Probabilistic Unrolling: Scalable, Inverse-Free Maximum Likelihood Estimation for Latent Gaussian Models

Alexander Lin [1] Bahareh Tolooshams [1] Yves Atchadé [2] Demba Ba [1]

Abstract

Latent Gaussian models have a rich history in statistics and machine learning, with applications ranging from factor analysis to compressed sensing to time series analysis. The classical method for maximizing the likelihood of these models is the expectation-maximization (EM) algorithm. For problems with high-dimensional latent variables and large datasets, EM scales poorly because it needs to invert as many large covariance matrices as the number of data points. We introduce *probabilistic unrolling*, a method that combines Monte Carlo sampling with iterative linear solvers to circumvent matrix inversion. Our theoretical analyses reveal that unrolling and backpropagation through the iterations of the solver can accelerate gradient estimation for maximum likelihood estimation. In experiments on simulated and real data, we demonstrate that probabilistic unrolling learns latent Gaussian models up to an order of magnitude faster than gradient EM, with minimal losses in model performance.

1. Introduction

Latent variable models with Gaussian prior and Gaussian likelihood, i.e. *latent Gaussian models* (LGMs), are popular and powerful tools within statistics and machine learning. They have found applications in many settings, such as factor analysis (Basilevsky, 2009), sparse Bayesian learning (Tipping, 2001), state-space models (Durbin & Koopman, 2012), and neural linear models (Ober & Rasmussen, 2019). In these models, the means and/or covariances of the Gaussian distributions are functions of parameters that must be optimized to fit observed data.

The expectation-maximization (EM) algorithm (Dempster et al., 1977) is a popular way to optimize the parameters by maximum likelihood estimation. One variant called *gradient EM* (Lange, 1995) implements the M-step through a single iteration of gradient descent. For problems with high-dimensional latent variables and many training examples, gradient EM scales poorly due to the need to invert as many large covariance matrices as the number of examples.

Advances in numerical linear algebra have demonstrated, in various contexts, that iterative solvers often provide a much faster alternative to matrix inversion (Saad, 2003; Ubaru et al., 2017; Gardner et al., 2018; Lin et al., 2022b). A separate, burgeoning literature on unrolled optimization has shown theoretical and practical benefits to differentiating through the iterations of deterministic optimizers (Maclaurin et al., 2015; Shaban et al., 2019; Ablin et al., 2020; Tolooshams & Ba, 2022; Malézieux et al., 2021). This literature begs questions as to the potential benefits, in a latent variable setting, of unrolling the iterations of a sampler (i.e. stochastic solver), and differentiating through them.

Contributions We introduce *probabilistic unrolling*, a computational framework that accelerates maximum likelihood estimation for large-scale, high-dimensional LGMs. Our method provides a way to run gradient EM without matrix inversions. Specifically, we design iterative linear solvers to yield the probabilistic quantities needed by the EM algorithm (i.e. posterior means and covariance samples). Our method reduces the complexity of gradient EM from a cubic function of the latent dimension to a quadratic function in the general case, and a linear function in special cases.

We theoretically analyze the faithfulness of probabilistic unrolling to gradient EM when encountering two sources of error: (a) the *statistical error* from using a finite number of covariance samples, and (b) the *optimization error* from stopping the solver before convergence. We provide bounds for both of these factors, producing insights on how to pick the number of samples and the number of solver iterations. Finally, we show that our method can further improve its approximation to the true EM gradient by backpropagating through the unrolled iterations of the solver.

Probabilistic unrolling can be viewed as training a recurrent network in which each layer applies a matrix operation

[1] School of Engineering and Applied Sciences, Harvard University, Boston, MA, USA [2] Department of Mathematics and Statistics, Boston University, Boston, MA, USA. Correspondence to: Alexander Lin <alin@seas.harvard.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

from the unrolled linear solver. We implement this highly structured architecture in modern deep learning frameworks to further benefit from GPU acceleration. We perform several experiments with simulated and real data, showing that probabilistic unrolling can fit LGMs of practical interest up to 70 times faster than gradient EM. Our code is available at https://github.com/al5250/prob-unroll.

2. Background: Latent Gaussian Model

Let $\{\boldsymbol{y}^{(n)}\}_{n=1}^{N}$ denote N i.i.d. observations, each associated with a latent variable $\boldsymbol{z}^{(n)}$. In a LGM, the *prior* on each latent variable and *likelihood* (i.e. conditional distribution) of each observation both follow Gaussian distributions,

$$\begin{aligned}\boldsymbol{z}^{(n)}|\boldsymbol{\theta} &\sim \mathcal{N}(\boldsymbol{\nu}_{\boldsymbol{\theta}}, \boldsymbol{\Gamma}_{\boldsymbol{\theta}}^{-1}), \\ \boldsymbol{y}^{(n)}|\boldsymbol{z}^{(n)}, \boldsymbol{\theta} &\sim \mathcal{N}(\boldsymbol{\Phi}_{\boldsymbol{\theta}}\boldsymbol{z}^{(n)} + \boldsymbol{\eta}_{\boldsymbol{\theta}}, \boldsymbol{\Psi}_{\boldsymbol{\theta}}^{-1}), \quad n = 1, \ldots, N.\end{aligned} \tag{1}$$

The prior and likelihood depend on a set of *canonical parameters* ($\boldsymbol{\nu}_{\boldsymbol{\theta}} \in \mathbb{R}^{D}, \boldsymbol{\Gamma}_{\boldsymbol{\theta}} \in \mathbb{R}^{D\times D}, \boldsymbol{\Phi}_{\boldsymbol{\theta}} \in \mathbb{R}^{M\times D}, \boldsymbol{\eta}_{\boldsymbol{\theta}} \in \mathbb{R}^{M}$, and a diagonal matrix $\boldsymbol{\Psi}_{\boldsymbol{\theta}} \in \mathbb{R}^{M\times M}$) that form the means and covariances of the Gaussian distributions. The canonical parameters are themselves functions of the model's *free parameters* $\boldsymbol{\theta}$, which are individual values that can be learned through maximum likelihood estimation.

Examples The LGM (1) generalizes many models within statistics and machine learning. Some famous examples include (a) *factor analysis*, a probabilistic generalization of PCA (Basilevsky, 2009), (b) *sparse Bayesian learning*, a Bayesian approach to compressed sensing (Wipf & Rao, 2004), and (c) *state-space models*, one of the most popular class of probabilistic time series models (Durbin & Koopman, 2012). With the advent of deep learning, the LGM class has broadened to include complex, non-linear structures such as (d) *neural linear models*, i.e. neural networks whose trainable weights correspond to free parameters (Ober & Rasmussen, 2019). For each of these models (and others), we work out the definition of free parameters $\boldsymbol{\theta}$ and how they map to the canonical parameters in Appendix A.

Missing Data In many applications of LGMs, $\boldsymbol{y}^{(n)}$ may have missing values, i.e. we may not observe all its entries. To account for missing data, we assume that for each n, we observe $\tilde{\boldsymbol{y}}^{(n)} = \boldsymbol{\Omega}^{(n)}\boldsymbol{y}^{(n)}$, where the mask $\boldsymbol{\Omega}^{(n)} \in \mathbb{R}^{M_n\times M}$ is a row-wise subset of the $M \times M$ identity matrix.

