or LWS distributions to render the inference. As the output, the end user is provided with either the EF or LWS posteriors. Next, we discuss how posteriors, messages and free energies are computed in the EVMP algorithm. The different types can be used to identify which computational recipe applies. As an aside, Julia's support for multiple dispatch in functions [14] makes this a very elegant mechanism that requires almost no if–then rules.

3.5. Computation of Posteriors

Here, we discuss how EVMP updates the posteriors in (4). In an FFG, computation of the posterior $q(z)$ is realized by a multiplication of colliding forward and backward messages, respectively $\overrightarrow{m}_z(z)$ and $\overleftarrow{m}_z(z)$, followed by normalization. We distinguish four types of updates.

(1) In the case that the colliding forward and backward messages both carry EF distributions with the same sufficient statistics $\phi(z)$, then computing the posterior simplifies to a summation of natural parameters:

$$\begin{aligned}\overrightarrow{m}_z(z) &\propto \exp(\phi(z)^\intercal \eta_1)\\ \overleftarrow{m}_z(z) &\propto \exp(\phi(z)^\intercal \eta_2)\\ q(z) &\propto \overrightarrow{m}_z(z) \cdot \overleftarrow{m}_z(z) \propto \exp(\phi(z)^\intercal(\eta_1 + \eta_2)).\end{aligned}$$

In this case, the posterior $q(z)$ will also be represented by the EF distribution type. This case corresponds to classical VMP with conjugate factor pairs.

(2) The forward message again carries a standard EF distribution. The backward message carries either an NEF distribution or a non-conjugate EF distribution.

(a) If the forward message is Gaussian, i.e., $\overrightarrow{m}_z(z) = \mathcal{N}(z; \mu_1, V_1)$, we use a Laplace approximation to compute the posterior:

$$\begin{array}{rl} \mu & = \arg\max_z \big(\log \overrightarrow{m}_z(z) + \log \overleftarrow{m}_z(z)\big), \\ V & = (-\nabla\nabla_z(\log \overrightarrow{m}_z(z) + \log \overleftarrow{m}_z(z))|_{z=\mu})^{-1} \\ q(z) & \propto \overrightarrow{m}_z(z) \cdot \overleftarrow{m}_z(z) = \mathcal{N}(z; \mu, V) \end{array} \tag{10}$$

(b) Otherwise ($\overrightarrow{m}_z(z)$ is not a Gaussian), we use Importance Sampling (IS) to compute the posterior:

$$\begin{array}{rl} z^{(1)}, \ldots, z^{(N)} & \sim \overrightarrow{m}_z(z), \\ \tilde{w}^{(i)} & = \overleftarrow{m}_z(z^{(i)}) \text{ for } i = 1, \ldots, N \\ w^{(i)} & = \tilde{w}^{(i)} / \sum_{j=1}^{N} \tilde{w}^{(j)} \text{ for } i = 1, \ldots, N \\ q(z) & \propto \overrightarrow{m}_z(z) \cdot \overleftarrow{m}_z(z) = \left\{ \left(w^{(1)}, z^{(1)}\right), \ldots, \left(w^{(N)}, z^{(N)}\right) \right\}. \end{array} \tag{11}$$

(3) The forward message carries an LWS distribution, i.e., the following:

$$\overrightarrow{m}_z(z) := \left\{ \left(w_1^{(1)}, z_1^{(1)}\right), \ldots, \left(w_1^{(N)}, z_1^{(N)}\right) \right\},$$

and the backward message carries either an EF or NEF distribution. In that case, the posterior computation refers to updating the weights in $\overrightarrow{m}_z(z)$ (see Appendix E):

$$\begin{array}{rl} \tilde{w}^{(i)} & = w_1^{(i)} \overleftarrow{m}_z(z_1^{(i)}) \text{ for } i = 1, \ldots, N \\ w^{(i)} & = \tilde{w}^{(i)} / \sum_{j=1}^{N} \tilde{w}^{(j)} \text{ for } i = 1, \ldots, N \\ z^{(1)}, \ldots, z^{(N)} & = z_1^{(1)}, \ldots, z_1^{(N)} \\ q(z) & \propto \overrightarrow{m}_z(z) \cdot \overleftarrow{m}_z(z) = \left\{ \left(w^{(1)}, z^{(1)}\right), \ldots, \left(w^{(N)}, z^{(N)}\right) \right\}. \end{array} \tag{12}$$

3.6. Computation of Messages

Here, we discuss how EVMP compute the messages (3). We specify different message calculation rules depending of the type of the factor.

(1) If factor $f_a(z_1, z_2, \ldots, z_k)$ is a soft factor of the form (see Figure 3a)

$$f_a(z_{1:k}) = p(z_k|z_{1:k-1}) = h_a(z_k) \exp(\phi_a(z_k)^\intercal \eta_a(z_{1:k-1}) - A_a(z_{1:k-1})).$$

then the outgoing VMP message to z_k is the following EF-distributed message:

$$\overrightarrow{m}_{z_k}(z_k) \propto h_a(z_k) \exp\left(\phi_a(z_k)^\intercal \langle \eta_a(z_{1:k-1}) \rangle_{q(z_{1:k-1})}\right). \tag{13}$$

If rather z_1 (or $z_2, \ldots, z_{k-1}$) than z_k is the output variable of f_a, i.e., if the following is true:

$$f_a(z_{1:k}) = p(z_1|z_{2:k}) = h_a(z_1) \exp(\phi_a(z_1)^\intercal \eta_a(z_{2:k}) - A_a(z_{2:k})).$$

then the outgoing message to z_k is either an EF or an NEF distribution of the following form:

$$\overleftarrow{m}_{z_k}(z_k) \propto \exp\left(\langle \phi_a(z_1) \rangle_{q(z_1)}^\intercal \langle \eta_a(z_{2:k}) \rangle_{q(z_{2:k-1})} - \langle A_a(z_{2:k}) \rangle_{q(z_{2:k-1})}\right). \tag{14}$$

In this last expression, we chose to assign a backward arrow to $\overleftarrow{m}_{z_k}(z_k)$ since it is customary to align the message direction with the direction of the factor, which in this case points to z_1.

Note that the message calculation rule for $\overrightarrow{m}_{z_k}(z_k)$ requires the computation of expectation $\langle \eta_a(z_{1:k-1}) \rangle_{q(z_{1:k-1})}$, and for $\overleftarrow{m}_{z_k}(z_k)$ we need to compute expectations $\langle \phi_a(z_1) \rangle_{q(z_1)}$ and $\langle \eta_a(z_{2:k}) \rangle_{q(z_{2:k-1})}$. In the update rules to be shown below, we will see these expectations of statistics of z appear over and again. In Section 3.8 we detail how we calculate these expectations and in Appendix A, we further discuss the origins of these expectations.

(2) In the case that f_δ is a deterministic factor (see Figure 3b):

$$f_\delta(x, z_{1:k}) = p(x|z_{1:k}) = \delta(x - g(z_{1:k})). \tag{15}$$

then the forward message from f_δ to x is of LWS type and is calculated as follows:

$$\overrightarrow{m}_x(x) = \left\{ \left(\frac{1}{N}, g(z_{1:k}^{(1)}) \right), \ldots, \left(\frac{1}{N}, g(z_{1:k}^{(N)}) \right) \right\}, \tag{16}$$
$$\text{where } z_j^{(i)} \sim \overrightarrow{m}_{z_j}(z_j) \text{ for } j = 1:k.$$

For the computation of the backward message toward z_k, we distinguish two cases:

(a) If all forward incoming messages from the variables $z_{1:k}$ are Gaussian, we first use a Laplace approximation to obtain a Gaussian joint posterior $q(z_{1:k}) = \mathcal{N}(z_{1:k}; \mu_{1:k}, V_{1:k})$; see Appendices B.1.2 and B.2.2 for details. Then, we evaluate the posteriors for individual random variables, e.g., $q(z_k) = \int q(z_{1:k}) \mathrm{d}z_{1:k-1} = \mathcal{N}(z_k; \mu_k, V_k)$. Finally, we send the following Gaussian backward message:

$$\overleftarrow{m}_{z_k}(z_k) \propto q(z_k) / \overrightarrow{m}_{z_k}(z_k). \tag{17}$$

(b) Otherwise (the incoming messages from the variables $z_{1:k}$ are not all Gaussian), we use Monte Carlo and send a message to z_k as a NEF distribution:

$$\overleftarrow{m}_{z_k}(z_k) \approx \frac{1}{N} \sum_{i=1}^{N} \overleftarrow{m}_x(g(z_{1:k-1}^{(i)}, z_k)), \text{ where } z_j^{(i)} \sim \overrightarrow{m}_{z_j}(z_j). \tag{18}$$

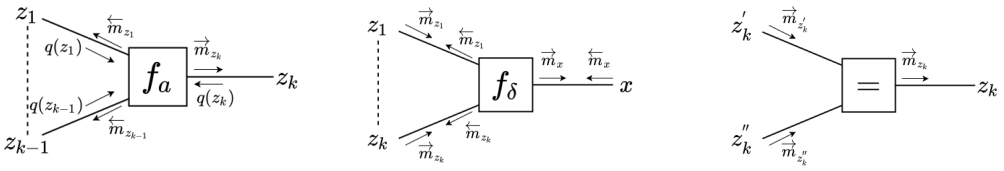
Note that if f_δ is a single input deterministic node, i.e., $f_\delta(x, z_k) = p(x|z_k) = \delta(x - g(z_k))$, then the backward message simplifies to $\overleftarrow{m}_{z_k}(z_k) = \overleftarrow{m}_x(g(z_k))$ (Appendix B.1.1).

(3) The third factor type that leads to a special message computation rule is the equality node; see Figure 3c. The outgoing message from an equality node

$$f_=(z, z', z'') = \delta(z - z')\delta(z - z'')$$

is computed by following the sum–product rule:

$$\begin{aligned} \overrightarrow{m}_{z_k}(z_k) &= \int \underbrace{\delta(z_k - z_k')\delta(z_k - z_k'')}_{\text{node function}} \underbrace{\overrightarrow{m}_{z_k'}(z_k') \overrightarrow{m}_{z_k''}(z_k'')}_{\text{incoming messages}} \mathrm{d}z_k' \mathrm{d}z_k'' \\ &= \overrightarrow{m}_{z_k'}(z_k) \overrightarrow{m}_{z_k''}(z_k). \end{aligned} \tag{19}$$



(**a**) Soft factor. (**b**) Deterministic factor. (**c**) Equality factor.

Figure 3. Different factor types for outgoing message computation rules.

3.7. Computation of Free Energy

Here, we discuss how EVMP computes the FE update from (5). Note that the FE can be decomposed into a subtraction of energy and entropy terms:

$$\begin{aligned}\mathcal{F}_k &= \left\langle \log \frac{q(z_k)}{f_a(z_{1:k}) f_b(z_{k:K})} \right\rangle_{q(z_{1:K})} \\ &= \underbrace{\left\langle \log \frac{1}{f_a(z_{1:k}) f_b(z_{k:K})} \right\rangle_{q(z_{1:K})}}_{\text{(average) energy } \mathcal{U}_a + \mathcal{U}_b} - \underbrace{\left\langle \log \frac{1}{q(z_k)} \right\rangle_{q(z_k)}}_{\text{entropy } \mathcal{H}_k}\end{aligned} \tag{20}$$

These energy and entropy terms can be evaluated because $f_a(z_{1:k}) f_b(z_{k:K})$ contains only factors that are defined in the generative model and $q(z_{1:K})$ is also accessible as a result of variational inference. Thus, we evaluate the FE by evaluating the energy and entropy terms separately.

For an EF-encoded soft factor

$$f_a(z_{1:k}) = p(z_k | z_{1:k-1}) = h_a(z_k) \exp\left(\phi_a(z_k)^\intercal \eta_a(z_{1:k-1}) - A_\eta(\eta_a(z_{1:k-1}))\right),$$

the energy over the factor f_a evaluates to

$$\begin{aligned}\mathcal{U}_a &= -\langle \log(f_a(z_{1:k})) \rangle_{q(z_{1:k})} \\ &= -\langle \log(h_a(z_k)) \rangle_{q(z_k)} - \langle \phi_a(z_k) \rangle_{q(z_k)}^\intercal \langle \eta_a(z_{1:k-1}) \rangle_{q(z_{1:k-1})} + \left\langle A_\eta(\eta_a(z_{1:k-1})) \right\rangle_{q(z_{1:k-1})}.\end{aligned}$$

The entropy terms only need to be evaluated for variables z that are not associated with output edges of deterministic nodes. In that case, we calculate the entropy of $q(z)$ as follows:

1. If $q(z)$ is a represented by a standard EF distribution, i.e.,

$$q(z) = h(z) \exp\left(\phi(z)^\intercal \eta - A_\eta(\eta)\right),$$

then

$$\mathcal{H}_z = -\langle \log(h(z)) \rangle_{q(z)} - \langle \phi(z) \rangle_{q(z)}^\intercal \eta + A_\eta(\eta).$$

2. Otherwise, if $q(z)$ is represented by a LWS, i.e.,

$$q(z) := \left\{ \left(w^{(1)}, z^{(1)} \right), \ldots, \left(w^{(N)}, z^{(N)} \right) \right\},$$

then

$$\mathcal{H}_z = \hat{\mathcal{H}}_z^1 + \hat{\mathcal{H}}_z^2,$$

where $\hat{\mathcal{H}}_z^1$ and $\hat{\mathcal{H}}_z^2$ are evaluated as discussed in (A43) and (A45).

3.8. Expectations of Statistics

In many of the above computations for messages, posteriors and free energies, we need to compute certain expectations of statistics of z, e.g., the computation of the forward message in (13) requires evaluation of $\langle \eta_a(z_{1:k-1}) \rangle_{q(z_{1:k-1})}$. Here, we discuss how EVMP evaluates these expectations. Let us denote a statistic of random variable z by $\Phi(z)$ and assume we are interested in the expected value $\langle \Phi(z) \rangle_{q(z)}$. The calculation rule depends on the type of $q(z)$:

(1) We have two cases when $q(z)$ is coded as an EF distribution, i.e.,

$$q(z) = h(z) \exp\left(\phi(z)^\intercal \eta - A_\eta(\eta)\right) :$$

(a) If $\Phi(z) \in \phi(z)$, i.e., the statistic $\Phi(z)$ matches with elements of the sufficient statistics vector $\phi(z)$, then $\langle \Phi(z) \rangle_{q(z)}$ is available in closed form as the gradient of the log-partition function (this is worked out in Appendix A.1.1, see (A14) and (A15)):

$$\langle \Phi(z) \rangle_{q(z)} \in \nabla_\eta A_\eta(\eta).$$

(b) Otherwise ($\Phi(z) \notin \phi(z)$), then we evaluate

$$\langle \Phi(z) \rangle_{q(z)} \approx \frac{1}{N} \sum_{i=1}^{N} \Phi(z^{(i)}),$$

where $z^{(i)} \sim q(z)$.

(2) In case $q(z)$ is represented by a LWS, i.e., the following:

$$q(z) = \left\{ \left(w^{(1)}, z^{(1)}\right), \ldots, \left(w^{(N)}, z^{(N)}\right) \right\},$$

then, we evaluate the following:

$$\langle \Phi(z) \rangle_{q(z)} \approx \sum_{i=1}^{N} w^{(i)} \Phi(z^{(i)}).$$

3.9. Pseudo-Code for the EVMP Algorithm

Sections 3.1–3.8 provide a recipe for almost universal evaluation of variational inference in factor graphs. We use classical VMP with closed-form solutions when possible, and resort to Laplace or IS approximations when needed. We now summarize the EVMP algorithm by a pseudo-code fragment in Algorithm 1. We use the following notation: $V = V_f \cup V_\delta \cup V_=$ is the set of factor nodes (vertices), where V_f, V_δ, $V_=$ stand for the subsets of soft factor nodes, deterministic nodes and equality nodes, respectively. E is the set of edges that connect the nodes. $G = (V, E)$ represents the entire factor graph. $h_{1:M+L} = z_{1:M} \cup x_{1:L}$ is the set of hidden variables, where $x_{1:L}$ are the variables at the output edges of deterministic nodes. $z_{1:M}$ are also associated with edges in E, but in contrast to $x_{1:L}$, $z_{1:M}$ are not output edges of deterministic nodes.