EM Inference To fit the parameters $\boldsymbol{\theta} \in \Theta$ to data $\tilde{\boldsymbol{y}}^{(1)}, \ldots, \tilde{\boldsymbol{y}}^{(N)}$, we perform maximum likelihood estimation or, equivalently, minimize the negative log-likelihood,

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) :=& \frac{1}{N}\sum_{n=1}^{N} -\log p(\tilde{\boldsymbol{y}}^{(n)}|\boldsymbol{\theta}) \\ =& \frac{1}{N}\sum_{n=1}^{N} -\log \int p(\tilde{\boldsymbol{y}}^{(n)}|\boldsymbol{z}^{(n)}, \boldsymbol{\theta})p(\boldsymbol{z}^{(n)}|\boldsymbol{\theta})d\boldsymbol{z}^{(n)}.\end{aligned} \tag{2}$$

Due to the latent variable $\boldsymbol{z}^{(n)}$, one common approach to minimizing (2) is to use the *expectation-maximization* (EM) algorithm (Dempster et al., 1977). EM revolves around the $\mathcal{Q}$-function, which is defined for any $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2\} \in \Theta \times \Theta$ as

$$\begin{aligned}\mathcal{Q}(\boldsymbol{\theta}_1|\boldsymbol{\theta}_2) &:= \frac{1}{N}\sum_{n=1}^{N} q^{(n)}(\boldsymbol{\theta}_1|\boldsymbol{\theta}_2), \\ q^{(n)}(\boldsymbol{\theta}_1|\boldsymbol{\theta}_2) &:= \mathbb{E}_{p(\boldsymbol{z}^{(n)}|\tilde{\boldsymbol{y}}^{(n)}, \boldsymbol{\theta}_2)}[-\log p(\boldsymbol{z}^{(n)}, \tilde{\boldsymbol{y}}^{(n)}|\boldsymbol{\theta}_1)].\end{aligned} \tag{3}$$

The $\mathcal{Q}$-function is called the *expected complete-data negative log-likelihood* because it averages the negative log-likelihood of the observed data $\boldsymbol{y}^{(n)}$ and the unobserved data $\boldsymbol{z}^{(n)}$ over all possible realizations of $\boldsymbol{z}^{(n)}$ (Bishop & Nasrabadi, 2006, Ch. 9). EM iterations repeatedly alternate between constructing $\mathcal{Q}$ and minimizing it to make progress on $\mathcal{L}$: Given a current solution $\boldsymbol{\theta}^{\text{old}}$, the *E*-step computes the posterior distribution $p(\boldsymbol{z}^{(n)}|\tilde{\boldsymbol{y}}^{(n)}, \boldsymbol{\theta}^{\text{old}})$ to form the function $\mathcal{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{\text{old}})$, defined for all $\boldsymbol{\theta} \in \Theta$. The *M*-step then finds a new solution $\boldsymbol{\theta}^{\text{new}}$ such that $\mathcal{Q}(\boldsymbol{\theta}^{\text{new}}|\boldsymbol{\theta}^{\text{old}}) \leq \mathcal{Q}(\boldsymbol{\theta}^{\text{old}}|\boldsymbol{\theta}^{\text{old}})$. This guarantees that $\mathcal{L}(\boldsymbol{\theta}^{\text{new}}) \leq \mathcal{L}(\boldsymbol{\theta}^{\text{old}})$.

Variants of EM differ in how they implement the *M*-step. *Classical EM* (Dempster et al., 1977) solves an optimization problem, i.e. $\boldsymbol{\theta}^{\text{new}} := \arg\min_{\boldsymbol{\theta}\in\Theta} \mathcal{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{\text{old}})$. We focus on a computationally-simpler alternative called *gradient EM* (Lange, 1995; Balakrishnan et al., 2017),

$$\boldsymbol{\theta}^{\text{new}} := \boldsymbol{\theta}^{\text{old}} - \alpha \cdot \nabla_1\mathcal{Q}(\boldsymbol{\theta}^{\text{old}}|\boldsymbol{\theta}^{\text{old}}), \tag{4}$$

where $\alpha \in \mathbb{R}$ is the step size and $\nabla_1\mathcal{Q}$ means the gradient with respect to the first argument of $\mathcal{Q}$, as defined in (3).

EM for the LGM For latent Gaussian models, $\mathcal{Q}$ and its gradient are computable in closed-form. Each $q^{(n)}$ in (3) simplifies to (dropping the index n for convenience):

$$\begin{aligned}q(\boldsymbol{\theta}_1|\boldsymbol{\theta}_2) = \frac{1}{2}\boldsymbol{\mu}_{\boldsymbol{\theta}_2}^{\top}\mathbf{A}_{\boldsymbol{\theta}_1}\boldsymbol{\mu}_{\boldsymbol{\theta}_2} - \boldsymbol{b}_{\boldsymbol{\theta}_1}^{\top}\boldsymbol{\mu}_{\boldsymbol{\theta}_2} \\ + \frac{1}{2}\text{Tr}(\mathbf{A}_{\boldsymbol{\theta}_1}\boldsymbol{\Sigma}_{\boldsymbol{\theta}_2}) + c_{\boldsymbol{\theta}_1},\end{aligned} \tag{5}$$

where, for all $\boldsymbol{\theta} \in \Theta$, we define the quantities

$$\begin{aligned}\mathbf{A}_{\boldsymbol{\theta}} &:= \boldsymbol{\Gamma}_{\boldsymbol{\theta}} + \boldsymbol{\Phi}_{\boldsymbol{\theta}}^{\top}\boldsymbol{\Omega}^{\top}\boldsymbol{\Omega}\boldsymbol{\Psi}_{\boldsymbol{\theta}}\boldsymbol{\Omega}^{\top}\boldsymbol{\Omega}\boldsymbol{\Phi}_{\boldsymbol{\theta}}, \\ \boldsymbol{b}_{\boldsymbol{\theta}} &:= \boldsymbol{\Gamma}_{\boldsymbol{\theta}}\boldsymbol{\nu}_{\boldsymbol{\theta}} + \boldsymbol{\Phi}_{\boldsymbol{\theta}}^{\top}\boldsymbol{\Omega}^{\top}\boldsymbol{\Omega}\boldsymbol{\Psi}_{\boldsymbol{\theta}}\boldsymbol{\Omega}^{\top}(\tilde{\boldsymbol{y}} - \boldsymbol{\Omega}\boldsymbol{\eta}_{\boldsymbol{\theta}}), \\ c_{\boldsymbol{\theta}} &:= \tfrac{1}{2}(\tilde{\boldsymbol{y}} - \boldsymbol{\Omega}\boldsymbol{\eta}_{\boldsymbol{\theta}})^{\top}\boldsymbol{\Omega}\boldsymbol{\Psi}_{\boldsymbol{\theta}}\boldsymbol{\Omega}^{\top}(\tilde{\boldsymbol{y}} - \boldsymbol{\Omega}\boldsymbol{\eta}_{\boldsymbol{\theta}}) + \tfrac{1}{2}\boldsymbol{\nu}_{\boldsymbol{\theta}}^{\top}\boldsymbol{\Gamma}_{\boldsymbol{\theta}}\boldsymbol{\nu}_{\boldsymbol{\theta}} \\ &\quad - \tfrac{1}{2}\log\det\boldsymbol{\Omega}\boldsymbol{\Psi}_{\boldsymbol{\theta}}\boldsymbol{\Omega}^{\top} - \tfrac{1}{2}\log\det\boldsymbol{\Gamma}_{\boldsymbol{\theta}},\end{aligned} \tag{6}$$

and the posterior $p(\boldsymbol{z}|\tilde{\boldsymbol{y}}, \boldsymbol{\theta}) \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}})$ is given by

$$\boldsymbol{\mu}_{\boldsymbol{\theta}} := \boldsymbol{\Sigma}_{\boldsymbol{\theta}}\boldsymbol{b}_{\boldsymbol{\theta}}, \qquad \boldsymbol{\Sigma}_{\boldsymbol{\theta}} := \mathbf{A}_{\boldsymbol{\theta}}^{-1}. \tag{7}$$

The derivation for equations (5)-(7) is given in Appendix B.

Computational Challenges Gradient EM involves computing the gradient of (5) (which we call the *exact gradient*),

$$\boldsymbol{g}^{\star}(\boldsymbol{\theta}) := \nabla_1 q(\boldsymbol{\theta}|\boldsymbol{\theta}). \tag{8}$$

Since $\boldsymbol{g}^\star(\boldsymbol{\theta})$ depends on the posterior moments $(\boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}})$, it requires inverting a large matrix of size $D \times D$. This has time cost $\mathcal{O}(D^3)$ and storage cost $\mathcal{O}(D^2)$, which becomes prohibitive for large D. Furthermore, for N different data vectors, we need to compute N posterior moments $(\boldsymbol{\mu}_{\boldsymbol{\theta}}^{(n)}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{(n)})$, which requires N separate matrix inversions.

We now arrive at the main goal of the paper: In the ensuing sections, we introduce a computational framework called *probabilistic unrolling* that can provably accelerate gradient EM by avoiding explicit matrix inversions. This allows us to fit latent Gaussian models at substantially greater scale in high dimensions D and for large dataset sizes N.

3. Method: Probabilistic Unrolling

Probabilistic unrolling circumvents matrix inversion by iteratively solving multiple linear systems in parallel. We design the systems to perform posterior inference, i.e. the solutions are the posterior mean $\boldsymbol{\mu}_{\boldsymbol{\theta}}$ and covariance samples distributed as $\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}})$. We use these quantities to estimate the EM objective (5) and its gradient $\boldsymbol{g}^\star(\boldsymbol{\theta})$ (8). This process requires less time and memory than computing $\boldsymbol{g}^\star(\boldsymbol{\theta})$ directly. We also show that backpropagating through the linear solvers further improves our estimation of $\boldsymbol{g}^\star(\boldsymbol{\theta})$.

From a deep learning perspective, the overall method looks like a recurrent network (Fig. 1). We can view the unrolled sequence of solver iterations as a *recurrent encoder*, with weights $\boldsymbol{\theta}$, that takes the observed data $\tilde{\boldsymbol{y}}$ and refines *hidden states* representing the distribution $p(\boldsymbol{z}|\tilde{\boldsymbol{y}}, \boldsymbol{\theta})$. The hidden states are then passed through an *output layer*, also parameterized by $\boldsymbol{\theta}$, to evaluate the loss (5). Training this network is equivalent to running gradient EM for the LGM.

3.1. Monte Carlo Gradient EM

In high-dimensional settings, inverting a matrix to compute $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}$ is the main bottleneck of (5). The first step of our method replaces the trace term containing $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}$ with an unbiased estimator. Given any square matrix $\mathbf{A}$ and a sample $\boldsymbol{\sigma}_{\boldsymbol{\theta}} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}})$, it follows that $\mathbb{E}[\boldsymbol{\sigma}_{\boldsymbol{\theta}}^\top \mathbf{A} \boldsymbol{\sigma}_{\boldsymbol{\theta}}] = \mathrm{Tr}(\mathbf{A}\boldsymbol{\Sigma}_{\boldsymbol{\theta}})$ (Skilling, 1989; Hutchinson, 1989). Using $K > 1$ independent samples $\boldsymbol{\sigma}_{1,\boldsymbol{\theta}}, \ldots, \boldsymbol{\sigma}_{K,\boldsymbol{\theta}} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}})$ (to reduce variance) leads to the following approximation of (5),

$$q^\#(\boldsymbol{\theta}_1|\boldsymbol{\theta}_2) := \frac{1}{2}\boldsymbol{\mu}_{\boldsymbol{\theta}_2}^\top \mathbf{A}_{\boldsymbol{\theta}_1} \boldsymbol{\mu}_{\boldsymbol{\theta}_2} - \boldsymbol{b}_{\boldsymbol{\theta}_1}^\top \boldsymbol{\mu}_{\boldsymbol{\theta}_2} + \frac{1}{2K}\sum_{k=1}^{K} \boldsymbol{\sigma}_{k,\boldsymbol{\theta}_2}^\top \mathbf{A}_{\boldsymbol{\theta}_1} \boldsymbol{\sigma}_{k,\boldsymbol{\theta}_2} + c_{\boldsymbol{\theta}_1}. \tag{9}$$

Eq. (9) satisfies $\mathbb{E}[q^\#(\boldsymbol{\theta}_1|\boldsymbol{\theta}_2)] = q(\boldsymbol{\theta}_1|\boldsymbol{\theta}_2)$, where the expectation is taken with respect to $\boldsymbol{\sigma}_{1,\boldsymbol{\theta}}, \ldots, \boldsymbol{\sigma}_{K,\boldsymbol{\theta}}$. We now define the *Monte Carlo gradient*

$$\boldsymbol{g}^\#(\boldsymbol{\theta}) := \nabla_1 q^\#(\boldsymbol{\theta}|\boldsymbol{\theta}), \tag{10}$$

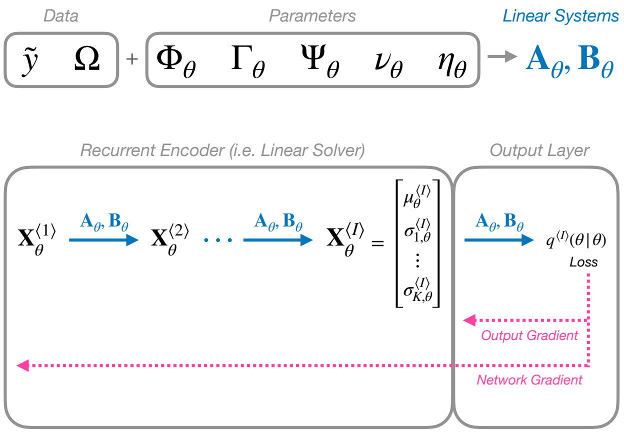
Data
Parameters
Linear Systems
$\tilde{y}$ Ω
+
Φ_θ Γ_θ Ψ_θ ν_θ η_θ
$\mathbf{A}_\theta, \mathbf{B}_\theta$
Recurrent Encoder (i.e. Linear Solver)
Output Layer
$\mathbf{X}_\theta^{\langle 1 \rangle}$
$\mathbf{A}_\theta, \mathbf{B}_\theta$
$\mathbf{X}_\theta^{\langle 2 \rangle}$
$\cdots$
$\mathbf{A}_\theta, \mathbf{B}_\theta$
$\mathbf{X}_\theta^{\langle I \rangle} = [\mu_\theta^{\langle I \rangle}, \sigma_{1,\theta}^{\langle I \rangle}, \vdots, \sigma_{K,\theta}^{\langle I \rangle}]$
$\mathbf{A}_\theta, \mathbf{B}_\theta$
$q^{\langle I \rangle}(\theta|\theta)$
Loss
Output Gradient
Network Gradient

Figure 1. The probabilistic unrolling architecture: The data point $\tilde{\boldsymbol{y}}$, mask $\boldsymbol{\Omega}$, and parameters $\boldsymbol{\theta}$ define the linear operator $\mathbf{A}_{\boldsymbol{\theta}}$ and construct the matrix $\mathbf{B}_{\boldsymbol{\theta}}$. A linear solver, unrolled for I steps, solves the matrix equation $\mathbf{A}_{\boldsymbol{\theta}}\mathbf{X}_{\boldsymbol{\theta}} = \mathbf{B}_{\boldsymbol{\theta}}$, yielding the posterior mean $\boldsymbol{\mu}_{\boldsymbol{\theta}}$ and samples $\{\boldsymbol{\sigma}_{1,\boldsymbol{\theta}}\}_{k=1}^K$ with covariance $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}$. These posterior quantities are used to compute either the output gradient (16) or network gradient (17) to approximate the true EM gradient.

which can take the place of $\boldsymbol{g}^\star(\boldsymbol{\theta})$ for updating $\boldsymbol{\theta}$ in gradient EM. The estimator satisfies $\mathbb{E}[\boldsymbol{g}^\#(\boldsymbol{\theta})] = \boldsymbol{g}^\star(\boldsymbol{\theta})$.