For structured factorizations, the overall structure remains the same, but messages and posteriors are calculated for sub-graphs instead of single random variables.

An example to illustrate the calculation of messages and posteriors in the EVMP algorithm is provided in Appendix F.

Algorithm 1 Extended VMP (Mean-field assumption)

Require: $G = (V, E)$, $h_{1:M+L}$, $N_{iterations}$, $q_{initial}$
  for $j = 1 \dots M + L$ **do**
    initialize posteriors $q(h_j)$ using $q_{initial}$
  end for
  for i = $1 \dots N_{iterations}$ **do**
    Set Free Energy $\mathcal{F} = 0$
    for $j = 1 \dots M + L$ **do**
      Calculate messages $\overrightarrow{m}_{h_j}(h_j)$ and $\overleftarrow{m}_{h_j}(h_j)$ using Section 3.6
      Calculate posterior $q(h_j)$ using Section 3.5
      if $h_j \in z_{1:M}$ **then**
        Calculate entropy $\mathcal{H}_{h_j}$ using Section 3.7
        Update Free Energy $\mathcal{F} = \mathcal{F} - \mathcal{H}_{h_j}$
      end if
    end for
    for all $v \in V_f$ **do**
      Calculate energy $\mathcal{U}_v$ using Section 3.7
      Update Free Energy $\mathcal{F} = \mathcal{F} + \mathcal{U}_v$
    end for
    return $q(h)$ **for all** $h \in h_{1:M+L}$ and $\mathcal{F}$
  end for

4. Experiments

We illustrate EVMP-based inference on three different applications (code for experiments can be found at https://github.com/biaslab/ExtendedVMP (accessed on 25 June 2021)). For each application, we show the favorable features of EVMP together with its shortcomings in comparison to Turing [5], which is a general purpose Julia probabilistic programming package.

4.1. Filtering with the Hierarchical Gaussian Filter

The Hierarchical Gaussian Filter (HGF) [19,20] is a popular generative model in the neuroscience community. The HGF consists of a Gaussian random walk model, where the variance of the Gaussian is a nonlinear function of the state of the next higher layer, that in turn evolves according to a Gaussian random walk, an so on. Due to the nonlinear link between the layers, classical VMP rules do not have a closed-form solution. While in principle, variational updates through Laplace approximation can be manually derived for the HGF model [19], automatically generated EVMP update rules alleviate the need for cumbersome and error-prone manual derivations.

The 2-layer HGF model is defined as

$$z_t \sim \mathcal{N}(z_{t-1}, \sigma_z^2) \tag{21a}$$
$$w_t = \exp(z_t) \tag{21b}$$
$$x_t \sim \mathcal{N}(x_{t-1}, w_t) \tag{21c}$$
$$y_t \sim \mathcal{N}(x_t, \sigma_y^2)\,. \tag{21d}$$

For this experiment, we generated $T = 400$ data points by the following process. First, we generated noisy hidden states using $z_t \sim \mathcal{N}\left(\sin\left(\frac{\pi}{60}t\right), 0.01\right), t = 1 \dots 400$. Next, we generated observations following model (21a–d) with $\sigma_y^2 = 0.1$. The generated data set is visualized in (the lower subgraph of) Figure 4.

Next, we filtered the data set by a second HGF, also given by (21a–d) with priors $z_0 \sim \mathcal{N}(0, 1)$, $x_0 \sim \mathcal{N}(0, 1)$ and parameters $\sigma_z^2 = \sigma_y^2 = 0.1$. We used EVMP to track the hidden states z_t and x_t. All inference steps including the message passing schedule for filtering in the HGF are detailed in [19]. For each time step, EVMP was run for 10 iterations at each filtering step.

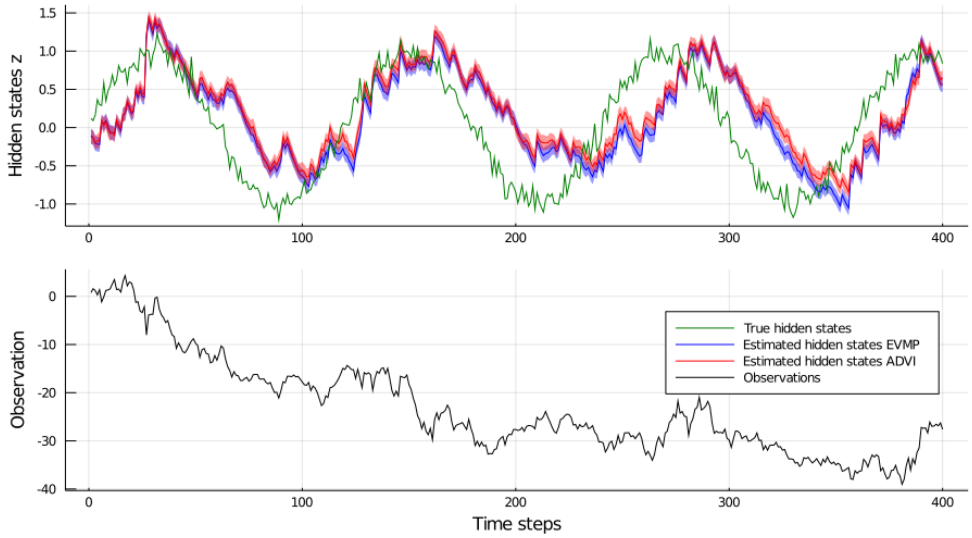


Figure 4. Above: Hidden states $z_{1:400}$ and their estimates (ribbon is one variance). The estimates of ForneyLab's Extended VMP are designated by blue while the estimates of Turing's ADVI are marked by red. Below: Observed synthetic data.

For comparison, we implemented a similar filtering procedure by Automatic Differentiation Variational Inference (ADVI) [21], executed by Julia's *Turing.jl* [5] package. At each time step t, the priors over z_{t-1} and x_{t-1} are set to Gaussian distributions, the mean and variance parameters of which are determined by sampling from the variational posteriors at $t-1$. The only difference between the ForneyLab and Turing implementations, in terms of posterior distribution factorization, is that in Turing's ADVI, we posit a fully factorized posterior distribution. This assumption decreases the number of parameters to be estimated via automatic differentiation and speeds up the inference procedure. On the other hand, pre-defined message passing rules in ForneyLab enable us to retain the dependency structure between x_{t-1} and x_t at time step t in exchange for almost no run-time loss. To be more precise, at time step t, we run inference on the following model: $q_f(z_{t-1})p(z_t|z_{t-1})\delta(w_t - \exp(z_t))q_f(x_{t-1})p(x_t|x_{t-1}, w_t)p(y_t|x_t)$ where $q_f(z_{t-1})$ and $q_f(x_{t-1})$ are the posterior approximations from the previous time step. In ForneyLab, we run the inference with variational distribution $q(z_{t-1})q_f(z_t)q(x_{t-1}, x_t)$ with $q_f(x_t) = \int q(x_{t-1}, x_t)\mathrm{d}x_{t-1}$. We plot estimations for $q_f(z_t)$ in Figure 4. In ADVI, the variational distribution is $q(z_{t-1})q_f(z_t)q(x_{t-1})q_f(x_t)$. Once inference has completed, Turing allows drawing samples from the variational distribution. We then calculate the mean and variance of these samples to fit Gaussian distributions on $q_f(z_t)$ and $q_f(x_t)$.

The estimated tracks of z_t are visualized in Figure 4. For both EVMP and ADVI, the estimated hidden states largely coincide. However, we observe that both methods capture the periodic character of the true hidden states $z_{1:400}$ with a delay. We believe that there are two plausible explanations for the delayed estimations: (1) in the model specification, we assume that the data generative process is not known fully. The variables $z_{1:400}$ are originally generated from a sinusoidal function of discrete time steps. However, in the model specification, we do not use this information; (2) in the model specification, we define a random walk over hidden variables $z_{1:400}$ that posits the mean of z_t as z_{t-1}. Elaborating the latter factor, the random walk avoids a hidden variable z_t to change drastically, compared to z_{t-1} while x_t forces z_t to explain the volatility in the process. Reconciling the beliefs from x_t and z_{t-1}, both Extended VMP and ADVI estimate z_t with a delay.

In Turing's ADVI procedure, we used 10 samples per iteration for gradient estimation and set the maximum number of iterations to 4000 per time step to be able to capture this periodic behavior. The overall inference is completed in roughly 1.5 min (this and furtherexperiments were carried out on a machine with the following specs: Julia 1.5.0,

Turing v0.15.9, AMD Ryzen 7 3700X 3.6 GHz 8-Core CPU, 16 GB DDR4-3200 MHz RAM.). ForneyLab's EVMP procedure, on the other hand, is able to perform inference in under 7 s on this time series; see Table 1. The speed of ForneyLab stems from the hybrid inference nature of EVMP. EVMP resorts to gradient-based optimization only to infer $q_f(z_t)$ and the sampling procedure is required only to estimate statistics related to w_t to be used in the update steps of $q(x_{t-1}, x_t)$. In contrast, ADVI requires sampling and employs noisy gradients in the estimation of all the components of the variational distribution. This experiment validates EVMP as a fast automated variational inference solution for filtering in hierarchical dynamic models.

Table 1. Run-time comparison of EVMP (in *ForneyLab.jl*) vs. ADVI (in *Turing.jl*) for the hierarchical Gaussian filter model.

Algorithm	Run Time (s)
EVMP (ForneyLab)	6.366 ± 0.081
ADVI (Turing)	91.468 ± 3.694

4.2. Parameter Estimation for a Linear Dynamical System

In this experiment, we focused attention on a system identification task in a Linear Dynamical System (LDS) [22,23]. An LDS is generally defined as

$$x_t | x_{t-1} \sim \mathcal{N}(Ax_{t-1}, Q) \tag{22a}$$
$$y_t | x_t \sim \mathcal{N}(Bx_t, R) \tag{22b}$$

where y_t are observations and x_t are hidden states.

In this experiment, we are interested in inferring the transition matrix A together with the hidden states from a set of observations. Manually derived closed-form solutions for the system identification task are available both in maximum likelihood estimation [24] and a variational Bayesian approximation [25] contexts. Nevertheless, the goal in this and other papers on probabilistic programming packages is to automatically infer posteriors over the hidden states and parameters without resorting to manual derivations. In principle, EVMP supports to infer the hidden states, A, B, Q and R concurrently. Of course, depending on specific circumstances such as system identifiability and the richness of the observed data, the performance may vary.

In order to execute our experiment, we first extend (22a,b) with a prior on A as follows:

$$a \sim \mathcal{N}(\mu_a, V_a) \tag{23a}$$
$$A = \text{reshape}(a, (m, m)) \tag{23b}$$
$$x_t | x_{t-1} \sim \mathcal{N}(Ax_{t-1}, Q) \tag{23c}$$
$$y_t | x_t \sim \mathcal{N}(Bx_t, R) \tag{23d}$$

In (23a–d), a holds the vectorized representation of the transition matrix A. Note that (23b) can be written as follows:

$$p(A|a) = \delta(A - \text{reshape}(a, (m, m)))\,,$$

and through this manipulation we identify $\text{reshape}(a, (m, m))$ as the deterministic factor in (15). As a result, ForneyLab's EVMP works out-of-the-box for inference of the transition matrix in (23a–d).

We first generated a data set of $T = 40$ number of samples by running model (22a,b) with parameters $Q = 0.01 \times I_{2\times 2}$, $R = 0.1 \times I_{2\times 2}$, $B = I_{2\times 2}$ and $A = \begin{bmatrix} 1.0 & 0.2 \\ -0.5 & 0.8 \end{bmatrix}$.

Next, we presented the data set to a second LDS model and aimed to infer posteriors over hidden states and transition matrix A. The prior on a was set to $a \sim \mathcal{N}(0_4, I_{4\times 4})$ and all other parameters were set to the same values as in the data generation process.

We compared the performance of ForneyLab's EVMP with Turing's ADVI and NUTS (No U-Turn Sampler, a Hamiltonian Monte Carlo sampling-based inference method) [26] engines, see Figure 5. Both EVMP, ADVI and NUTS successfully converged to almost coinciding estimates of the transition matrix (no notable difference when visualized). We also show free energy tracks for EVMP and ADVI in Figure 5. In this experiment, Turing's ADVI outperformed ForneyLab's EVMP in terms of total execution time and the free energy minimization. As a mitigating factor in this analysis, the pre-compilation of the message passing schedule in ForneyLab takes about 13 s, while actual execution of the generated inference algorithm is on par with Turing's ADVI. Execution time details are shown in Table 2.

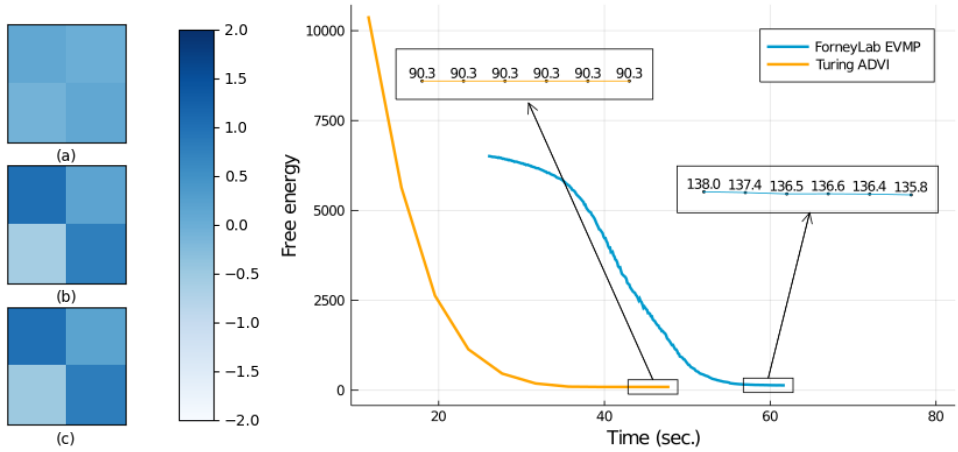


Figure 5. Free energy tracks for EVMP on the LDS transition matrix identification task. Left: (**a**) Mean estimate EVMP for the transition matrix A after 50 iterations, (**b**) mean estimate after 300 iterations, (**c**) true transition matrix A that was used to generate the synthetic data. Right: Free energy tracks by ForneyLab's EVMP and Turing's ADVI procedures.

Table 2. Run-time results for transition matrix estimation in the LDS model.

Algorithm	Free Energy	Total Time (s)
EVMP (ForneyLab)	135.837	58.674 ± 0.467
ADVI (Turing)	90.285	47.405 ± 1.772
NUTS (Turing)	-	78.407 ± 4.206

4.3. EVMP for a Switching State Space Model

In this experiment, we went beyond models that only contain continuously valued variables and inquired the capabilities of EVMP on a Switching State Space Model (SSSM) [27], which consists of both continuous and discrete hidden variables. The assumption of constant model parameters in the LDS of Section 4.2 does not account for the regime changes that occur in many dynamical systems of interest. The SSSM does allow for modeling parameter switches, and in this experiment we used the following model:

$$p(A) = \prod_{k=1}^{3} \mathrm{Dir}(A[:,k]; \alpha_k) \tag{24a}$$

$$p(z_t|z_{t-1}) = \prod_{k=1}^{3}\prod_{j=1}^{3} A_{kj}^{z_{tk}z_{t-1,j}} \tag{24b}$$

$$p(x_t|x_{t-1}, z_t) = \prod_{k=1}^{3} \mathcal{N}(x_t|x_{t-1}, v_k)^{z_{tk}} \tag{24c}$$

$$p(y_t|x_t) = \mathcal{N}(y_t|x_t, 1) \tag{24d}$$

In this system, $y_t \in \mathbb{R}$ are observations, $x_t \in \mathbb{R}$ is a continuously valued hidden state and z_t is a one-hot coded three-dimensional selection variable, i.e., $z_{tk} \in \{0, 1\}$ and $\sum_{k=1}^{3} z_{tk} = 1$. The parameters of the system are the state variances v_k and concentration parameters α_k. The elements of α_k are all 1, except the k^{th} element which is set to 100 to disfavor frequent regime switches, e.g., $\alpha_2 = [1, 100, 1]^\intercal$.