Constructing Samples The question remains as to how we draw each sample $\boldsymbol{\sigma}_{k,\boldsymbol{\theta}}$. Consider independent random vectors $\boldsymbol{\xi}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Gamma}_{\boldsymbol{\theta}})$ and $\boldsymbol{\zeta}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Psi}_{\boldsymbol{\theta}})$, and let

$$\boldsymbol{\delta}_k := \boldsymbol{\xi}_k + \boldsymbol{\Phi}_{\boldsymbol{\theta}}^\top \boldsymbol{\Omega}^\top \boldsymbol{\Omega} \boldsymbol{\zeta}_k. \tag{11}$$

It follows from properties of Gaussian random vectors that $\boldsymbol{\delta}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{A}_{\boldsymbol{\theta}})$, where $\mathbf{A}_{\boldsymbol{\theta}}$ is defined in (6). Then, we let

$$\boldsymbol{\sigma}_{k,\boldsymbol{\theta}} := \boldsymbol{\Sigma}_{\boldsymbol{\theta}} \boldsymbol{\delta}_k, \quad k = 1, \ldots, K. \tag{12}$$

As a result, $\boldsymbol{\sigma}_{k,\boldsymbol{\theta}}$ has covariance $\boldsymbol{\Sigma}_{\boldsymbol{\theta}} \mathbf{A}_{\boldsymbol{\theta}} \boldsymbol{\Sigma}_{\boldsymbol{\theta}} = \boldsymbol{\Sigma}_{\boldsymbol{\theta}}$.

3.2. Linear Systems and Iterative Solvers

Although the large covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}$ is no longer explicitly written in the new objective (9), it still appears in the definitions for $\boldsymbol{\mu}_{\boldsymbol{\theta}}$ and $\boldsymbol{\sigma}_{k,\boldsymbol{\theta}}$ in (7) and (12), respectively. In this section, we show how to obtain $\boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\sigma}_{k,\boldsymbol{\theta}}$ *without* explicitly forming the covariance matrix.

First, we cast $\boldsymbol{\mu}_{\boldsymbol{\theta}}$ and $\boldsymbol{\sigma}_{k,\boldsymbol{\theta}}$ as the solutions to linear systems,

$$\mathbf{A}_{\boldsymbol{\theta}} \boldsymbol{\mu}_{\boldsymbol{\theta}} = \boldsymbol{b}_{\boldsymbol{\theta}}, \qquad \mathbf{A}_{\boldsymbol{\theta}} \boldsymbol{\sigma}_{k,\boldsymbol{\theta}} = \boldsymbol{\delta}_k, \qquad k = 1, \ldots, K, \tag{13}$$

where $\mathbf{A}_{\boldsymbol{\theta}} = \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1}$ is defined in (6). Then, we solve (13) using an *iterative linear solver* (Saad, 2003). For a system $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$, iterative solvers refine a solution $\boldsymbol{x}^{\langle i \rangle}$ over iterations $i = 1, \ldots, I$ until $\boldsymbol{x}^{\langle I \rangle} \approx \mathbf{A}^{-1}\boldsymbol{b}$. At iteration i,

$$\boldsymbol{x}^{\langle i+1 \rangle} := \boldsymbol{x}^{\langle i \rangle} + \boldsymbol{p}^{\langle i \rangle}, \tag{14}$$

where $\boldsymbol{p}^{\langle i \rangle}$ is the search direction. Different solvers vary in how they construct $\boldsymbol{p}^{\langle i \rangle}$. Examples of popular solvers include *gradient descent*, *steepest descent*, and *conjugate gradient* (Saad, 2003), which we review in Appendix D.

3.3. Gradients from Truncated Linear Solvers

High-dimensional latent spaces D may require a large number of iterations I (hence a high computational cost) to obtain exact solutions. Thus, in practice, it is desirable to run the solver for small I, which leads to approximations $(\boldsymbol{\mu}_{\boldsymbol{\theta}}^{\langle I \rangle}, \boldsymbol{\sigma}_{k,\boldsymbol{\theta}}^{\langle I \rangle})$ of the true quantities $(\boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\sigma}_{k,\boldsymbol{\theta}})$. This section proposes two ways to obtain an approximate EM gradient from these partial solutions $(\boldsymbol{\mu}_{\boldsymbol{\theta}}^{\langle I \rangle}, \boldsymbol{\sigma}_{k,\boldsymbol{\theta}}^{\langle I \rangle})$. We defer a theoretical analysis of the gradient error to Section 5.

First, we substitute $(\boldsymbol{\mu}_{\boldsymbol{\theta}}^{\langle I \rangle}, \boldsymbol{\sigma}_{k,\boldsymbol{\theta}}^{\langle I \rangle})$ for $(\boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\sigma}_{k,\boldsymbol{\theta}})$ in (9), i.e.

$$\begin{aligned} q^{\langle I \rangle}(\boldsymbol{\theta}_1|\boldsymbol{\theta}_2) :=& \frac{1}{2}(\boldsymbol{\mu}_{\boldsymbol{\theta}_2}^{\langle I \rangle})^\top \mathbf{A}_{\boldsymbol{\theta}_1} \boldsymbol{\mu}_{\boldsymbol{\theta}_2}^{\langle I \rangle} - \boldsymbol{b}_{\boldsymbol{\theta}_1}^\top \boldsymbol{\mu}_{\boldsymbol{\theta}_2}^{\langle I \rangle} \\ &+ \frac{1}{2K}\sum_{k=1}^{K} (\boldsymbol{\sigma}_{k,\boldsymbol{\theta}_2}^{\langle I \rangle})^\top \mathbf{A}_{\boldsymbol{\theta}_1} \boldsymbol{\sigma}_{k,\boldsymbol{\theta}_2}^{\langle I \rangle} + c_{\boldsymbol{\theta}_1}, \end{aligned} \tag{15}$$

which satisfies $\lim_{I\to\infty} q^{\langle I \rangle}(\boldsymbol{\theta}_1|\boldsymbol{\theta}_2) = q^{\#}(\boldsymbol{\theta}_1|\boldsymbol{\theta}_2)$.

Option 1: Output Gradient We can take the gradient of (15) in a manner similar to (10) to obtain the *output gradient*

$$\widehat{\boldsymbol{g}}^{\langle I \rangle}(\boldsymbol{\theta}) := \nabla_1 q^{\langle I \rangle}(\boldsymbol{\theta}|\boldsymbol{\theta}), \tag{16}$$

which satisifies $\lim_{I\to\infty} \widehat{\boldsymbol{g}}^{\langle I \rangle}(\boldsymbol{\theta}) = \boldsymbol{g}^{\#}(\boldsymbol{\theta})$. We interpret this gradient as backpropagating through only the output layer of the architecture in Fig. 1, hence the terminology.

Option 2: Network Gradient Since the inputs to the output layer $(\boldsymbol{\mu}_{\boldsymbol{\theta}}^{\langle I \rangle}, \boldsymbol{\sigma}_{k,\boldsymbol{\theta}}^{\langle I \rangle})$ are themselves functions of the parameters $\boldsymbol{\theta}$, a natural question arises as to the benefits of additionally propagating the gradient through these quantities (and the linear solver). This leads to the *network gradient*

$$\widetilde{\boldsymbol{g}}^{\langle I \rangle}(\boldsymbol{\theta}) := \frac{\partial}{\partial \boldsymbol{\theta}}\left[q^{\langle I \rangle}(\boldsymbol{\theta}|\boldsymbol{\theta}) - \frac{1}{K}\sum_{k=1}^{K} \boldsymbol{\delta}_k^\top \boldsymbol{\sigma}_{k,\boldsymbol{\theta}}^{\langle I \rangle} \right], \tag{17}$$

which backpropagates through the whole architecture in Fig. 1. There are two changes that (17) makes to (16): (a) the use of $\frac{\partial}{\partial \boldsymbol{\theta}}$ instead of ∇_1 means that (17) differentiates with respect to both variables in (15) (not just the first argument); (b) (17) has an extra term with $\boldsymbol{\delta}_k$, which is absent from (16) but is necessary in (17) to ensure $\lim_{I\to\infty} \widetilde{\boldsymbol{g}}^{\langle I \rangle}(\boldsymbol{\theta}) = \boldsymbol{g}^{\#}(\boldsymbol{\theta})$ (short proof in Appendix C; longer proof in Appendix E.3). In Section 5.2, we will show that compared to the output gradient $\widehat{\boldsymbol{g}}^{\langle I \rangle}$, the network gradient $\widetilde{\boldsymbol{g}}^{\langle I \rangle}$ exhibits a "super-efficiency" phenomenon (Ablin et al., 2020; Tolooshams & Ba, 2022), which means that it converges faster to $\boldsymbol{g}^{\#}$.

3.4. Full Algorithm

The probabilistic unrolling algorithm is given in Algorithm 1. The LINEARSOLVER step depends on the particular choice of solver; options include gradient descent, steepest descent, and conjugate gradient. In addition to circumventing matrix inversion, probabilistic unrolling provides several computational benefits over EM, which we explain below.

Covariance-Free Computation. The iterative solvers eliminate the need to explicitly form the $D \times D$ covariance matrix $\mathbf{\Sigma}_{\boldsymbol{\theta}}$ (or even its inverse $\mathbf{A}_{\boldsymbol{\theta}}$). At each iteration i, a linear solver simply needs to compute matrix-vector products of the form $\mathbf{A}_{\boldsymbol{\theta}}\boldsymbol{v}$, for any $\boldsymbol{v} \in \mathbb{R}^D$, efficiently. For an LGM, the matrix $\mathbf{A}_{\boldsymbol{\theta}}$ is a highly-structured function of its canonical parameters and the data mask $\mathbf{\Omega}$, as shown in (6).

Exploiting LGM Structure. In many cases, the canonical parameters of the LGM exhibit additional structure, such as diagonal, Toeplitz, low rank, and sparse structure, to name a few examples. This can significantly reduce the computational and storage costs of each iteration of the linear solver. For example, in applications of sparse Bayesian learning (Lin et al., 2022b), $\mathbf{\Phi}_{\boldsymbol{\theta}}$ and its transpose often arise as Fourier-like operators. Efficient algorithms, in both computation and storage, exist for applying such operators to vectors. For a single linear system, the time cost of the solver is $\mathcal{O}(I\tau_{\boldsymbol{\theta}})$, where I is the number of iterations and $\tau_{\boldsymbol{\theta}}$ is the time needed to compute the matrix-vector multiplication $\mathbf{A}_{\boldsymbol{\theta}}\boldsymbol{v}$. The space cost is $\mathcal{O}(D + \omega_{\boldsymbol{\theta}})$, where $\omega_{\boldsymbol{\theta}}$ is the space needed to store the canonical parameters.

Amenability to Parallelization. Iterative solvers are simple and straightforward to parallelize for solving multiple linear systems (e.g. in (13)) through $\mathbf{A}_{\boldsymbol{\theta}}\mathbf{X}_{\boldsymbol{\theta}} = \mathbf{B}_{\boldsymbol{\theta}}$, where

$$\mathbf{X}_{\boldsymbol{\theta}} := [\boldsymbol{\mu}_{\boldsymbol{\theta}} | \boldsymbol{\sigma}_{1,\boldsymbol{\theta}} | \cdots | \boldsymbol{\sigma}_{K,\boldsymbol{\theta}}], \quad \mathbf{B}_{\boldsymbol{\theta}} := [\boldsymbol{b}_{\boldsymbol{\theta}} | \boldsymbol{\delta}_1 | \cdots | \boldsymbol{\delta}_K].$$

For example, Gardner et al. (2018) and Lin et al. (2022b) show how to parallelize the preconditioned conjugate gradient algorithm to solve for $\mathbf{X}_{\boldsymbol{\theta}}$. They demonstrate that matrix-based parallelization is especially suitable for multi-core hardware, such as graphics processing units. In this work, we go a step further by parallelizing the solver across data points $\{\widetilde{\boldsymbol{y}}^{(n)}\}_{n=1}^N$ to obtain solutions $\{\mathbf{X}_{\boldsymbol{\theta}}^{(n)}\}_{n=1}^N$ for every n. By (6), the operators $\{\mathbf{A}_{\boldsymbol{\theta}}^{(n)}\}_{n=1}^N$ only differ in the masks $\{\mathbf{\Omega}^{(n)}\}_{n=1}^N$. Thus, the total storage needed for performing NK matrix-vector multiplications with $\{\mathbf{A}_{\boldsymbol{\theta}}^{(n)}\}_{n=1}^N$ is only $\mathcal{O}(NKD + \omega_{\boldsymbol{\theta}})$ (where $\omega_{\boldsymbol{\theta}}$ is at most $\mathcal{O}(D^2)$) even though the matrices $\{\mathbf{A}_{\boldsymbol{\theta}}^{(n)}\}_{n=1}^N$ have $\mathcal{O}(ND^2)$ entries.

We compare the computational complexities of gradient EM using matrix inversion and probabilistic unrolling in Table 1. The additional factor of I in the space complexity of the network gradient comes from the need to store all I intermediate states of the solver for backpropagation.

Algorithm 1 PROBABILISTICUNROLLING

```
Inputs: parameters $\boldsymbol{\theta}$, dataset $\{\tilde{\boldsymbol{y}}^{(1)},\dots,\tilde{\boldsymbol{y}}^{(N)}\}$, masks
  $\{\boldsymbol{\Omega}^{(1)},\dots,\boldsymbol{\Omega}^{(N)}\}$, unrolling iterations $I$, samples $K$
repeat for number of EM iterations
    for $n = 1, 2, \dots, N$ do
        Define $\mathbf{A}_{\boldsymbol{\theta}}^{(n)}$ and compute $\boldsymbol{b}_{\boldsymbol{\theta}}^{(n)}$ by (6).
        Draw $\boldsymbol{\delta}_1^{(n)},\dots,\boldsymbol{\delta}_K^{(n)}$ using the scheme in (11).
        Define $\mathbf{B}_{\boldsymbol{\theta}}^{(n)} \leftarrow [\boldsymbol{b}_{\boldsymbol{\theta}}^{(n)}|\boldsymbol{\delta}_1^{(n)}|\dots|\boldsymbol{\delta}_K^{(n)}]$.
        $\mathbf{X}_{\boldsymbol{\theta}}^{(n)} \leftarrow \text{LINEARSOLVER}(\mathbf{A}_{\boldsymbol{\theta}}^{(n)}, \mathbf{B}_{\boldsymbol{\theta}}^{(n)}, I)$.
        Let $[\boldsymbol{\mu}_{\boldsymbol{\theta}}^{\langle I\rangle,(n)}|\boldsymbol{\sigma}_{1,\boldsymbol{\theta}}^{\langle I\rangle,(n)}|\dots|\boldsymbol{\sigma}_{K,\boldsymbol{\theta}}^{\langle I\rangle,(n)}] \leftarrow \mathbf{X}_{\boldsymbol{\theta}}^{(n)}$.
        if use output gradient then
            Compute gradient $\widehat{\boldsymbol{g}}^{\langle I\rangle,(n)}$ using (16).
        else if use network gradient then
            Compute gradient $\widetilde{\boldsymbol{g}}^{\langle I\rangle,(n)}$ using (17).
        end if
    end for
    Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \cdot \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{g}^{\langle I\rangle,(n)}$.
```