We generated $T = 120$ data points from a random walk process (24c) and (24d) with process noise variance parameter $v = [v_1, v_2, v_3] = [10, 4, 1]$. From time step $t = 2$ to $t = 25$, we set $z_{t,1} = 1$ and consequently $p(x_t|x_{t-1}) = \mathcal{N}(x_{t-1}, 10)$. From time step $t = 26$ to $t = 75$, we set $z_{t,2} = 1$ and between $t = 76$ to $t = T = 120$ we set $z_{t,3} = 1$. The generated time series is shown in Figure 6.

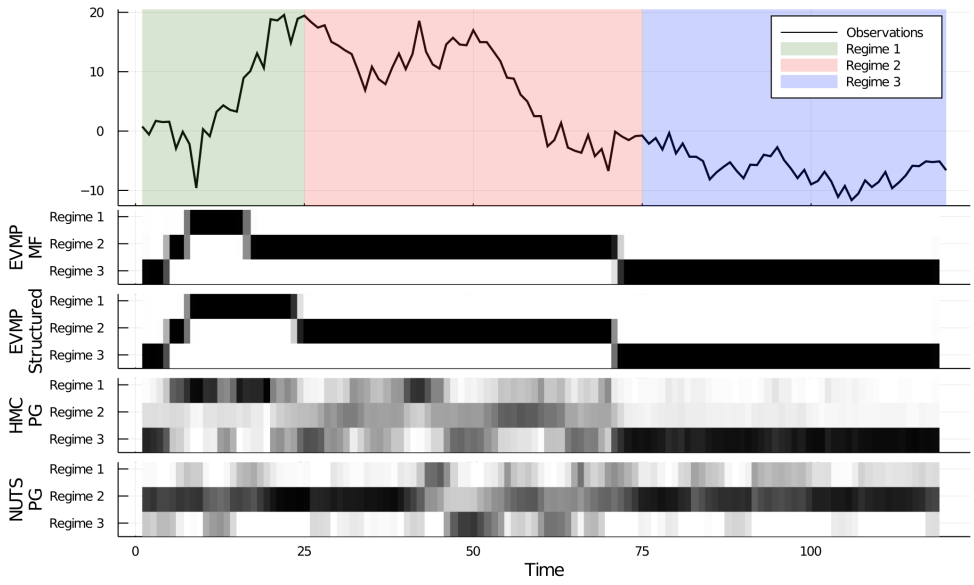


Figure 6. Performance results for automated inference in SSSM. **Top**: generated data set. **Bottom** 4 subgraphs: posterior for regime selection variable z_t by MF-EVMP, SMF-EVMP, HMC and NUTS procedures respectively. In the Turing simulations (HMC and NUTS), the number of particles in the Particle Gibbs sampler was set to 50. In the NUTS sampler, the adaptation step size is 1000 and the target accept ratio is 0.65. The HMC sampler was tried with varying step sizes, including 0.001, 0.01, 0.1, 0.2, and 0.4 and leapfrog step numbers 10, 20, and 30. The best results are shown.

The main difficulty in state inference for the SSSM stems from the coupling between x and z. This is because the variational message passing rules around the node $p(x_t|x_{t-1}, z_t)$ are not pre-defined in ForneyLab, although technically they can be worked out to closed-form expressions [27]. If EVMP were not available either, then a ForneyLab end user would be expected to manually derive closed-form update rules and implement these rules in an additional ForneyLab node. This type of manually assisted inference by end user calculations is what we try to avoid with EVMP and with probabilistic programming

packages in general. EVMP enables the user to compensate for the lack of stored message-passing rules by introducing an auxiliary variable s in the model with a deterministic relation between s and z:

$$g(z_t) = \sum_{k=1}^{3} z_{tk} \cdot v_k \tag{25a}$$

$$p(s_t|z_t) = \delta(s_t - g(z_t)) \tag{25b}$$

$$p(x_t|x_{t-1}, s_t) = \mathcal{N}(x_t; x_{t-1}, s_t) \tag{25c}$$

$$p(x_t|x_{t-1}, z_t) = \int p(x_t|x_{t-1}, s_t)p(s_t|z_t)\mathrm{d}s_t. \tag{25d}$$

After we extend model specification (24a–d) by (25a–d), then ForneyLab can run EVMP-based inference out of the box. Note that there is no need for manual inference calculations, but rather a simple manipulation of the generative model that makes the system suited for automated inference.

We tested the performance of two different constraints on the posterior distribution: (1) a mean-field assumption, i.e., $q(x_{1:T}, z_{1:T}) = \prod_{t=1}^{T} q(x_t)q(z_t)$; (2) a *structured* mean-field assumption, i.e., $q(x_{1:T}, z_{1:T}) = q(x_{1:T}) \prod_{t=1}^{T} q(z_t)$, see Figure 6. We observe that the structured factorization, being a less stringent constraint on q, yields a slightly better performance than the mean-field factorization, particularly in estimating the length of the first regime.

We also compared the performance of ForneyLab's EVMP method to Turing's inference method. As opposed to the previous two experiments, we could not use solely ADVI, nor Hamiltonian Monte Carlo (HMC, [28,29]) and NUTS samplers in this experiment since these procedures do not allow inference for discrete random variables. Turing does provide the option to use a Particle Gibbs (PG) sampler [30,31] for the estimation of the discrete random variables ($z_{1:T}$) in conjunction with the estimation of the continuous random variables ($x_{1:T}$, A) by HMC and NUTS. The performance results for NUTS-PG and HMC-PG are shown in Figure 6. The performance of the NUTS-PG and HMC-PG samplers in estimating the correct regimes is far below the EVMP results, although the HCM-PG sampler correctly identified the third regime. The run-time scores are shown in Table 3.

Table 3. Experimental results for switching state space model.

Algorithm	Free Energy	Total Time (s)
EVMP (Mean-field)	283.991	42.722 ± 0.197
EVMP (Structured)	273.596	51.684 ± 0.311
HMC-PG (Turing)	-	116.291 ± 0.886
NUTS-PG (Turing)	-	51.715 ± 0.441

5. Related Work

Hybrid Monte Carlo variational inference techniques have been studied prior to our work. However, mainstream research predominantly consists of variational methods within Monte Carlo techniques as opposed to our Monte Carlo methods within a variational inference approach.

For instance, ref. [32] casts variational distributions as proposal distributions in a Markov-Chain Monte Carlo (MCMC) procedure. Similarly, ref. [33] employs variational methods to design adaptive proposal distributions for Importance Sampling (IS). In [34], gradient estimates of a variational objective are used to tune the parameters of the proposal distributions for MCMC. On the other hand, Monte-Carlo Co-Ordinate Ascent Variational Inference (MC-CAVI), proposed in [35], differs from the aforementioned methods in that it

uses MCMC in the calculation of expectations required within the fixed-point iterations of Coordinate Ascent Variational Inference (CAVI).

In this paper, we follow a similar approach as [35], but we use IS to estimate the expectation quantities required in VMP. Both MCMC and IS have their own merits. IS smoothly interfaces with the message passing interpretation of Bayesian inference, which further leads to automated design of proposal distributions. We use Laplace approximation for Gaussian posteriors for variables with Gaussian priors. In the context of dynamical systems, this approach notably overlaps with Gaussian filtering techniques ([36], Section 6) that is often achieved by Assumed Density Filtering ([37], Section 8.4).

As we show in Appendix E, in the approach that we propose, it is also possible to run automated Bootstrap particle filtering [36,38] rather than Gaussian filtering methods. As show in [18], particle filtering can also be framed as message passing on a factor graph. The connection between the particle filtering and variational optimization was introduced in [39]. Their formalism is based on an extension of Particle Belief Propagation [40] to Tree-Reweighted Belief Propagation [41], while ours revolves around VMP. Similar to our approach, Particle Variational Inference (PVI) [42] aims at optimizing a variational objective by successive IS approximations to true posterior distributions. While PVI applies well to inference for discrete random variables, our EVMP proposal applies to both continuous and discrete random variables.

Variational inference in the context of deterministic building blocks in probabilistic models was studied in [43]. Wheras [43] allows non-linearities to be placed only after Gaussian nodes, the proposed EVMP method generalizes this concept to EF distributed factors.

Non-conjugate Variational Message Passing (NC-VMP) [44] addresses the non-conjugate factor issue in VMP. Assuming that the posterior distribution is an EF distribution, NC-VMP projects the messages to the distribution space of the posterior by equating their sufficient statistics. Thus NC-VMP tunes the natural parameters of the messages in such a way that they converge to the stationary points of the KL divergence between the approximated and true posteriors. Ref. [44] also reports that the algorithm necessitates damping for convergence in practice. In response, ref. [45] presents Conjugate-Computation Variational Inference (CVI) as a universal inference method that is based on stochastic optimization techniques. As opposed to alternative stochastic variational inference techniques, such as Black-Box Variational Inference [46] and Automatic Differentiation Variational Inference [21], CVI exploits the conjugacy structure of the probabilistic models, which leads to faster convergence. In CVI, non-conjugate factors are incorporated into coordinate ascent steps of mean-field variational inference (with ELBO objective) through a stochastic optimization procedure to form compact posterior approximations with standard probability distributions. In our EVMP approach, the Laplace approximation entails a similarly nested optimization procedure to form compact approximations with Gaussian distributions. Nevertheless, our particle approximations to the true posteriors obviate the need for additional gradient-based optimizations to estimate the parameters of the posteriors.

Finally, the original VMP paper [9] itself briefly mentions sampling methods to overcome the issues with non-conjugate priors. However, they do not extend this idea to deterministic nodes and rather present it as a fallback method whenever soft factors are tied to non-conjugate soft factor priors. Inspired by their vision of approximating the expectation quantities by sampling techniques, we introduce here a fully automated, very broadly applicable extended VMP procedure.

6. Discussion

In this paper, we present a method for almost universal variational inference on factorized probabilistic models. The core of our method is the locality feature of VMP: the messages at a soft factor are functions of expectations related to arguments of the factor. We employ IS to estimate these expectations or directly approximate posteriors by Laplace approximation if a Gaussian posterior is reasonable. We also extended the Julia package ForneyLab with the proposed EVMP method. In contrast to many alternative PPLs that are

solely based on Monte Carlo methods, ForneyLab allows end users to take full advantage of closed-form message passing rules while resorting to small-scale numerical approximations only when needed. We showed that ForneyLab provides an efficient automated variational Bayesian inference tool that in some instances may be preferable to the state-of-the-art Turing package, especially for tasks that include filtering in dynamical models or discrete variables in state space models.

While the experiments support the notion that EVMP is a promising method for inference in non-linear and non-conjugate models, we have not tested our method yet in high-dimensional problems. It is well-known that importance sampling is not efficient in high dimensions [47]. Therefore, we anticipate that for high-dimensional inference tasks with continuous random variables, Hamiltonian Monte Carlo-based methods could outperform EVMP both in terms of run-time and quality of the estimates. Nevertheless, it should be possible to alleviate the deficiencies of EVMP in high dimensions by replacing IS and Laplace approximations by HMC samplers. In essence, HMC is an MCMC method and ref. [35] shows the efficiency of MCMC methods in estimation of the expectations that are required in variational inference. Yet, in lower dimensions, we favor IS and Laplace approximations both because of their promising performance scores in the experiments and also because EVMP relieves users of choosing hyperparameters for the best performance. Recall that in the SSSM experiments in Section 4.3, we tested HMC with various hyperparameters to attain the best performance, and yet EVMP was more successful in detecting the hidden regimes. Moreover, in contrast to EVMP, plain HMC is not applicable to estimate discrete variables and needs to be combined with other samplers to run inference on the models with discrete and continuous variables.

In Appendix C, we introduce a variational free energy estimation method that resorts to approximations only if the closed-form expressions of the information-theoretic measures are not available. This differs from alternative automated variational inference techniques, such as Automatic Differentiation Variational Inference (ADVI), which estimates the entire free energy over Monte Carlo summation. Moreover, like HMC, the applicability of ADVI is also limited to continuous variables.

In EVMP, proposal distributions for importance sampling are automatically set to forward messages. Although it is a practical solution with an elegant interpretation in a message passing context, forward messages do not carry information regarding observations. Therefore, we may not acquire useful samples from forward messages if the observations lead to peaky backward messages. In future work, we aim to investigate the effects of alternative proposal distribution design methods.

One major drawback of our ForneyLab implementation is that ForneyLab does not allow loops during the inference procedure. We rarely encounter this problem with soft factors since the mean field assumption breaks the loops by imposing additional factorizations in variational distributions. However, this may not be the case with deterministic nodes. This is because the input and the output variables of deterministic nodes are tied to each other through a deterministic mapping even after the mean field assumption. For example, consider the following mixture model specification: $p(z) = \mathcal{B}er(\rho)$, $p(x|z) = \delta(x - g_1(z))$ with $g_1(z) = \mu_1 \cdot z + \mu_2 \cdot (1 - z)$, $p(w|z) = \delta(w - g_2(z))$ with $g_2(z) = w_1 \cdot z + w_2 \cdot (1 - z)$ and $p(y|x, w) = \mathcal{N}(x, 1/w)$. Although it is a valid model specification with properly defined message passing rules, the EVMP algorithm is precluded due to the loop: the variable z is connected to two deterministic nodes ($p(w|z)$ and $p(x|z)$) the outputs of which are connected to the same node $p(y|x, w)$. Belief propagation (BP) [48,49] faces with a similar problem on loopy graphs. Nonetheless, it has been proven that iteratively running BP on loopy graphs often yields satisfactory approximations though the convergence is not guaranteed ([12], Section 8.4.7), ([37], Section 22.2). Therefore, it is worth investigating the performance of EVMP executed in a loopy setting.

There are similarities between EVMP and Expectation Propagation (EP) [50,51] in the sense that both methods estimate the moment parameters of posteriors. In contrast to EP, which approximates belief propagation (BP) [48,49] messages, EVMP approximates VMP

messages, which is applicable to a broader range of model specifications. In future work, we aim to investigate and exploit this relation.

7. Conclusions

We developed a hybrid message passing-based approach to variational Bayesian inference that supports deterministic and non-conjugate model segments. The proposed Extended VMP (EVMP) method defaults to analytical updates for conjugate factor pairs and uses a local Laplace approximation or importance sampling when numerical methods are needed. EVMP was implemented in Julia's ForneyLab package (see Appendix D) and a set of simulations shows very competitive inference performance on various inference tasks, particularly for state and parameter tracking in state-space models.

Author Contributions: Conceptualization, S.A.; methodology, S.A. and I.B.; software, I.B. and S.A.; validation, I.B. and S.A.; writing—original draft preparation, S.A. and I.B.; writing—review and editing, B.d.V. and S.A.; supervision, B.d.V.; funding acquisition, B.d.V. All authors have read and agreed to the published version of the manuscript.

Funding: This work is partly funded by GN Advanced Science.

Data Availability Statement: Not applicable.