Table 1. Comparing computational complexities of EM and PU (probabilistic unrolling). In the worst case, $\tau_{\boldsymbol{\theta}}$ and $\omega_{\boldsymbol{\theta}}$ are $\mathcal{O}(D^2)$.

	Time	Space
EM	$\mathcal{O}(ND^3)$	$\mathcal{O}(ND^2)$
PU (Output Gradient)	$\mathcal{O}(NKI\tau_{\boldsymbol{\theta}})$	$\mathcal{O}(NKD + \omega_{\boldsymbol{\theta}})$
PU (Network Gradient)	$\mathcal{O}(NKI\tau_{\boldsymbol{\theta}})$	$\mathcal{O}(NKDI + \omega_{\boldsymbol{\theta}})$

4. Related Work

Efficient Learning with Linear Solvers. Using iterative solvers to circumvent matrix inversion is a widely-known technique within numerical linear algebra (Saad, 2003; Halko et al., 2011). Recently, solvers such as the Lanczos algorithm (Lanczos, 1950) and conjugate gradient (CG) (Hestenes & Stiefel, 1952) have become popular for accelerating gradient-based learning for Gaussian processes (Dong et al., 2017; Gardner et al., 2018; Wang et al., 2019; Wenger et al., 2022). In addition, Lin et al. (2022b) and Lin et al. (2022c) used CG to accelerate the classical EM algorithm for sparse Bayesian learning. Many of these works consider when the number of data vectors $N = 1$, as opposed to the setting of the LGM where N can be large. They also do not consider the idea of backpropagation through the solver.

Backpropagating through Optimization Algorithms. Automatic differentiation (or "backpropagation") (Baydin et al., 2018) has been widely used and studied in machine learning (Domke, 2012; Deledalle et al., 2014; Shaban et al., 2019). Domke (2012) studied truncated backpropagation as a replacement for implicit differentiation (e.g. Foo et al., 2007; Blondel et al., 2022; Bertrand et al., 2022) when performing incomplete energy minimization. Shaban et al. (2019) studied the use of truncated backpropagation for parameter estimation using unrolled networks. Backpropagating through an unrolled parameter estimation mapping has also been applied to hyperparameter optimization (Maclaurin et al., 2015; Franceschi et al., 2018), and constructing generative adversarial networks (Metz et al., 2016). Ablin et al. (2020) theoretically studied how backpropagation can accelerate gradient estimation for bilevel (i.e. min-min) optimization problems, in the setting where the inner and outer objectives are the same, and when the inner optimization algorithm is gradient descent. Moreover, Tolooshams & Ba (2022); Malézieux et al. (2021) studied the acceleration phenomenon for the sparse coding problem. This paper differs from the aforementioned prior work as follows: (a) probabilistic unrolling is designed for the specific setting of the LGM (as opposed to the general energy minimization problem of Domke (2012)), and contains a novel Monte Carlo sampling step to avoid inversion of the covariance matrix, (b) the fact that our inner optimization originates from this sampling step necessitates statistical considerations and analyses absent from previous work, (c) we extend the result of Ablin et al. (2020), showing that backpropagation can accelerate gradient estimation even in cases in which the inner and outer objectives are different, and (d) we provide gradient convergence analysis for steepest descent (an algorithm that is more sophisticated than gradient descent, requiring analysis of backpropagation through the step size).

Unrolled Networks. Our interpretation of unrolled solvers as a deep neural network is known as unrolled/unfolded networks in the literature. Gregor & LeCun (2010) introduced this approach for solving the sparse coding problem. Prior works designed and studied deep unrolled networks (Chen et al., 2018; Ablin et al., 2019). Moreover, unrolled networks have found advantages in various applications such as compressed sensing MRI (Sun et al., 2016), Poisson image denoising (Tolooshams et al., 2020), and pattern learning from physiological data (Malézieux et al., 2021).

Variational EM and Variational Auto-Encoders. Variational inference (VI) is a popular approach for approximating posterior distributions with simpler surrogates. Using VI for the E-Step of EM leads to the *variational EM (VEM)* algorithm (Murphy, 2023, Sec. 10.3.5), which is a potential alternative to probabilistic unrolling for accelerating EM inference. VEM is more flexible than probabilistic unrolling because it can perform inference for models outside the LGM family. However, the most common form of VEM learns a "mean-field" approximation to the posterior, which does not model covariance between latent variables and therefore biases the learning process away from the negative log-likelihood objective $\mathcal{L}(\boldsymbol{\theta})$ (2) (Lin et al., 2022b); in contrast, probabilistic unrolling captures rich covariance structure using samples from the true posterior and like EM, still optimizes $\mathcal{L}(\boldsymbol{\theta})$ as its central objective. The variational auto-encoder (VAE) (Kingma & Welling, 2013) is one of the most widely-used instances of VEM that trains a deep neural network to perform VI. Although VAEs are efficient tools for inference, they (a) require a separate inference network that is different from the generative model, increasing the number of parameters for training, and (b) require custom design of this network's architecture (e.g. layers, activations, etc.). In contrast, the probabilistic unrolling architecture (Fig. 1) is based on an interpretable linear solver that uses the same parameters as the generative model.

5. Theoretical Analysis

How well probabilistic unrolling approximates the exact EM gradient depends on the number of solver iterations, and the quality of the Monte Carlo approximation. We conduct a theoretical analysis of these two sources of error. We begin by defining population-level quantities for each gradient,

$$\boldsymbol{h}^{\star} := \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{g}^{\star,(n)} \qquad \boldsymbol{h}^{\#} := \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{g}^{\#,(n)} \tag{18}$$

$$\widehat{\boldsymbol{h}}^{\langle I\rangle} := \frac{1}{N}\sum_{n=1}^{N} \widehat{\boldsymbol{g}}^{\langle I\rangle,(n)} \quad \widetilde{\boldsymbol{h}}^{\langle I\rangle} := \frac{1}{N}\sum_{n=1}^{N} \widetilde{\boldsymbol{g}}^{\langle I\rangle,(n)},$$

where $\boldsymbol{h}^{\star}(\boldsymbol{\theta}) = \nabla_1 \mathcal{Q}(\boldsymbol{\theta}|\boldsymbol{\theta})$ is the exact gradient EM update of (4). We denote the approximate gradient after I iterations of probabilistic unrolling by $\boldsymbol{h}^{\langle I\rangle}$, with variants $\widehat{\boldsymbol{h}}^{\langle I\rangle}$ and $\widetilde{\boldsymbol{h}}^{\langle I\rangle}$ corresponding, respectively, to the output and network gradients defined previously. The quantity of interest is

$$\|\boldsymbol{h}^{\star} - \boldsymbol{h}^{\langle I\rangle}\| \leq \underbrace{\|\boldsymbol{h}^{\star} - \boldsymbol{h}^{\#}\|}_{\text{statistical error}} + \underbrace{\|\boldsymbol{h}^{\#} - \boldsymbol{h}^{\langle I\rangle}\|}_{\text{optimization error}}, \tag{19}$$

which decomposes into two terms. The first term, which we name *statistical error*, comes from approximating $\boldsymbol{h}^{\star}$ with Monte Carlo samples. We name the second term *optimization error*: this term captures the error due to performing a finite number I of iterations of the linear solver.

5.1. Statistical Error

Given $\boldsymbol{\theta} \in \Theta$, we first bound $\|\boldsymbol{h}^{\star}(\boldsymbol{\theta}) - \boldsymbol{h}^{\#}(\boldsymbol{\theta})\|_{\infty}$.

Proposition 5.1. *Let N be the number of data points, K be the number of samples for each data point, and L be the dimesionality of $\boldsymbol{\theta}$. For every $n \in \{1, \ldots, N\}$ we define*

$$\mathbf{M}^{(n,\ell)} := (\boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{(n)})^{1/2} \frac{\partial \mathbf{A}_{\boldsymbol{\theta}}^{(n)}}{\partial \theta_{\ell}} (\boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{(n)})^{1/2}, \tag{20}$$

where $\mathbf{A}_{\boldsymbol{\theta}}^{(n)}$ is defined in (6), $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{(n)}$ is defined in (7), and $\frac{\partial \mathbf{A}_{\boldsymbol{\theta}}^{(n)}}{\partial \theta_{\ell}}$ is the $D \times D$ matrix of partial derivatives of the entries of $\mathbf{A}_{\boldsymbol{\theta}}^{(n)}$ with respect to θ_{ℓ}. Let $\xi := \max_{\ell} \max_{n} \|\mathbf{M}^{(n,\ell)}\|_F$, where $\|\cdot\|_2$ denotes the spectral norm and $\|\cdot\|_F$ denotes Frobenius norm.

Then, there is an absolute constant C such that if the number of Monte Carlo samples K satisfies

$$K \geq \frac{\log(4NL)}{C} \max_{\ell} \left(\frac{\max_{n} \|\mathbf{M}^{(n,\ell)}\|_2^2}{\sum_{n} \|\mathbf{M}^{(n,\ell)}\|_F^2} \right), \tag{21}$$

it follows that

$$\Pr\left(\|\boldsymbol{h}^{\star} - \boldsymbol{h}^{\#}\|_{\infty} > \xi \sqrt{\frac{\log(4NL)}{CNK}} \right) \leq \frac{1}{N}. \tag{22}$$

We give the proof in Appendix E.1. The implication of Prop. 5.1 is that with high probability, $\boldsymbol{h}^{\#}$ is close to $\boldsymbol{h}^{\star}$. The condition in (21) is a mild Monte Carlo sample size requirement and is satisfied for instance if $K \geq \frac{\log(4NL)}{C \max(1, \kappa N)}$, where κ is any number such that for all ℓ, n, n', $\frac{\|\mathbf{M}^{(n,\ell)}\|_2^2}{\|\mathbf{M}^{(n',\ell)}\|_2^2} \geq \kappa$.

5.2. Optimization Error

Next, we bound optimization error $\|\boldsymbol{h}^{\#}(\boldsymbol{\theta}) - \boldsymbol{h}^{\langle I\rangle}(\boldsymbol{\theta})\|_2$.

Proposition 5.2. *Let I denote the number of linear solver iterations. Then, the output gradient $\widehat{\boldsymbol{h}}^{\langle I\rangle}$ and the network gradient $\widetilde{\boldsymbol{h}}^{\langle I\rangle}$ converge to $\boldsymbol{h}^{\#}$ with the following rates:*

$$\|\boldsymbol{h}^{\#} - \widehat{\boldsymbol{h}}^{\langle I\rangle}\|_2 = \mathcal{O}(\rho^{I}), \quad \|\boldsymbol{h}^{\#} - \widetilde{\boldsymbol{h}}^{\langle I\rangle}\|_2 = \mathcal{O}(I\rho^{2I}),$$

where $\rho < 1$ is the solver convergence rate. For gradient descent (GD) and steepest descent (SD), these rates are

$$\rho_{\mathrm{GD}} := \frac{\iota - 1}{\iota}, \qquad \rho_{\mathrm{SD}} := \frac{\iota - 1}{\iota + 1}, \tag{23}$$

where ι denotes the condition number (i.e. ratio between largest and smallest eigenvalues) of the matrix $\mathbf{A}_{\boldsymbol{\theta}}$ (6).

From Prop. 5.2, we draw three conclusions: First, both the output gradient $\widehat{\boldsymbol{h}}^{\langle I\rangle}$ and the network gradient $\widetilde{\boldsymbol{h}}^{\langle I\rangle}$ converge to $\boldsymbol{h}^{\#}$ as $I \to \infty$. Second, $\widetilde{\boldsymbol{h}}^{\langle I\rangle}$ achieves asymptotically better estimation of $\boldsymbol{h}^{\#}$ (compared to $\widehat{\boldsymbol{h}}^{\langle I\rangle}$). Third, the results suggest that the error in both gradients can be decreased by the use of solvers that converge faster than gradient descent, e.g., using steepest descent (as shown in Prop. 5.2), or conjugate gradient (CG), which has convergence rate $\rho_{\mathrm{CG}} = \frac{\sqrt{\iota}-1}{\sqrt{\iota}+1}$ (Shewchuk et al., 1994).

The proof of Prop. 5.2 is given in Appendix E.2. It relies on a connection we build between probabilistic unrolling and *bilevel optimization*, i.e. minimizing functions defined as a minimum (Ablin et al., 2020). Probabilistic unrolling (15) is an instance of bilevel optimization in which the outer level optimizes the EM objective by estimating its gradient with respect to parameters $\boldsymbol{\theta}$. This gradient is itself dependent on the solutions $\boldsymbol{\mu}_{\boldsymbol{\theta}}, \{\boldsymbol{\sigma}_{k,\boldsymbol{\theta}}\}_{k=1}^{K}$ of $K + 1$ linear systems, each equivalent to minimizing an inner quadratic function that depends on $\boldsymbol{\theta}$. As part of our proof, we introduce the following two lemmas, which may be of broader interest beyond our particular setting of probabilistic unrolling for LGMs. The first result (Lemma 5.3, proof in Appendix E.3) is a general statement on gradient convergence for bilevel optimization problems; it extends Prop. 2.2 of Ablin et al. (2020) to settings in which the outer and inner objectives have different forms. The second result (Lemma 5.4, proof in Appendix E.4) analyzes Jacobian convergence for iterative solvers based on gradient descent and steepest descent.