Acknowledgments: The authors want to extend gratitude to our fellow researchers at BIASlab for interesting discussions on the topics in this paper.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

VMP	Variational Message Passing
EVMP	Extended Variational Message Passing
BP	Belief propagation
EP	Expectation propagation
FFG	Forney-style Factor Graph
EF	Exponential family
NEF	Non-standard exponential family
LWS	List of Weighted Samples
IS	Importance sampling
MCMC	Markov Chain Monte Carlo
HMC	Hamiltonian Monte Carlo
ADVI	Automatic Differentiation Variational Inference
PG	Particle Gibbs

Appendix A. On the Applicability of VMP

In this section, we show that the applicability of the VMP algorithm relies on connected factors being conjugate pairs in exponential family of distributions. Non-conjugate connected factors lead to intractable posteriors and messages. Nevertheless, we show that for a given soft factor, the corresponding VMP messages are locally expressed in terms of some expectation quantities. If these expectations are not available in closed form, then we can estimate them to approximate the VMP messages around the non-conjugate factor pairs.

Let us focus on Figure 2. We postulate the following assumptions:

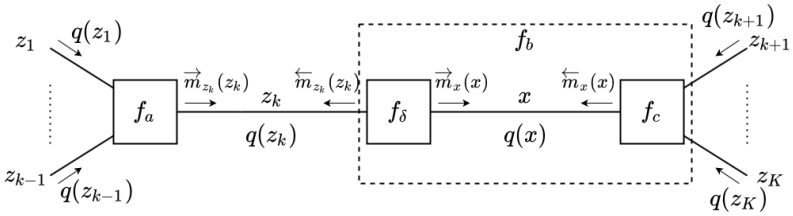


Figure A1. Deterministic conditional distributions often complicate VMP. An example deterministic conditional is visualized for $f_\delta(x, z_k) = p(x|z_k) = \delta(x - g(z_k))$. $f_\delta(x, z_k)$ and $f_c(x, z_{k+1:K})$ together form the composite node $f_b(z_{k:K})$.

- $f_a(z_{1:k})$ is an element of the exponential family (EF) of distributions, i.e.,

$$\begin{aligned} f_a(z_{1:k}) &= p(z_k|z_{1:k-1}) \\ &= h_a(z_k) \exp(\phi_a(z_k)^\intercal \eta_a(z_{1:k-1}) - A_a(z_{1:k-1})). \end{aligned} \tag{A1}$$

  In this equation, $h_a(z_k)$ is a base measure, $\eta_a(z_{1:k-1})$ is a vector of natural (or canonical) parameters, $\phi_a(z_k)$ are the sufficient statistics, and $A_a(z_{1:k-1})$ is the log-partition function, i.e., $A_a(z_{1:k-1}) = \log(\int h_a(z_k) \exp(\phi_a(z_k)^\intercal \eta_a(z_{1:k-1}))dz_k)$. It is always possible to write the log-partition function as a function of natural parameters $A_{\eta_a}(\eta_a)$, such that $A_{\eta_a}(\eta_a) = A_a(z_{1:k-1})$. Throughout the paper, we sometimes prefer the natural parameter parameterization of the log partition.
- We differentiate a few cases for f_b:
  1. f_b is also an element of the EF, given by the following:

$$\begin{aligned} f_b(z_{k:K}) &= p(z_K|z_{k:K-1}) \\ &= h_b(z_K) \exp(\phi_b(z_K)^\intercal \eta_b(z_{k:K-1}) - A_b(z_{k:K-1})), \end{aligned} \tag{A2}$$

     and is a *conditionally conjugate pair* with f_a for z_k. This (conditional conjugacy) property implies that, given $z_{k+1:K}$, we can modify f_b in such a way that its sufficient statistics will match the sufficient statistics of f_a. Technically, this means we can rewrite f_b as follows:

$$f_b(z_k, z_{k+1:K}) = h_b(z_K) \exp(\phi_a(z_k)^\intercal \eta_{ba}(z_{k+1:K}) + c_{ba}(z_{k+1:K})). \tag{A3}$$

     The crucial element of this rewrite is that both $f_a(z_{1:k})$ and $f_b(z_{k:K})$ are written as exponential functions of the same sufficient statistics function $\phi_a(z_k)$. This case leads to the regular VMP update equations, see Appendix A.1.
     Our Extended VMP does not need this assumption and derives approximate VMP update rules for the following extensions.
  2. f_b is an element of the EF, but not amenable to the modification given in (A3), i.e., it cannot be written as an exponential function of sufficient statistics $\phi_a(z_k)$. Therefore, f_b is not a conjugate pair with f_a for z_k.
  3. $f_b(z_{k:K})$ is a composition of a deterministic node with an EF node, see Figure A1. In particular, in this case $f_b(z_{k:K})$ can be decomposed as follows:

$$f_b(z_{k:K}) = \int \delta(x - g(z_k)) f_c(x, z_{k+1:K}) \mathrm{d}x \tag{A4a}$$

$$= f_c(g(z_k), z_{k+1:K}), \tag{A4b}$$

     where $x = g(z_k)$ is a deterministic, possibly nonlinear transformation and $f_c(x, z_{k+1:K})$ is an element of the EF:

$$\begin{aligned} f_c(x, z_{k+1:K}) &= p(z_K|x, z_{k+1:K-1}) \\ &= h_c(z_K) \exp(\phi_c(z_K)^\intercal \eta_c(x, z_{k+1:K-1}) - A_c(x, z_{k+1:K-1})). \end{aligned} \tag{A5}$$

We assume that the conjugate prior to f_c for random variable x has sufficient statistics vector $\hat{\phi}_c(x)$, and hence (A5) can be modified as follows:

$$f_c(x, z_{k+1:K}) = h_c(z_K) \exp(\hat{\phi}_c(x)^\intercal \hat{\eta}_c(z_{k+1:K}) + \hat{c}_c(z_{k+1:K})), \tag{A6}$$

where $\hat{c}_c(z_{k+1:K})$ refers to the terms that does not include x.

Appendix A.1. VMP with Conjugate Soft Factor Pairs

The original VMP algorithm arises as an efficient inference procedure in models that solely consist of conjugate factor pairs. This is because conjugate factor pairs yield analytically tractable messages and posterior calculations. Next, we shortly review the effect of conjugate factor pairs on VMP updates.

Appendix A.1.1. Messages and Posteriors

The VMP message from the factor f_a to z_k can easily be evaluated by applying (3) to (A1):

$$\overrightarrow{m}_{z_k}(z_k) \propto h_a(z_k) \exp\Big(\phi_a(z_k)^\intercal \langle \eta_a(z_{1:k-1}) \rangle_{q(z_{1:k-1})}\Big). \tag{A7}$$

Since f_b is conjugate to f_a, its functional form can be modified as (A3) and by applying (3) to (A3), we find the VMP message from the factor f_b to z_k:

$$\overleftarrow{m}_{z_k}(z_k) \propto \exp\Big(\phi_a(z_k)^\intercal \langle \eta_{ba}(z_{k+1:K}) \rangle_{q(z_{k+1:K})}\Big). \tag{A8}$$

Given that the messages $\overrightarrow{m}_{z_k}(z_k)$ and $\overleftarrow{m}_{z_k}(z_k)$ have the same sufficient statistics, the posterior update step reduces to summation of the messages' natural parameters:

$$q(z_k) \propto h_a(z_k) \exp\left(\phi_a(z_k)^\intercal \underbrace{(\langle \eta_a(z_{1:k-1}) \rangle + \langle \eta_{ba}(z_{k+1:K}) \rangle)}_{\eta_{ab}}\right). \tag{A9}$$

For the sake of brevity, the distribution subscripts in the expectation notations are dropped. Note that we evaluate the posterior up to a normalization constant. Nevertheless, the log-normalizer function $A_{\eta_a}(\cdot)$ is readily available for EF distributions having sufficient statistics vector $\phi_a(z_k)$. As a consequence, the posterior evaluates to the following:

$$q(z_k) = h_a(z_k) \exp(\phi_a(z_k)^\intercal \eta_{ab} - A_{\eta_a}(\eta_{ab})). \tag{A10}$$

Having showed that the conjugate factor pairs lead to a closed-form expression for posterior $q(z_k)$, we now investigate which expectation quantities related to $q(z_k)$ are required in the outgoing VMP messages from the factors f_a and f_b to, say z_1 and z_K:

$$\overleftarrow{m}_{z_1}(z_1) \propto \exp\Big(\langle \log f_a(z_{1:k}) \rangle_{q(z_{2:k})}\Big), \tag{A11a}$$

$$\overrightarrow{m}_{z_K}(z_K) \propto \exp\Big(\langle \log f_b(z_{k:K}) \rangle_{q(z_{k:K-1})}\Big). \tag{A11b}$$

In practice, the message $\overleftarrow{m}_{z_1}(z_1)$ is explicitly calculated by isolating the terms with z_1 in a sufficient statistics vector as it is done for z_k in (A8). Similarly, $\overrightarrow{m}_{z_K}(z_K)$ is explicitly calculated, analogous to message $\overrightarrow{m}_{z_k}(z_k)$ in (A7). Here, we follow a rather different approach to explicitly show the expectations related to $q(z_k)$ in the message calculations. Substituting f_a and f_b with (A1) and (A3) in (A11a,b), and keeping in mind that the mean-field assumption allows separation of the expectation quantities with distinct random variables, the messages evaluate to the following:

$$\overleftarrow{m}_{z_1}(z_1) \propto \exp\Big(\langle \phi_a(z_k)\rangle_{q(z_k)}^{\intercal} \langle \eta_a(z_{1:k-1})\rangle_{q(z_{2:k-1})} - \langle A_a(z_{1:k-1})\rangle_{q(z_{2:k-1})}\Big), \tag{A12a}$$

$$\overrightarrow{m}_{z_K}(z_K) \propto h_b(z_K) \exp\Big(\langle \phi_a(z_k)\rangle_{q(z_k)}^{\intercal} \langle \eta_{ba}(z_{k+1:K})\rangle_{q(z_{k+1:K-1})} + \langle c_{ba}(z_{k+1:K})\rangle_{q(z_{k+1:K-1})}\Big). \tag{A12b}$$

Notice that both messages require the expectation of the sufficient statistic vector $\phi_a(z_k)$. Fortunately, in EF distributions, $\langle \phi_a(z_k)\rangle_{q(z_k)}$ is available in closed-form as the gradient of the log-normalizer ([52], Proposition 3.1):

$$\langle \phi_a(z_k)\rangle_{q(z_k)} = \nabla_{\eta_a} A_{\eta_a}|_{\eta_a=\eta_{ab}}. \tag{A13}$$

For the sake of completeness, we now show that this equality holds. Recall that $A_{\eta_a}(\eta_a) = \log(\int h_a(z_k)\exp(\phi_a(z_k)^{\intercal}\eta_a)\mathrm{d}z_k)$. Then,

$$\begin{aligned}
\nabla_{\eta_a} A_{\eta_a}(\eta_a) &= \frac{\nabla_{\eta_a}\int h_a(z_k)\exp(\phi_a(z_k)^{\intercal}\eta_a)\mathrm{d}z_k}{\int h_a(z_k)\exp(\phi_a(z_k)^{\intercal}\eta_a)\mathrm{d}z_k} \\
&= \frac{\int \phi_a(z_k) h_a(z_k)\exp(\phi_a(z_k)^{\intercal}\eta_a)\mathrm{d}z_k}{\exp(A_{\eta_a}(\eta_a))} \\
&= \int \phi_a(z_k) h_a(z_k)\exp(\phi_a(z_k)^{\intercal}\eta_a - A_{\eta_a}(\eta_a))\mathrm{d}z_k.
\end{aligned} \tag{A14}$$

Evaluating this gradient at $\eta_a = \eta_{ab}$, we reach the following:

$$\begin{aligned}
\nabla_{\eta_a} A_{\eta_a}|_{\eta_a=\eta_{ab}} &= \int \phi_a(z_k) \underbrace{h_a(z_k)\exp(\phi_a(z_k)^{\intercal}\eta_{ab} - A_{\eta_a}(\eta_{ab}))}_{q(z_k)} \mathrm{d}z_k \\
&= \langle \phi_a(z_k)\rangle_{q(z_k)}.
\end{aligned} \tag{A15}$$

Appendix A.1.2. Free Energy

As in the message and the posterior calculations, conjugacy eases the free energy calculation. We investigate it for (5), the free energy terms that include z_k. $\mathcal{F}_k$ is decomposed as follows:

$$\mathcal{F}_k = \underbrace{-\mathbb{E}_{q(z_{1:k})}[\log f_a(z_{1:k})] - \mathbb{E}_{q(z_{k:K})}[\log f_b(z_{k:K})]}_{\text{Average energy terms}} + \underbrace{\mathbb{E}_{q(z_k)}[\log q(z_k)]}_{\text{Negative entropy}}. \tag{A16}$$

Substituting f_a, f_b and $q(z_k)$ with (A1), (A3) and (A10) in the above expression:

$$\begin{aligned}
\mathcal{F}_k = &- \langle \log(h_a(z_k))\rangle_{q(z_k)} - \langle \phi_a(z_k)\rangle_{q(z_k)}^{\intercal} \langle \eta_a(z_{1:k-1})\rangle_{q(z_{1:k-1})} + \langle A_a(z_{1:k-1})\rangle_{q(z_{1:k-1})} \\
&- \langle \log(h_b(z_K))\rangle_{q(z_K)} - \langle \phi_a(z_k)\rangle_{q(z_k)}^{\intercal} \langle \eta_{ba}(z_{k+1:K})\rangle_{q(z_{k+1:K})} - \langle c_{ba}(z_{k+1:K})\rangle_{q(z_{k+1:K})} \\
&+ \langle \log(h_a(z_k))\rangle_{q(z_k)} + \langle \phi_a(z_k)\rangle_{q(z_k)}^{\intercal} \eta_{ab} - A_{\eta_a}(\eta_{ab}).
\end{aligned} \tag{A17}$$

The expectation terms related to z_k in $\mathcal{F}_k$ are $\langle \phi_a(z_k)\rangle_{q(z_k)}$ and $\langle \log(h_a(z_k))\rangle_{q(z_k)}$. The former expectation is available in closed form, (A13). Thus $\mathcal{F}_k$ is analytically tractable for those distributions that possess closed-form solution for $\langle \log(h_a(z_k))\rangle_{q(z_k)}$.

In short, conjugate factor pairs facilitate the VMP procedure by allowing closed-form expressions for updates of messages (3), posteriors (4) and FE (5). Moreover, although exceptions exist, similar to the normalization of the posterior (A10), the messages $\overrightarrow{m}_{z_k}(z_k)$ and $\overleftarrow{m}_{z_k}(z_k)$ can be effortlessly normalized if the required expectations are known. Therefore, we can directly parameterize them with standard probability distributions and draw samples from them. This property of EF distributions plays a pivotal role in our automation of the importance sampling procedure.

Appendix A.2. VMP with Non-Conjugate Soft Factor Pairs

Suppose that the soft factors f_a and f_b are no longer conjugate pairs, i.e., f_b given in (A2) can be written in the following form:

$$f_b(z_{k:K}) = h_b(z_K) \exp(\hat{\phi}_b(z_k)^\intercal \hat{\eta}_b(z_{k+1:K}) + \hat{c}_b(z_{k+1:K})), \tag{A18}$$

where crucially $\hat{\phi}_b(z_k) \neq \phi_a(z_k)$. Notice that $\hat{\eta}_b (\neq \eta_b)$ is the natural parameters after the modification of (A2) to isolate the terms with z_k in the sufficient statistics vector. Therefore, the messages $\overrightarrow{m}_{z_k}(z_k)$ and $\overleftarrow{m}_{z_k}(z_k)$ differ in sufficient statistics:

$$\overrightarrow{m}_{z_k}(z_k) \propto h_a(z_k) \exp\Big(\phi_a(z_k)^\intercal \langle \eta_a(z_{1:k-1}) \rangle_{q(z_{1:k-1})}\Big), \tag{A19a}$$

$$\overleftarrow{m}_{z_k}(z_k) \propto \exp\Big(\hat{\phi}_b(z_k)^\intercal \langle \hat{\eta}_b(z_{k+1:K}) \rangle_{q(z_{k+1:K})}\Big). \tag{A19b}$$

In this case, the normalization constant calculation in the posterior update step (5) is not straightforward anymore; and worse, it is often intractable. The term *intractable* refers to integrals that are not available in closed-form for continuous variables. For discrete variables, it refers to summations that are not achievable in a feasible amount of time. The lack of the normalization constant, $\int \overrightarrow{m}_{z_k}(z_k) \overleftarrow{m}_{z_k}(z_k) \mathrm{d}z_k$, hinders the calculation of the expectations with z_k terms that appear in out-going VMP messages from f_a and f_b to variables $z_{1:k-1}$ and $z_{k+1:K}$, respectively, e.g., $\langle \phi_a(z_k) \rangle_{q(z_k)}$ and $\langle \hat{\phi}_b(z_k) \rangle_{q(z_k)}$ in

$$\overleftarrow{m}_{z_1}(z_1) \propto \exp\Big(\langle \phi_a(z_k) \rangle_{q(z_k)}^\intercal \langle \eta_a(z_{1:k-1}) \rangle_{q(z_{2:k-1})} - \langle A_a(z_{1:k-1}) \rangle_{q(z_{2:k-1})}\Big), \tag{A20a}$$

$$\overrightarrow{m}_{z_K}(z_K) \propto h_b(z_K) \exp\Big(\langle \hat{\phi}_b(z_k) \rangle_{q(z_k)}^\intercal \langle \hat{\eta}_b(z_{k+1:K}) \rangle_{q(z_{k+1:K-1})} + \langle \hat{c}_b(z_{k+1:K}) \rangle_{q(z_{k+1:K-1})}\Big). \tag{A20b}$$

As a result, non-conjugacies obstruct the VMP procedure by hampering closed-form expectation calculations. Bear in mind that even though VMP procedure is obstructed due to intractable expectations, the messages are distinctly fixed for soft factors as functions of certain expectation quantities that are supposed to be calculated over their arguments. We use this property in our Extended VMP method.

Appendix A.3. VMP with Composite Nodes

In this subsection, we shed light on the issues with composite nodes that are constructed by composition of EF distribution soft factors and deterministic conditionals, i.e., the following:

$$f_b(z_{k:K}) = \int f_\delta(x, z_k) f_c(x, z_{k+1:K}) \mathrm{d}x, \tag{A21}$$

where $f_\delta(x, z_k) = p(x|z_k) = \delta(x - g(z_k))$ is a deterministic conditional distribution. Composite nodes enable us to build almost arbitrary factor nodes. For example, a mixture likelihood distribution $p(y|z) = \mathcal{N}(y; \mu_1, \sigma^2)^z \mathcal{N}(y; \mu_2, \sigma^2)^{(1-z)}$ with z a one-hot coded selection variable, can be constructed by composing an EF soft factor, a Gaussian, with a deterministic factor as $\int p(y|x) p(x|z) \mathrm{d}x$, where $p(y|x) = \mathcal{N}(y; x, \sigma^2)$ and $p(x|z) = \delta(x - z\mu_1 - (1-z)\mu_2)$. However, composite nodes impose new challenges on inference procedures.

Now, let us try to calculate $q(z_k)$. The forward VMP message $\overrightarrow{m}_{z_k}(z_k)$ is given in (A19a). Suppose that the conjugate prior to EF soft factor f_c for x has the sufficient statistics vector $\hat{\phi}_c(x)$ (see (A6)). Then, the VMP message from f_b to z_k is as follows:

$$\begin{aligned}\overleftarrow{m}_{z_k}(z_k) &\propto \exp\left(\left\langle \log\left(\int \delta(x-g(z_k))\exp\left(\hat{\phi}_c(x)^\intercal \hat{\eta}_c(z_{k+1:K})\right)\mathrm{d}x\right)\right\rangle_{q(z_{k+1:K})}\right)\\ &\propto \underbrace{\int \delta(x-g(z_k))\underbrace{\exp\left(\hat{\phi}_c(x)^\intercal \langle\hat{\eta}_c(z_{k+1:K})\rangle_{q(z_{k+1:K})}\right)}_{\text{VMP: }\overleftarrow{m}_x(x)}\mathrm{d}x}_{\text{BP}}\\ &= \underbrace{\exp\left(\hat{\phi}_c(g(z_k))^\intercal \langle\hat{\eta}_c(z_{k+1:K})\rangle_{q(z_{k+1:K})}\right)}_{\overleftarrow{m}_x(g(z_k))}. \end{aligned} \tag{A22}$$

Note that the above message reduces to VMP message from f_c to x followed by Belief Propagation (BP) [48,49]. The resulting backward message $\overleftarrow{m}_{z_k}(z_k)$ has the sufficient statistics vector $\hat{\phi}_c(g(z_k))$. If $\hat{\phi}_c(\cdot) = \phi_a(g^{-1}(\cdot))$, this case reduces to ordinary VMP as discussed in Appendix A.1; otherwise this case is a special case of Appendix A.2 and $q(z_k)$ is not available in closed-form. Hence, the outgoing messages from the factor nodes f_a and f_b:

$$\overleftarrow{m}_{z_1}(z_1) \propto \exp\left(\langle\phi_a(z_k)\rangle_{q(z_k)}^\intercal \langle\eta_a(z_{1:k-1})\rangle_{q(z_{2:k-1})} - \langle A_a(z_{1:k-1})\rangle_{q(z_{2:k-1})}\right), \tag{A23a}$$

$$\begin{aligned}\overrightarrow{m}_{z_K}(z_K) &\propto \exp\left(\left\langle \log\left(\int \delta(x-g(z_k))\right.\right.\right.\\ &\qquad \left.\left.\left. h_c(z_K)\exp\left(\hat{\phi}_c(x)^\intercal \hat{\eta}_c(z_{k+1:K}) + \hat{c}_c(z_{k+1:K})\right)\mathrm{d}x\right)\right\rangle_{q(z_{k:K-1})}\right)\\ &= h_c(z_K)\exp\left(\left\langle\hat{\phi}_c(g(z_k))\right\rangle_{q(z_k)}^\intercal \langle\hat{\eta}_c(z_{k+1:K})\rangle_{q(z_{k+1:K-1})} + \langle\hat{c}_c(z_{k+1:K})\rangle_{q(z_{k+1:K-1})}\right)\\ &= h_c(z_K)\exp\left(\left\langle\hat{\phi}_c(x)\right\rangle_{q(x)}^\intercal \langle\hat{\eta}_c(z_{k+1:K})\rangle_{q(z_{k+1:K-1})} + \langle\hat{c}_c(z_{k+1:K})\rangle_{q(z_{k+1:K-1})}\right)\end{aligned} \tag{A23b}$$

are intractable. The last line in the above derivations follows from the transformation of variables [53], i.e., $q(z_k) = q(x)|\frac{\mathrm{d}x}{\mathrm{d}z_k}|$, and expose the automatable nature of Variational Message Passing: the VMP message $\overrightarrow{m}_{z_K}(z_K)$ requires expectation quantities that are related to arguments of the soft factor z_K is tied to, which is in this case f_c. Therefore, once the VMP message passing rule is defined for the factor f_c as a function of its arguments, we can instantiate the messages by providing the required expectation quantities. For example, the required expectation quantities related to argument x are contained in the sufficient statistics vector $\hat{\phi}_c(x)$.

Appendix B. Derivation of Extended VMP

Here, we show the details of our solution approach that is based on importance sampling (IS) and Laplace approximation. First, we address the issues with deterministic mappings of random variables. The resulting technique emerges as a remedy for non-conjugate soft factor pairs problem as well.

Appendix B.1. Deterministic Mappings with Single Inputs

We first address the issues with single-input deterministic mappings and generalize our solution to multiple inputs later on. Consider the sub-graph given in Figure A1, where the deterministic conditional $p(x|z_k)$ is defined as $f_\delta(x, z_k) = \delta(x - g(z_k))$. As derived in (A23a,b), we need the expectations $\langle\phi_a(z_k)\rangle_{q(z_k)}$ and $\langle\hat{\phi}_c(x)\rangle_{q(x)}$ to calculate VMP messages towards edges $z_{1:k-1}$ and $z_{k+1:K}$, respectively. Suppose that $\Phi(\cdot)$ is an element in the sufficient statistic vectors $\phi_a(z_k)$ and $\hat{\phi}_c(x)$. Then we need to be able to calculate $\langle\Phi(x)\rangle_{q(x)}$ and $\langle\Phi(z_k)\rangle_{q(z_k)}$. Let us start with evaluating $\langle\Phi(x)\rangle_{q(x)}$ first:

$$\langle \Phi(x) \rangle_{q(x)} = \int q(x)\Phi(x)\mathrm{d}x = \int q(z_k)\Phi(g(z_k))\mathrm{d}z_k.$$

The second equality in the above expression is due to the transformation of variables, i.e., $q(x) = q(z_k)|\frac{\mathrm{d}z_k}{\mathrm{d}x}|$ [53].

Substituting $q(z_k)$ with (4) in the above integral yields the following:

$$\langle \Phi(x) \rangle_{q(x)} = \int \frac{\overrightarrow{m}_{z_k}(z_k)\overleftarrow{m}_{z_k}(z_k)}{\int \overrightarrow{m}_{z_k}(z_k)\overleftarrow{m}_{z_k}(z_k)\mathrm{d}z_k}\Phi(g(z_k))\mathrm{d}z_k, \tag{A24}$$

where $\overleftarrow{m}_{z_k}(z_k) = \overleftarrow{m}_{x}(g(z_k))$, as given in (A22). Recall from Appendix A.3 that the normalizer, $\int \overrightarrow{m}_{z_k}(z_k)\overleftarrow{m}_{z_k}(z_k)dz_k$, is often hard to calculate analytically.

We use importance sampling [11,36] to approximate the integral in (A24):

$$\begin{aligned}\langle \Phi(x) \rangle_{q_x} &= \int \frac{\frac{\overrightarrow{m}_{z_k}(z_k)\overleftarrow{m}_{x}(g(z_k))\pi(z_k)}{\pi(z_k)}}{\int \frac{\overrightarrow{m}_{z_k}(z_k)\overleftarrow{m}_{x}(g(z_k))\pi(z_k)}{\pi(z_k)}\mathrm{d}z_k}\Phi(g(z_k))\mathrm{d}z_k \\ &\approx \sum_{i=1}^{N} \underbrace{\left[\frac{\frac{\overrightarrow{m}_{z_k}(z_k^{(i)})\overleftarrow{m}_{x}(g(z_k^{(i)}))}{\pi(z_k^{(i)})}}{\sum_{j=1}^{N} \frac{\overrightarrow{m}_{z_k}(z_k^{(j)})\overleftarrow{m}_{x}(g(z_k^{(j)}))}{\pi(z_k^{(j)})}} \right]}_{w^{(i)}} \Phi(g(z_k^{(i)})),\end{aligned} \tag{A25}$$

where $z_k^{(i)}$ for $i = 1, \ldots, N$ are drawn from the proposal distribution $\pi(z_k)$, i.e., $z_k^{(i)} \sim \pi(z_k)$. $g(z_k^{(i)})$ for $i = 1, \ldots, N$ are particles and their corresponding weights are denoted by $w^{(i)}$.

The design of a good proposal distribution has a critical role in IS. First, it is supposed to be an easy-to-sample distribution. Secondly, its support is required to be no smaller than the support of $\overrightarrow{m}_{z_k}(z_k)\overleftarrow{m}_{x}(g(z_k))$ [36]. Lastly, the proposal distribution is desired to be a good representation of $q_k(z_k) = \frac{\overrightarrow{m}_{z_k}(z_k)\overleftarrow{m}_{z_k}(z_k)}{\int \overrightarrow{m}_{z_k}(z_k)\overleftarrow{m}_{z_k}(z_k)\mathrm{d}z_k}$ to attain a fast convergence [47]. In our automated design, $\overrightarrow{m}_{z_k}(z_k)$ constitutes the proposal distribution. Our choice is not optimal in a sense that information regarding the evidence is most often carried out by the backward message and it is not incorporated in our proposal design. However, $\overrightarrow{m}_{z_k}(z_k)$ satisfies the first two conditions since the messages are parameterized with standard distributions (easy-to-sample) and it has nonzero probability everywhere that the posterior has, too. Substituting $\pi(z_k)$ with $\overrightarrow{m}_{z_k}(z_k)$ in (A25) yields the following:

$$\widehat{\langle \Phi(x) \rangle}_{q(x)} = \sum_{i=1}^{N} \underbrace{\left[\frac{\overleftarrow{m}_{x}(g(z_k^{(i)}))}{\sum_{j=1}^{N} \overleftarrow{m}_{x}(g(z_k^{(j)}))} \right]}_{w^{(i)}} \Phi(g(z_k^{(i)})), \tag{A26}$$

where $z_k^{(i)} \sim \overrightarrow{m}_{z_k}(z_k)$ for $i = 1, \ldots, N$ and $\widehat{\langle \Phi(x) \rangle}_{q(x)}$ denotes our estimator for $\langle \Phi(x) \rangle_{q_x}$.

Let us summarize the procedure in (A26) to define our first set of rules related to the deterministic nodes. (A26) consists of samples that are drawn from $\overrightarrow{m}_{k}(z_k)$ and transformed through deterministic mapping $g(.)$. We cast this process as the forward message $\overrightarrow{m}_{x}(x)$ calculation. Once the samples are transformed, i.e., $g(z_k^{(i)})$, the weights $w^{(i)}$ are determined over $\overleftarrow{m}_{x}(.)$. We interpret this process as the collision of the forward $\overrightarrow{m}_{x}(x)$ and the backward messages $\overleftarrow{m}_{x}(.)$; hence, we relate it to the posterior calculation. Setting $\Phi(g(z_k^{(i)}))$ to $\delta(x - g(z_k^{(i)}))$, our interpretation of message collision becomes obvious since

$\Phi(g(z_k^{(i)})) := \delta(x - g(z_k^{(i)}))$ results in a Monte Carlo estimate for $q(x)$. As a result, we introduce our first set of rules related to deterministic nodes and the posterior approximation at the output edge of the deterministic node:

$$\overrightarrow{m}_x(x) \approx \left\{ \left(\frac{1}{N}, g(z_k^{(1)}) \right), \ldots, \left(\frac{1}{N}, g(z_k^{(N)}) \right) \right\}, \tag{A27a}$$

$$q(x) \propto \overrightarrow{m}_x(x) \cdot \overleftarrow{m}_x(x) \approx \left\{ \left(w_x^{(1)}, x^{(1)} \right), \ldots, \left(w_x^{(N)}, x^{(N)} \right) \right\}, \tag{A27b}$$

$$\text{where } z_k^{(i)} \sim \overrightarrow{m}_k(z_k), x^{(i)} = g(z_k^{(i)}), w_x^{(i)} = \frac{\overleftarrow{m}_x(x^{(i)})}{\sum_{j=1}^{N} \overleftarrow{m}_x(x^{(i)})}. \tag{A27c}$$

Here, we introduce the term *list of weighted samples* (LWS) to refer to the distributions that are represented by a set of samples and corresponding weights. Above, $\overrightarrow{m}_x(x)$ and $q(x)$ are represented by LWS distributions.

Now, we turn our attention to calculation of $q(z_k)$ and the expectation quantity $\langle \Phi(z_k) \rangle_{q(z_k)}$. For this task we have two different strategies: if $\overrightarrow{m}_{z_k}(z_k)$ is a Gaussian message, we approximate $q(z_k)$ by Laplace approximation which is also automatable thanks to automatic differentiation and otherwise we follow the IS procedure introduced above. Let us go over them starting from the latter.

Appendix B.1.1. Non-Gaussian Case

This time we are supposed to evaluate $\langle \Phi(z_k) \rangle_{q(z_k)}$ so that the VMP messages toward $z_1, \ldots, z_{k-1}$ can be computed. Notice that the procedure is exactly same with (A25), except that the expectation quantity of interest, $\langle \Phi(z_k) \rangle_{q(z_k)}$, does not involve the deterministic mapping $g(.)$, this time. Therefore, by using $\overrightarrow{m}_{z_k}(z_k)$ as the proposal distribution, we can estimate $\langle \Phi(z_k) \rangle_{q(z_k)}$ as the following:

$$\widehat{\langle \Phi(z_k) \rangle}_{q(z_k)} = \sum_{i=1}^{N} \underbrace{\left[\frac{\overleftarrow{m}_x(g(z_k^{(i)}))}{\sum_{j=1}^{N} \overleftarrow{m}_x(g(z_k^{(j)}))} \right]}_{w^{(i)}} \Phi(z_k^{(i)}). \tag{A28}$$

This gives us the second set of rules related to deterministic mappings. An element of this new set of rules is that the backward message is directly passed in probability distribution function (pdf) form:

$$\overleftarrow{m}_{z_k}(z_k) = \overleftarrow{m}_x(g(z_k)). \tag{A29}$$

Recall from Appendix A.1 that the messages used to carry standard EF distributions. Now, we make an exception and introduce $\overleftarrow{m}_{z_k}(z_k)$, which is no longer associated with any of the standard EF distributions. Nonetheless, $\overleftarrow{m}_{z_k}(z_k)$ takes an exponential form since $\overleftarrow{m}_x(\cdot)$ is an EF distribution (see (A22)). Therefore, we call $\overleftarrow{m}_{z_k}(z_k)$ a *non-standard exponential family* (NEF) distribution. Having defined the backward message, let us evaluate the posterior $q(z_k)$. Similar to $q(x)$, substituting $\Phi(z_k)$ with $\delta(z_k - z_k^{(i)})$ in (A28) gives us a Monte Carlo estimate of $q(z_k)$:

$$q(z_k) \propto \overrightarrow{m}_{z_k}(z_k) \cdot \overleftarrow{m}_{z_k}(z_k) \approx \left\{ \left(w_{z_k}^{(1)}, z_k^{(1)} \right), \ldots, \left(w_{z_k}^{(N)}, z_k^{(N)} \right) \right\}, \tag{A30a}$$

$$\text{where } z_k^{(i)} \sim \overrightarrow{m}_{z_k}(z_k), w_{z_k}^{(i)} = \frac{\overleftarrow{m}_{z_k}(z_k^{(i)})}{\sum_{j=1}^{N} \overleftarrow{m}_{z_k}(z_k^{(j)})}. \tag{A30b}$$

Appendix B.1.2. Gaussian Case

In FFGs, the models are often constructed in such a way that the most prevailing message types will be Gaussians. This is because Gaussian messages facilitate inference by allowing many inference related operations to be executed in closed form, such as summation, conditioning, scaling and shifting by constants, etc. In order to retain the computational advantages of Gaussian distribution, we take it as an implicit hint that the posterior distribution is *Gaussian-like*, if $\overrightarrow{m}_{z_k}(z_k)$ is a Gaussian message. Then, we use Laplace approximation ([12], Section 4.4) to approximate $q(z_k)$ with $\mathcal{N}(z_k; \mu_{z_k}, V_{z_k})$:

$$\mu_{z_k} = \arg\max_{z_k} \left(\log \overrightarrow{m}_{z_k}(z_k) + \log \overleftarrow{m}_{z_k}(z_k)\right), \tag{A31a}$$

$$V_{z_k} = \left(-\nabla\nabla_{z_k}(\log \overrightarrow{m}_{z_k}(z_k) + \log \overleftarrow{m}_{z_k}(z_k))|_{z_k=\mu_k}\right)^{-1}, \tag{A31b}$$

where $\nabla_{z_k} f$ denotes the gradient of f with respect to z_k and $\nabla\nabla_{z_k} f|_{z_k=\mu_k}$ refers to the Hessian of f with respect to z_k evaluated at μ_k. Note that the gradient and the Hessian respectively reduce to the first and the second derivatives if z_k is scalar. Laplace approximation is a mode-seeking algorithm. We use automatic differentiation (autodiff) [13] to evaluate the gradient $\nabla_{z_k}(\log \overrightarrow{m}_{z_k}(z_k) + \log \overleftarrow{m}_{z_k}(z_k))$ and employ it in a gradient-ascent algorithm to seek the mode (we supply the implementation details in Appendix D). Once the mode is reached, we evaluate the Hessian at the mode to fit the variance term for our Gaussian approximation.

The assumption we make here that $\overrightarrow{m}_{z_k}(z_k)$ implies a *Gaussian-like* $q(z_k)$ paves the way of automating many well known inference procedures achieved through Laplace approximation, such as Bayesian logistic regression ([37], Section 8.4), Laplace-Gaussian filtering and smoothing in state space models [54], Poisson Linear Dynamical Systems [55], etc. However, our assumption would not be appropriate for all configurations. For example, Gaussian prior on rate parameter of a Poisson distribution would result in an ambiguous posterior since the domain of the rate is the positive real numbers while the Gaussian approximated posterior has a support on the entire real axis. A better model specification could be achieved by mapping a Gaussian distributed random variable to the rate parameter through an inverse-link function, exp in this example. Likewise, a multi-modal backward message $\overleftarrow{m}_{z_k}(z_k)$ with a support on real numbers often yields a multi-modal posterior which can be better captured with particle methods. (In Appendix E, we show that it is possible to run particle filtering through Gaussian factor nodes in our technique.)

In summary, our method resorts to Laplace approximation to approximate $q(z_k)$ with a Gaussian distribution whenever $\overrightarrow{m}_{z_k}(z_k)$ is Gaussian. Therefore, the user of our method must keep in mind the consequences of prior choices and build her model accordingly.

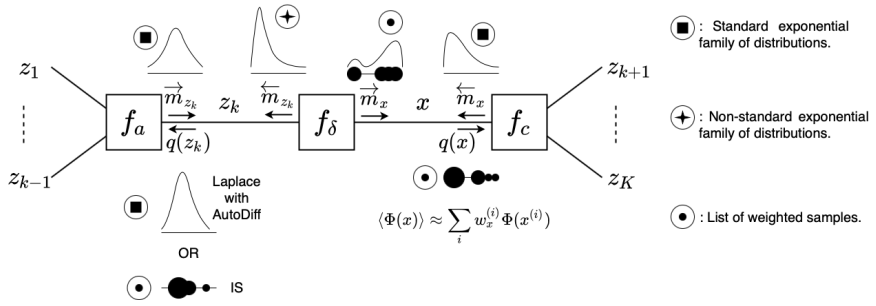


Figure A2. Messages around a deterministic node are visualized together with posterior approximations. In EVMP algorithm, forward messages from single input deterministic nodes are approximated by LWS representations. Backward messages, on the other hand, take non-standard exponential family distribution forms.

The overall procedure for single input deterministic functions is depicted in Figure A2. In the next subsection, we extend this procedure to multiple input deterministic mappings.

Appendix B.2. Deterministic Mappings with Multiple Inputs

Consider the deterministic node, $f_\delta(x, z_{1:k}) = \delta(x - g(z_{1:k}))$, given in Figure A3 where the inputs to the deterministic function $g(.)$ are $z_1, \ldots, z_k$ and the output is x.

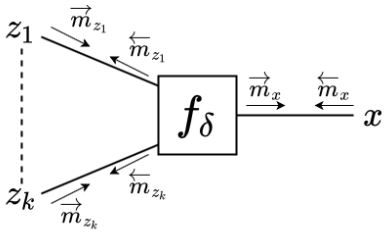


Figure A3. A deterministic node with inputs $z_1, \ldots, z_k$ and output x.

Before starting the discussion on the backward messages, let us define the forward message $\overrightarrow{m}_x(x)$. Analogous to the single input case, we define $\overrightarrow{m}_x(x)$ with an LWS as the following:

$$\overrightarrow{m}_x(x) \approx \left\{ \left(\tfrac{1}{N}, g(z_{1:k}^{(1)})\right), \ldots, \left(\tfrac{1}{N}, g(z_{1:k}^{(N)})\right) \right\}, \quad \text{where } z_1^{(i)} \sim \overrightarrow{m}_{z_1}(z_k), \ldots, z_k^{(i)} \sim \overrightarrow{m}_{z_k}(z_k). \tag{A32}$$

Once the message is calculated as a set of equally weighted samples, we scale the weights according to the importance score of their corresponding samples to represent $q(x)$:

$$q(x) \propto \overrightarrow{m}_x(x) \cdot \overleftarrow{m}_x(x) \approx \left\{ \left(w_x^{(1)}, x^{(1)}\right), \ldots, \left(w_x^{(N)}, x^{(N)}\right) \right\}, \tag{A33a}$$

$$\text{where } x^{(i)} = g(z_{1:k}^{(i)}), w_x^{(i)} = \frac{\overleftarrow{m}_x(x^{(i)})}{\sum_{j=1}^{N} \overleftarrow{m}_x(x^{(i)})}. \tag{A33b}$$

Now, let us define the backward messages propagated by the deterministic node. The exact backward message towards one of the input variables, say z_k, is the following:

$$\overleftarrow{m}_{z_k}(z_k) = \int \delta(x - g(z_{1:k}))\, \overrightarrow{m}_{z_1}(z_1) \ldots \overrightarrow{m}_{z_{k-1}}(z_{k-1})\, \overleftarrow{m}_x(x) \mathrm{d}z_1 \ldots \mathrm{d}z_{k-1} \mathrm{d}x. \tag{A34}$$

Unfortunately, the above integral is often intractable. Even if all the variables are discrete and integral is replaced by summation, it becomes intractable in practice as the number of variables increases. Here, we address this issue with two different approximation strategies. As it is in the above subsection, type of the approximation depends on the incoming messages to the deterministic node from the input edges: if the messages $\overrightarrow{m}_{z_1}(z_1), \ldots, \overrightarrow{m}_{z_k}(z_k)$ are all Gaussian, we approximate the joint posterior distribution of $z_1, \ldots, z_k$ by a Gaussian distribution. Then, we calculate the backward messages over the approximated joint posterior and incoming messages. Otherwise, we use Monte Carlo summation. Let us start with the latter case.

Appendix B.2.1. Monte Carlo Approximation to the Backward Message

Monte Carlo approximation to the the integral in (A34) is

$$\overleftarrow{m}_{z_k}(z_k) \approx \frac{1}{N} \sum_{i=1}^{N} \overleftarrow{m}_x(g(z_1^{(i)}, \ldots, z_{k-1}^{(i)}, z_k)), \tag{A35}$$

where $z_j^{(i)} \sim \overrightarrow{m}_{z_j}(z_j)$ for $j = 1, \ldots, k-1$.

Once the message $\overleftarrow{m}_{z_k}(z_k)$ is approximately calculated and propagated as an NEF distribution, $q(z_k)$ is also approximated either by IS or Laplace, depending on the message type $\overrightarrow{m}_{z_k}(z_k)$ as it is discussed in Appendix B.1.

Appendix B.2.2. Gaussian Approximation to the Backward Message

The above procedure yields two consecutive approximation processes in the calculation of $q(z_k)$. Considering that we assumed $\overrightarrow{m}_{z_k}(z_k)$ implies that $q(z_k)$ is *Gaussian-like*, we can avoid the approximation in (A35) if all the incoming messages $\overrightarrow{m}_{z_1}(z_1), \ldots, \overrightarrow{m}_{z_k}(z_k)$ are Gaussian. We achieve this by approximating the joint posterior $q(z_{1:k})$ with Laplace, followed by a marginalization to evaluate $q(z_k)$ and $\overleftarrow{m}_{z_k}(z_k) \propto q(z_k)/\overrightarrow{m}_{z_k}(z_k)$.

More precisely, consider the incoming messages $\overrightarrow{m}_{z_j}(z_j) = \mathcal{N}(z_j; \mu'_{z_j}, V'_{z_j})$ for $j = 1, \ldots, k$. Note that these messages carry posterior beliefs on $z_1, \ldots, z_k$, which can be represented with a joint belief $\overrightarrow{m}_{z_{1:k}}(z_{1:k})$ constituted by concatenation of $z_1, \ldots, z_k$: $z_{1:k} = z_1 \oplus z_2 \oplus \ldots \oplus z_k = \begin{bmatrix} z_1 & \ldots & z_k \end{bmatrix}^\mathsf{T}$:

$$\begin{aligned} \overrightarrow{m}_{z_{1:k}}(z_{1:k}) &= \mathcal{N}(\mu'_{z_{1:k}}, V'_{z_{1:k}}) \\ &= \mathcal{N}\left(\begin{bmatrix} \mu'_{z_1} \\ \vdots \\ \mu'_{z_k} \end{bmatrix}, \begin{bmatrix} V'_{z_1} & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & V'_{z_k} \end{bmatrix} \right). \end{aligned} \tag{A36}$$

Now, we approximate $q(z_{1:k})$ by a Gaussian distribution $\mathcal{N}(\mu_{z_{1:k}}, V_{z_{1:k}})$ with a Laplace approximation:

$$\mu_{z_{1:k}} = \underset{z_{1:k}}{\operatorname{argmax}} \log \overrightarrow{m}_{z_{1:k}}(z_{1:k}) + \log \overleftarrow{m}_x(g(z_1, \ldots, z_k)), \tag{A37a}$$

$$V_{z_{1:k}} = (-\nabla\nabla_{z_{1:k}}(\log \overrightarrow{m}_{z_{1:k}}(z_{1:k}) + \log \overleftarrow{m}_x(g(z_1, \ldots, z_k)))|_{z_{1:k}=\mu_{z_{1:k}}})^{-1}. \tag{A37b}$$

By marginalizing out $z_1, \ldots z_{k-1}$, we find $q(z_k)$:

$$q(z_k) = \int q(z_{1:k}) \mathrm{d}z_1 \ldots \mathrm{d}z_{k-1} = \mathcal{N}(\mu_{z_k}, V_{z_k}). \tag{A38}$$

Recall that $q(z_k) \propto \overrightarrow{m}_{z_k}(z_k)\overleftarrow{m}_{z_k}(z_k)$. This yields the following backward message

$$\overleftarrow{m}_{z_k}(z_k) \propto q(z_k)/\overrightarrow{m}_{z_k}(z_k) = \mathcal{N}(\mu''_{z_k}, V''_{z_k}), \tag{A39}$$

where $V''^{-1}_{z_k} = V^{-1}_{z_k} - V'^{-1}_{z_k}$ and $V''^{-1}_{z_k}\mu''_{z_k} = V^{-1}_{z_k}\mu_{z_k} - V'^{-1}_{z_k}\mu'_{z_k}$. Note that we intentionally parameterize the Gaussian backward message $\overleftarrow{m}_{z_k}(z_k)$ with a precision-weighted mean $V''^{-1}_{z_k}\mu''_{z_k}$ and precision $V''^{-1}_{z_k}$. The canonical parameterization (weighted-mean and precision) brings computational advantages, especially in state space models, by avoiding certain matrix inversions [15]. The approach that we introduced in this section resembles Expectation Propagation (EP) [50,51] in the sense that we first find the posterior, $q(z_k)$, and then the backward message is evaluated by dividing the posterior to the incoming message. As it is stated in [51], Laplace Propagation [56] proposes an iterative Laplace approximation approach to mitigate intractable integral issues that sometimes emerge in EP.

So far, we have discussed how to extend VMP to those models with deterministic conditional distributions. To summarize, the resulting technique approximates the forward messages in deterministic nodes by LWS. Backward messages, on the other hand, are either directly propagated in NEF form or approximated with Gaussian distributions. We also showed posterior approximations related to these message types. In the next subsection, we attack the problem regarding the non-conjugate soft factor pairs.

Appendix B.3. Non-Conjugate Soft Factor Pairs

Next, we address the problems defined in Appendix A.2. Consider the generic edge depicted in Figure 2. Suppose that the messages $\overrightarrow{m}_{z_k}(z_k)$ and $\overleftarrow{m}_{z_k}(z_k)$ differ in sufficient statistics and hence the normalization constant $\int \overrightarrow{m}_{z_k}(z_k)\overleftarrow{m}_{z_k}(z_k)dz_k$ is analytically intractable. Recall that the very much same problem emerges in Appendix B.1 while calculating $q(z_k)$. Therefore, the approximation rules defined in Appendix B.1 applies to non-conjugate factor pairs, as well. For the sake of comprehensiveness, the rules are summarized below.

1. If $\overrightarrow{m}_{z_k}(z_k)$ is a Gaussian message, apply Laplace to approximate $q(z_k)$ with a Gaussian distribution as in (A31a,b).
2. Otherwise, use IS as in (A30a,b).

Appendix C. Free Energy Approximation

Recall from Section 2 that variational inference transforms a difficult inference task to an easier optimization problem of a variational bound called the free energy $\mathcal{F}$. Considering the fact that VMP converges to a stationary point by updating one posterior factor at a time, we anticipate that our approximations approach near the local optima.

As it is shown inAppendix A.1.2, the free energy is amenable to analytical calculations for those models that are solely comprised of conjugate factor pairs. The models that we address here do not allow the free energy to be calculated analytically. This is because analytically intractable expectation quantities, which complicates VMP in practice, also appear in the free energy calculation. Therefore, we provide an approximate free energy to the user so that they can track the convergence of the inference and also make a model comparison [9,57].

We introduce our free energy approximation approach over the sub-graph given in Figure A1, where $f_a(z_{1:k})$ is a standard EF distribution (A1) and $f_b(z_{k:K})$ is a composite node, i.e., $f_b(z_{k:K}) = \int \delta(x - g(z_k))f_c(x, z_{k+1:K})\mathrm{d}x = f_c(g(z_k), z_{k+1:K})$. Recall that $f_c(x, z_{k+1:K})$ is modified as follows:

$$f_c(x, z_{k+1:K}) = h_c(z_K)\exp(\hat{\phi}_c(x)^\intercal \hat{\eta}_c(z_{k+1:K}) + \hat{c}_c(z_{k+1:K})).$$

This sub-graph is a part of a larger FFG. First, we decompose the free energy as the following:

$$\mathcal{F} = \tilde{\mathcal{F}} \underbrace{\underbrace{-\mathbb{E}_{q(z_{1:k})}[\log f_a(z_{1:k})] - \mathbb{E}_{q(z_{k:K})}[\log f_b(z_{k:K})]}_{\text{Average energy terms}} + \underbrace{\mathbb{E}_{q(z_k)}[\log q(z_k)]}_{\substack{\text{Negative} \\ \text{entropy}}}}_{\mathcal{F}_k}, \tag{A40}$$

where $\tilde{\mathcal{F}}$ stands for the free energy terms that are not subject to variables z_k. Explicitly writing the average energy terms, we have the following:

$$\begin{aligned} -\mathbb{E}_{q(z_{1:k})}[\log f_a(z_{1:k})] &= -\langle \log(h_a(z_k))\rangle_{q(z_k)} - \langle \phi_a(z_k)\rangle_{q(z_k)}^\intercal \langle \eta_a(z_{1:k-1})\rangle_{q(z_{1:k-1})} \\ &\quad + \langle A_a(z_{1:k-1})\rangle_{q(z_{1:k-1})} \end{aligned} \tag{A41a}$$

$$\begin{aligned} -\mathbb{E}_{q(z_{k:K})}[\log f_b(z_{k:K})] &= -\Big\langle \log\Big(\int \delta(x - g(z_k)) \\ &\qquad h_c(z_K)\exp\big(\hat{\phi}_c(x)^\intercal \hat{\eta}_c(z_{k+1:K}) + \hat{c}_c(z_{k+1:K})\big)\mathrm{d}x\Big)\Big\rangle_{q(z_{k:K})} \\ &= -\langle \log(h_c(z_K))\rangle_{q(z_K)} - \langle \hat{\phi}_c(x)\rangle_{q(x)}^\intercal \langle \hat{\eta}_c(z_{k+1:K})\rangle_{q(z_{k+1:K})} \\ &\quad - \langle \hat{c}_c(z_{k+1:K})\rangle_{q(z_{k+1:K})}. \end{aligned} \tag{A41b}$$

The above derivations closely follow the derivations in (A23a,b). Note that the expectation terms regarding z_k in (A41b) are substituted by the expectations related to x,

which are contained in the sufficient statistics vector $\hat{\phi}_c(x)$. This quantities are exactly same with the ones required to calculate VMP messages towards $z_{k+1:K}$, and we used IS to estimate them in (A26). Therefore, $\left\langle \hat{\phi}_c(x) \right\rangle_{q(x)}$ for the estimation of $-\mathbb{E}_{q(z_{k:K})}[\log f_b(z_{k:K})]$ is readily available.

Next, we investigate the terms related to z_k in $-\mathbb{E}_{q(z_{1:k})}[\log f_a(z_{1:k})]$. Recall that for $q(z_k)$, we have two approximation methods: (1) a Gaussian approximation to $q(z_k)$ with Laplace, (2) an LWS approximation. $q(z_k)$ is approximated with a Gaussian when $\overrightarrow{m}_{z_k}(z_k)$ is a Gaussian and this is the case if the factor node $f_a(z_{1:k}) = p(z_k|z_{1:k-1})$ is a Gaussian distribution. In this case, $\phi_a(z_k) = [z_k, z_k^2]^\intercal$ ($\phi_a(z_k) = [z_k, z_k z_k^\intercal]^\intercal$ for a multivariate Gaussian), $\log(h_a(z_k)) = -0.5\log(2\pi)$ ($\log(h_a(z_k)) = -0.5d\log(2\pi)$ for a d-dimensional multivariate Gaussian), and $\left\langle \log(h_a(z_k)) \right\rangle_{q(z_k)}$, $\left\langle \phi_a(z_k) \right\rangle_{q(z_k)}$ are available in closed-form. Similarly, the entropy term $-\mathbb{E}_{q(z_k)}[\log q(z_k)]$ is available in closed-form for a Gaussian $q(z_k)$. This completes the calculation of the expectation terms with z_k in $\mathcal{F}_k$.

In the case that $q(z_k)$ is approximated with LWS as in (A30a,b), we approximate $\left\langle \log(h_a(z_k)) \right\rangle_{q(z_k)}$ and $\left\langle \phi_a(z_k) \right\rangle_{q(z_k)}$ with IS as in (A28). Therefore, the approximations for the average energy terms are straightforward. For LWS approximated $q(z_k)$, the main difficulty in the estimation of $\mathcal{F}_k$ stems from the entropy calculation. This is because $\log(q(z_k))$ does not persist in functional form. The entropy approximation for weighted sample approximated distributions is often carried out by probability density estimates on weighted samples [58]. Fortunately, in our case, we do not need to fit a density estimate on LWS since the messages $\overrightarrow{m}_{z_k}(z_k)$ and $\overleftarrow{m}_{z_k}(z_k)$ afford the information regarding the density $q(z_k)$. Let us derive an estimator for the entropy $\mathcal{H}_{z_k}$:

$$\begin{aligned}
\mathcal{H}_{z_k} &= -\int q(z_k)\log q(z_k) dz_k \\
&= -\int q(z_k)\log\left(\frac{\overrightarrow{m}_{z_k}(z_k)\overleftarrow{m}_{z_k}(z_k)}{\int \overrightarrow{m}_{z_k}(z_k)\overleftarrow{m}_{z_k}(z_k)\mathrm{d}z_k}\right)\mathrm{d}z_k \\
&= \underbrace{-\int q(z_k)\log(\overrightarrow{m}_{z_k}(z_k)\overleftarrow{m}_{z_k}(z_k))\mathrm{d}z_k}_{H^1_{z_k}} + \underbrace{\int q(z_k)\log\left(\int \overrightarrow{m}_{z_k}(z_k)\overleftarrow{m}_{z_k}(z_k)\mathrm{d}z_k\right)\mathrm{d}z_k}_{H^2_{z_k}}.
\end{aligned} \tag{A42}$$

We estimate the first term with the following Monte Carlo summation:

$$\hat{\mathcal{H}}^1_{z_k} = -\sum_{i=1}^{N} w^{(i)}_{z_k}\log\left(\overrightarrow{m}_{z_k}(z^{(i)}_k)\overleftarrow{m}_{z_k}(z^{(i)}_k)\right). \tag{A43}$$

The term with the log in $\mathcal{H}^2_{z_k}$ is constant since z_k is integrated out inside the log. Therefore $\mathcal{H}^2_{z_k}$ simplifies further:

$$\mathcal{H}^2_{z_k} = \log\left(\int \overrightarrow{m}_{z_k}(z_k)\overleftarrow{m}_{z_k}(z_k)\mathrm{d}z_k\right)\underbrace{\int q(z_k)\mathrm{d}z_k}_{1}. \tag{A44}$$

Recall from (A28) that the samples $z^{(1)}_k, \ldots, z^{(N)}_k$ are drawn from the message $\overrightarrow{m}_{z_k}(z_k)$. Therefore, the Monte Carlo estimate of $H^2_{z_k}$ is as follows:

$$\hat{\mathcal{H}}^2_{z_k} = \log\left(\frac{1}{N}\sum_{i=1}^{N}\overleftarrow{m}_{z_k}(z^{(i)}_k)\right). \tag{A45}$$

This completes the estimation of the terms with z_k in $\mathcal{F}_k$.

Appendix D. Implementation Details in ForneyLab

Our extensions to VMP are readily available in ForneyLab [8], which is a Julia package for message passing based probabilistic programming. In this section, we provide the reader with some of the core implementation details and automation process of the method. First, the number of particles that commutes through deterministic nodes is set to 1000 by default. The user can change the number of samples during model specification. Similarly, the posterior of the variables that are connected to non-conjugate soft factor pairs are approximated by 1000 samples. For Laplace approximations, gradients are automatically calculated by automatic differentiation tools of Julia language. We use the *ForwardDiff* package [59] since it is a mature, universal automatic differentiation tool that aligns well with the needs of our approach.

Appendix E. Bonus: Bootstrap Particle Filtering

Having implemented importance sampling to get around the complications in VMP, we now show how our technique inherently supports bootstrap particle filtering in state space models [36,38].

Recall that we automate Laplace approximation to retain the computational convenience of Gaussian filtering and smoothing. Although this choice sounds reasonable for those cases, we believe that the distributions over hidden states possess unimodal behavior, so it would not be sufficient to capture multi-modal distributions [36]. Similarly, due to non-linearities in the model specification and/or non-Gaussian process noise, Gaussian distribution might not be a plausible representation of the hidden states. In these cases, Sequential Monte Carlo methods [60] could be appealing because they flexibly recover asymmetric and mixture distributions.

In VMP setting, our method employs samples and their corresponding weights to deploy VMP messages which are parameterized by exponential family distributions. Alternatively, in Belief Propagation (BP) setting, a soft factor collects samples to instantiate the conditional distributions, and then draw samples from these conditionals. This process is depicted in Figure A4 with two samples for the sake of ease of visualization.

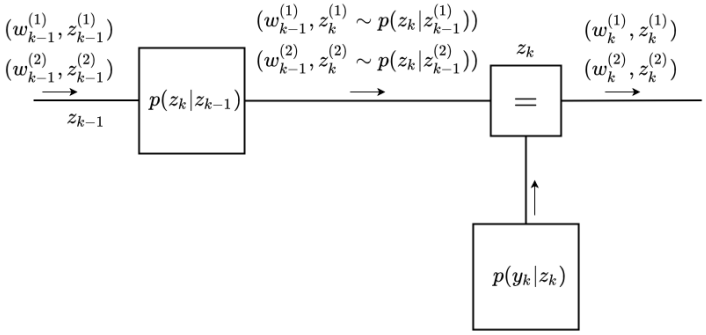


Figure A4. Bootstrap Particle Filtering employs the state transition distributions, $p(z_k|z_{k-1})$ as proposal distributions, which can easily be supported in our framework by defining BP rules at soft factors for incoming messages that are LWS. The rule is straightforward to implement: weights stay unchanged; for each incoming sample, instantiate a new conditional distribution and draw a sample from it. The weight update is automatically carried out at equality node by $\tilde{w}_k^{(i)} \propto p(y_k|z_k^{(i)}) w_{k-1}^{(i)}$, which is followed by a normalization: $w_k^{(i)} = \frac{\tilde{w}_k^{(i)}}{\sum_{j=1} \tilde{w}_k^{(j)}}$.

Having implemented the BP rule at a soft factor for incoming LWS messages, we have to show how posteriors are approximated through updating weights. Suppose that $\overrightarrow{m}_{z_k}(z_k)$ is a message that carries LWS, and $\overleftarrow{m}_{z_k}(z_k)$ is parameterized either by an EF or an NEF. Then, we define the posterior update rule as follows:

$$\text{Given } \overrightarrow{m}_{z_k}(z_k) = \left\{\left(w_{k-1}^{(1)}, z_k^{(1)}\right), \ldots, \left(w_{k-1}^{(N)}, z_k^{(N)})\right)\right\},$$
$$q(z_k) \propto \overrightarrow{m}_{z_k}(z_k)\overleftarrow{m}_{z_k}(z_k)$$
$$\approx \left\{\left(w_k^{(1)}, z_k^{(1)}\right), \ldots, \left(w_k^{(N)}, z_k^{(N)})\right)\right\}, \tag{A46a}$$
$$\text{where } w_k^{(i)} \propto w_{k-1}^{(i)}\overleftarrow{m}_{z_k}(z_k^{(i)}). \tag{A46b}$$

In Bootstrap particle filtering, these rules update the weights at equality nodes, automatically. A major drawback of sequential importance sampling methods is that the further samples commute over time steps, the more they lose their ability to recover the underlying process, and many of the weights approach to zero. This phenomenon is known as the degeneracy problem and can be alleviated by resampling [36,60]. In our automated setting, at each weight update step, we measure the effectiveness of the existing samples by $n_{\text{eff}} = \frac{1}{\sum_{i=1}^{N}\left(w^{(i)}\right)^2}$, as it is shown in [36]. Then, we resample if $n_{\text{eff}} < N/10$ [36]. A user can effortlessly execute a particle filtering procedure in our method by putting an LWS prior on the first hidden state of a sequential model and running BP inference on the model (For demonstration purposes, we implemented BP rules at Gaussian node for LWS messages. The user can implement the very same rules for other soft factors according to their needs. Visit https://github.com/biaslab/ForneyLab.jl/blob/master/demo/bootstrap_particle_filter.ipynb (accessed on 25 June 2021) for a toy example.).

Appendix F. Illustrative Example

Consider the following model visualized in Figure A5: $p(x) = \mathcal{N}(x; \mu_x, v_x)$, $p(z) = \mathcal{N}(z; \mu_z, v_z)$, $p(w|z) = \delta(w - \exp(z))$, $p(y|x, w) = \mathcal{N}(y; x, 1/w)$, with observation y. In [9], VMP messages are provided as an example for a normal node parameterized by mean and precision. Here, we augment their example by a deterministic node to illustrate how Extended VMP operates to approximate the posteriors for x, w and z:

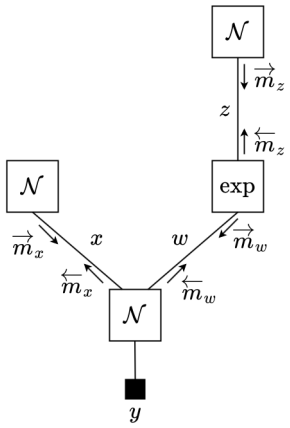


Figure A5. The model $p(x) = \mathcal{N}(x; \mu_x, v_x)$, $p(z) = \mathcal{N}(z; \mu_z, v_z)$, $p(w|z) = \delta(w - \exp(z))$, $p(y|x, w) = \mathcal{N}(y; x, 1/w)$ is visualized together with the messages. The EVMP algorithm approximates the backward VMP message towards x by estimating $\langle w \rangle$ with Monte Carlo summation. Once this VMP message is approximated, the update for $q(x)$ is available in closed form. The backward message toward w and z requires $\langle x \rangle$ and $\langle x^2 \rangle$. These expectations can be computed analytically since they are the sufficient statistics of $q(x)$. However, this time, the forward and the backward messages differ in sufficient statistics, which impedes the analytical calculations for $q(w)$ and $q(z)$. We approximate them by IS and Laplace, respectively.

- Initiate $q(x)$, $q(z)$ by Normal distributions and $q(w)$ by an LWS.
- Repeat until convergence the following three steps:

1. – Choose w for updating.
   – Calculate VMP message $\overleftarrow{m}_w(w)$ by (14). In this case,

$$\begin{aligned}\overleftarrow{m}_w(w) &\propto \exp\left(\begin{bmatrix}\log w\\ w\end{bmatrix}^{\mathsf{T}}\begin{bmatrix}0.5\\ \langle x\rangle y - 0.5(\langle x^2\rangle + y^2)\end{bmatrix}\right)\\ &\propto \mathcal{G}a\left(w; 1.5, -\langle x\rangle y + 0.5\left(\left\langle x^2\right\rangle + y^2\right)\right),\end{aligned}$$

   where $\mathcal{G}a(x;\alpha,\beta)$ is a Gamma distribution with shape α and rate β.
   – Calculate $\overrightarrow{m}_w(w)$ by the following (16):

$$\begin{aligned}&\overrightarrow{m}_w(w) = \left\{\left(\frac{1}{N}, \exp(z^{(1)})\right), \ldots, \left(\frac{1}{N}, \exp(z^{(N)})\right)\right\},\\ &\text{where } z^{(i)} \sim \overrightarrow{m}_z(z) = \mathcal{N}(z;\mu_z,v_z).\end{aligned}$$

   – Update $q(w)$ by Section 3.5 rule (3).
2. – Choose z for updating.
   – Calculate $\overleftarrow{m}_z(z)$ by (18), which is a NEF distribution:

$$\overleftarrow{m}_z(z) = \overleftarrow{m}_w(\exp(z)) \propto \exp\left(\begin{bmatrix}z\\ \exp(z)\end{bmatrix}^{\mathsf{T}}\begin{bmatrix}0.5\\ \langle x\rangle y - 0.5(\langle x^2\rangle + y^2)\end{bmatrix}\right).$$

   – The forward message is simply the prior: $\overrightarrow{m}_z(z) = \mathcal{N}(z;\mu_z,v_z)$
   – Update $q(z)$ by Section 3.5 rule (2)(a).
3. – Choose x for updating.
   – Calculate VMP message $\overleftarrow{m}_x(x)$ by (14). In this case,

$$\begin{aligned}\overleftarrow{m}_x(x) &\propto \exp\left(\begin{bmatrix}x\\ x^2\end{bmatrix}^{\mathsf{T}}\begin{bmatrix}\langle w\rangle y\\ -0.5\langle w\rangle\end{bmatrix}\right)\\ &\propto \mathcal{N}(x; y, 1/\langle w\rangle).\end{aligned}$$

   – The forward message is the prior:

$$\overrightarrow{m}_x(x) = \mathcal{N}(x;\mu_x,v_x) \propto \exp\left(\begin{bmatrix}x\\ x^2\end{bmatrix}^{\mathsf{T}}\begin{bmatrix}\mu_x/v_x\\ -0.5/v_x\end{bmatrix}\right).$$

   – Update $q(x)$ by Section 3.5 rule (1), i.e., the following:

$$\begin{aligned}q(x) &\propto \exp\left(\begin{bmatrix}x\\ x^2\end{bmatrix}^{\mathsf{T}}\begin{bmatrix}\mu_x/v_x + \langle w\rangle y\\ -0.5(1/v_x + \langle w\rangle)\end{bmatrix}\right)\\ &= \mathcal{N}\left(x; \frac{\mu_x + \langle w\rangle v_x y}{1 + \langle w\rangle v_x}, \frac{v_x}{1 + \langle w\rangle v_x}\right).\end{aligned}$$

The expectation quantities $\langle w\rangle$, $\langle x\rangle$, $\langle x^2\rangle$ that appear in the message calculations are computed according to Section 3.8. Therefore, while $\langle w\rangle$ is estimated via a Monte Carlo summation, $\langle x\rangle$ and $\langle x^2\rangle$ are available in closed form.

References

1. van de Meent, J.W.; Paige, B.; Yang, H.; Wood, F. An Introduction to Probabilistic Programming. *arXiv* **2018**, arXiv:1809.10756.
2. Carpenter, B.; Gelman, A.; Hoffman, M.D.; Lee, D.; Goodrich, B.; Betancourt, M.; Brubaker, M.; Guo, J.; Li, P.; Riddell, A. Stan: A Probabilistic Programming Language. *J. Stat. Softw.* **2017**, *76*, 1–32. [CrossRef]
3. Dillon, J.V.; Langmore, I.; Tran, D.; Brevdo, E.; Vasudevan, S.; Moore, D.; Patton, B.; Alemi, A.; Hoffman, M.; Saurous, R.A. TensorFlow Distributions. *arXiv* **2017**, arXiv:1711.10604.
4. Bingham, E.; Chen, J.P.; Jankowiak, M.; Obermeyer, F.; Pradhan, N.; Karaletsos, T.; Singh, R.; Szerlip, P.; Horsfall, P.; Goodman, N.D. Pyro: Deep Universal Probabilistic Programming. *J. Mach. Learn. Res.* **2019**, *20*, 1–6.

5. Ge, H.; Xu, K.; Ghahramani, Z. Turing: A Language for Flexible Probabilistic Inference. In *International Conference on Artificial Intelligence and Statistics*; PMLR, 2018; pp. 1682–1690.
6. Titsias, M.; Lázaro-Gredilla, M. Doubly stochastic variational Bayes for non-conjugate inference. In *International Conference on Machine Learning*; PMLR, 2014; pp. 1971–1979.
7. Minka, T.; Winn, J.; Guiver, J.; Zaykov, Y.; Fabian, D.; Bronskill, J. Infer.NET 0.3. 2018. Available online: https://dotnet.github.io/infer/ (accessed on 25 June 2021).
8. Cox, M.; van de Laar, T.; de Vries, B. A factor graph approach to automated design of Bayesian signal processing algorithms. *Int. J. Approx. Reason.* **2019**, *104*, 185–204. [CrossRef]
9. Winn, J.; Bishop, C.M. Variational message passing. *J. Mach. Learn. Res.* **2005**, *6*, 661–694.
10. Dauwels, J. On Variational Message Passing on Factor Graphs. In Proceedings of the IEEE International Symposium on Information Theory, Nice, France, 24–29 June 2007; pp. 2546–2550.
11. Tokdar, S.T.; Kass, R.E. Importance sampling: A review. *Wiley Interdiscip. Rev. Comput. Stat.* **2010**, *2*, 54–60. [CrossRef]
12. Bishop, C.M. *Pattern Recognition and Machine Learning*; Springer: Berlin/Heidelberg, Germany, 2006.
13. Baydin, A.G.; Pearlmutter, B.A.; Radul, A.A.; Siskind, J.M. Automatic differentiation in machine learning: A survey. *J. Mach. Learn. Res.* **2017**, *18*, 5595–5637.
14. Bezanson, J.; Karpinski, S.; Shah, V.B.; Edelman, A. Julia: A fast dynamic language for technical computing. *arXiv* **2012**, arXiv:1209.5145.
15. Loeliger, H.A.; Dauwels, J.; Hu, J.; Korl, S.; Ping, L.; Kschischang, F.R. The factor graph approach to model-based signal processing. *Proc. IEEE* **2007**, *95*, 1295–1322. [CrossRef]
16. Loeliger, H.A. An introduction to factor graphs. *IEEE Signal Process. Mag.* **2004**, *21*, 28–41. [CrossRef]
17. Blei, D.M.; Kucukelbir, A.; McAuliffe, J.D. Variational Inference: A Review for Statisticians. *J. Am. Stat. Assoc.* **2017**, *112*, 859–877. [CrossRef]
18. Dauwels, J.; Korl, S.; Loeliger, H.A. Particle methods as message passing. In Proceedings of the IEEE International Symposium on Information Theory, Seattle, WA, USA, 9–14 July 2006; pp. 2052–2056.
19. Şenöz, I.; De Vries, B. Online variational message passing in the hierarchical Gaussian filter. In Proceedings of the 2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP), Aalborg, Denmark, 17–20 September 2018; pp. 1–6.
20. Mathys, C.D.; Lomakina, E.I.; Daunizeau, J.; Iglesias, S.; Brodersen, K.H.; Friston, K.J.; Stephan, K.E. Uncertainty in perception and the Hierarchical Gaussian Filter. *Front. Hum. Neurosci.* **2014**, *8*, 825. [CrossRef]
21. Kucukelbir, A.; Tran, D.; Ranganath, R.; Gelman, A.; Blei, D.M. Automatic differentiation variational inference. *J. Mach. Learn. Res.* **2017**, *18*, 430–474.
22. Kalman, R.E. A new approach to linear filtering and prediction problems. *J. Basic Eng.* **1960**, *82*, 35–45. [CrossRef]
23. Barber, D. *Bayesian Reasoning and Machine Learning*; Cambridge University Press: Cambridge, UK, 2012.
24. Ghahramani, Z.; Hinton, G.E. *Parameter Estimation for Linear Dynamical Systems*; Technical Report CRG-TR-92-2; Department of Computer Science, University of Toronto: Toronto, ON, Canada, 1996.
25. Beal, M.J. Variational Algorithms for Approximate Bayesian Inference. Ph.D. Thesis, UCL (University College London), London, UK, 2003.
26. Hoffman, M.D.; Gelman, A. The No-U-Turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *J. Mach. Learn. Res.* **2014**, *15*, 1593–1623.
27. Ghahramani, Z.; Hinton, G.E. Variational learning for switching state-space models. *Neural Comput.* **2000**, *12*, 831–864. [CrossRef] [PubMed]
28. Neal, R.M. MCMC using Hamiltonian dynamics. *Handb. Markov Chain Monte Carlo* **2011**, *2*, 113–162.
29. Betancourt, M. A conceptual introduction to Hamiltonian Monte Carlo. *arXiv* **2017**, arXiv:1701.02434.
30. Wood, F.; Meent, J.W.; Mansinghka, V. A new approach to probabilistic programming inference. In *Artificial Intelligence and Statistics*; PMLR, 2014; pp. 1024–1032.
31. Andrieu, C.; Doucet, A.; Holenstein, R. Particle markov chain monte carlo methods. *J. R. Stat. Soc. Ser. B* **2010**, *72*, 269–342. [CrossRef]
32. De Freitas, N.; Højen-Sørensen, P.; Jordan, M.I.; Russell, S. Variational MCMC. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*; Morgan Kaufmann Publishers Inc.: Burlington, MA, USA, 2001; pp. 120–127.
33. Wexler, Y.; Geiger, D. Importance sampling via variational optimization. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*; AUAI Press: Arlington, VA, USA, 2007; pp. 426–433.
34. Salimans, T.; Kingma, D.; Welling, M. Markov chain monte carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning*; PMLR, 2015; pp. 1218–1226.
35. Ye, L.; Beskos, A.; De Iorio, M.; Hao, J. Monte Carlo co-ordinate ascent variational inference. In *Statistics and Computing*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 1–19.
36. Särkkä, S. *Bayesian Filtering and Smoothing*; Cambridge University Press: Cambridge, UK, 2013; Volume 3.
37. Murphy, K.P. *Machine Learning: A Probabilistic Perspective*; MIT Press: Cambridge, MA, USA, 2012.
38. Gordon, N.J.; Salmond, D.J.; Smith, A.F. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEEE Proc. Radar Signal Process.* **1993**, *140*, 107–113. [CrossRef]

39. Frank, A.; Smyth, P.; Ihler, A. Particle-based variational inference for continuous systems. *Adv. Neural Inf. Process. Syst.* **2009**, *22*, 826–834.
40. Ihler, A.; McAllester, D. Particle belief propagation. In *Artificial Intelligence and Statistics*; PMLR, 2009; pp. 256–263.
41. Wainwright, M.J.; Jaakkola, T.S.; Willsky, A.S. A new class of upper bounds on the log partition function. *IEEE Trans. Inf. Theory* **2005**, *51*, 2313–2335. [CrossRef]
42. Saeedi, A.; Kulkarni, T.D.; Mansinghka, V.K.; Gershman, S.J. Variational particle approximations. *J. Mach. Learn. Res.* **2017**, *18*, 2328–2356.
43. Raiko, T.; Valpola, H.; Harva, M.; Karhunen, J. Building Blocks for Variational Bayesian Learning of Latent Variable Models. *J. Mach. Learn. Res.* **2007**, *8*, 155–201.
44. Knowles, D.A.; Minka, T. Non-conjugate variational message passing for multinomial and binary regression. *Adv. Neural Inf. Process. Syst.* **2011**, *24*, 1701–1709.
45. Khan, M.; Lin, W. Conjugate-Computation Variational Inference: Converting Variational Inference in Non-Conjugate Models to Inferences in Conjugate Models. In *Artificial Intelligence and Statistics*; PMLR, 2017; pp. 878–887.
46. Ranganath, R.; Gerrish, S.; Blei, D. Black box variational inference. In *Artificial Intelligence and Statistics*; PMLR, 2014; pp. 814–822.
47. Mackay, D.J.C. Introduction to monte carlo methods. In *Learning in Graphical Models*; Springer: Berlin/Heidelberg, Germany, 1998; pp. 175–204.
48. Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*; Morgan Kaufmann: Burlington, MA, USA, 1988.
49. MacKay, D.J. *Information Theory, Inference and Learning Algorithms*; Cambridge University Press: Cambridge, UK, 2003.
50. Minka, T.P. Expectation Propagation for approximate Bayesian inference. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 2001; pp. 362–369.
51. Vehtari, A.; Gelman, A.; Sivula, T.; Jylänki, P.; Tran, D.; Sahai, S.; Blomstedt, P.; Cunningham, J.P.; Schiminovich, D.; Robert, C.P. Expectation Propagation as a Way of Life: A Framework for Bayesian Inference on Partitioned Data. *J. Mach. Learn. Res.* **2020**, *21*, 1–53.
52. Wainwright, M.J.; Jordan, M.I. Graphical Models, Exponential Families, and Variational Inference. *Found. Trends Mach. Learn.* **2008**, *1*, 1–305. [CrossRef]
53. Gelman, A.; Carlin, J.B.; Stern, H.S.; Dunson, D.B.; Vehtari, A.; Rubin, D.B. *Bayesian Data Analysis*; CRC Press: Boca Raton, FL, USA, 2013.
54. Koyama, S.; Castellanos Pérez-Bolde, L.; Shalizi, C.R.; Kass, R.E. Approximate methods for state-space models. *J. Am. Stat. Assoc.* **2010**, *105*, 170–180. [CrossRef]
55. Macke, J.H.; Buesing, L.; Cunningham, J.P.; Yu, B.M.; Shenoy, K.V.; Sahani, M. Empirical models of spiking in neural populations. *Adv. Neural Inf. Process. Syst.* **2011**, *24*, 1350–1358.
56. Smola, A.J.; Vishwanathan, S.; Eskin, E. Laplace propagation. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2004; pp. 441–448.
57. Acerbi, L. Variational bayesian monte carlo. *arXiv* **2018**, arXiv:1810.05558.
58. Ajgl, J.; Šimandl, M. Differential entropy estimation by particles. *IFAC Proc. Vol.* **2011**, *44*, 11991–11996. [CrossRef]
59. Revels, J.; Lubin, M.; Papamarkou, T. Forward-Mode Automatic Differentiation in Julia. *arXiv* **2016**, arXiv:1607.07892.
60. Doucet, A.; de Freitas, N.; Gordon, N. *Sequential Monte Carlo Methods in Practice*; Springer: Berlin/Heidelberg, Germany, 2001.