Lemma 5.3. *Consider a bilevel optimization problem with outer objective $r(\boldsymbol{\theta}, \boldsymbol{\beta})$ and inner objective $s(\boldsymbol{\theta}, \boldsymbol{\beta})$,*

$$\min_{\boldsymbol{\theta}} \; r(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#}) \quad \textit{s.t.} \quad \boldsymbol{\beta}^{\#} := \arg\min_{\boldsymbol{\beta}} s(\boldsymbol{\theta}, \boldsymbol{\beta}), \tag{24}$$

in which the gradients $\{\nabla_1 r(\boldsymbol{\theta}, \boldsymbol{\beta}), \nabla_2 s(\boldsymbol{\theta}, \boldsymbol{\beta})\}$ and the second derivatives $\{\nabla^2_{22} s(\boldsymbol{\theta}, \boldsymbol{\beta}), \nabla^2_{12} r(\boldsymbol{\theta}, \boldsymbol{\beta})\}$ are Lipschitz continuous in $\boldsymbol{\beta}$. Let $\boldsymbol{g}^{\#} := \nabla_1 r(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#})$ be the desired gradient. Let $\boldsymbol{\beta}^{\langle I \rangle}$ denote an approximation of $\boldsymbol{\beta}^{\#}$ obtained from running an iterative (and differentiable) optimizer for I steps. We use $\boldsymbol{\beta}^{\langle I \rangle}$ to define two approximate gradients: (1) the analytic gradient (called "output gradient" in our work) $\widehat{\boldsymbol{g}}^{\langle I \rangle} := \nabla_1 r(\boldsymbol{\theta}, \boldsymbol{\beta}^{\langle I \rangle})$ and (2) the automatic gradient (called "network gradient" in our work) $\widetilde{\boldsymbol{g}}^{\langle I \rangle} := \nabla_1 r(\boldsymbol{\theta}, \boldsymbol{\beta}^{\langle I \rangle}) + \frac{\partial \boldsymbol{\beta}^{\langle I \rangle}}{\partial \boldsymbol{\theta}} \cdot \nabla_2 s(\boldsymbol{\theta}, \boldsymbol{\beta}^{\langle I \rangle})$. Additionally, define the Jacobians $\boldsymbol{J}^{\#} := \frac{\partial \boldsymbol{\beta}^{\#}}{\partial \boldsymbol{\theta}}$ and $\boldsymbol{J}^{\langle I \rangle} := \frac{\partial \boldsymbol{\beta}^{\langle I \rangle}}{\partial \boldsymbol{\theta}}$, and let $\boldsymbol{J}^{\langle I \rangle}$ be bounded (i.e. $\|\boldsymbol{J}^{\langle I \rangle}\|_2 \le J_M$). If the outer and inner objectives share second-order derivatives, i.e. $\nabla^2_{12} r(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#}) = \nabla^2_{12} s(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#})$, then the analytic and automatic gradients converge at the following rates:

$$\|\widehat{\boldsymbol{g}}^{\langle I \rangle} - \boldsymbol{g}^{\#}\|_2 = \mathcal{O}(\|\boldsymbol{\beta}^{\langle I \rangle} - \boldsymbol{\beta}^{\#}\|_2), \tag{25}$$
$$\|\widetilde{\boldsymbol{g}}^{\langle I \rangle} - \boldsymbol{g}^{\#}\|_2 = \mathcal{O}(\|\boldsymbol{\beta}^{\langle I \rangle} - \boldsymbol{\beta}^{\#}\|_2 \cdot \|\boldsymbol{J}^{\langle I \rangle} - \boldsymbol{J}^{\#}\|_2).$$

Lemma 5.4. *Given the bilevel optimization problem from Prop. 5.3, let the inner objective $s(\boldsymbol{\theta}, \boldsymbol{\beta}) := \frac{1}{2}\boldsymbol{\beta}^\top \mathbf{A}_{\boldsymbol{\theta}} \boldsymbol{\beta} - \boldsymbol{u}_{\boldsymbol{\theta}}^\top \boldsymbol{\beta}$ be a strongly convex quadratic function with positive definite $\mathbf{A}_{\boldsymbol{\theta}}$. Given $\boldsymbol{\theta}$, let $\boldsymbol{\beta}^{\langle I \rangle} :=$ LinearSolver$(\mathbf{A}_{\boldsymbol{\theta}}, \boldsymbol{u}_{\boldsymbol{\theta}}, I)$ be the output of an I-step linear solver used to approximate $\boldsymbol{\beta}^{\#} := \arg\min_{\boldsymbol{\beta}} s(\boldsymbol{\theta}, \boldsymbol{\beta}) = \mathbf{A}_{\boldsymbol{\theta}}^{-1} \boldsymbol{u}_{\boldsymbol{\theta}}$. Then, for gradient descent and steepest descent as the linear solver, the Jacobian error is the following function of solver error: $\|\boldsymbol{J}^{\langle I \rangle} - \boldsymbol{J}^{\#}\|_2 = \mathcal{O}(I \cdot \|\boldsymbol{\beta}^{\langle I \rangle} - \boldsymbol{\beta}^{\#}\|_2)$.*

Insights on Unrolling Depth Taken together, Prop. 5.1 and 5.2 offer insights in choosing the number of unrolling iterations I. Since the overall gradient error is the sum of the optimization and statistical errors[1], the latter being impervious to I, the results suggest taking I just large enough to balance the two sources of error. A rough calculation yields $I \approx C \log(NK)/\log(1/\rho)$ for output gradient and $I \approx C \log(NK)/\log(1/\rho^2)$ for network gradient, for some dimension-dependent constant C, where ρ is the convergence rate of the solver.

6. Experiments

We perform experiments on several LGM applications, ranging from recovering unknown parameters to solving inverse problems to predicting movie ratings. We demonstrate that probabilistic unrolling provides significant scalability over EM, without loss in model performance. In all instances of EM, we use a single gradient step for the M-Step update (i.e. gradient EM). We implement all algorithms in PyTorch and on a single Nvidia T4 GPU with 16 GB RAM.

The main hyperparameters for probabilistic unrolling are the number of samples K and the number of solver iterations I. When solving a linear system $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$, we let I be just large enough so that the residual error $\|\boldsymbol{b} - \mathbf{A}\boldsymbol{x}^{\langle I \rangle}\|_2^2$ is below some small threshold (i.e. 10^{-8}). We set K based on our theoretical analysis (21). We keep K small if either the number of data points N is large or the number of parameters L is small; otherwise we increase K. In our experiments, we find that having I and K in the range [10, 30] is sufficient even when D increases to (tens of) thousands of dimensions.

6.1. Parameter Recovery for Noisy AR Models

The noisy auto-regressive (AR) model is a time series model with applications in radar (Çayır & Candan, 2021) and biomedical imaging (Luo et al., 2020). A noisy AR model of order P for a time series $\boldsymbol{y} := \{y_d\}_{d=1}^D$ is written as

$$\{z_1, \dots, z_P\} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{\boldsymbol{\phi}}), \quad \mathbf{Q}_{\boldsymbol{\phi}} \in \mathbb{R}^P, \tag{26}$$
$$z_d = \sum_{p=1}^{P} \phi_p \cdot z_{d-p} + w_d, \quad w_d \sim \mathcal{N}(0, \kappa), \quad P < d \le D,$$
$$y_d = z_d + v_d, \qquad v_d \sim \mathcal{N}(0, \lambda), \quad 1 \le d \le D.$$

The initial covariance matrix $\mathbf{Q}_{\boldsymbol{\phi}}$ is some function of the AR coefficients $\boldsymbol{\phi} := \{\phi_1, \dots, \phi_P\}$ that ensures stationarity for the latent process (see Appendix F.1.1 for details). The model's free parameters are $\boldsymbol{\theta} = \{\boldsymbol{\phi}, \lambda, \kappa\}$. We can write this model as an LGM (1), where $\boldsymbol{\nu}_{\boldsymbol{\theta}} = \mathbf{0}$, $\boldsymbol{\Phi}_{\boldsymbol{\theta}} = \mathbf{I}$, $\boldsymbol{\eta}_{\boldsymbol{\theta}} = \mathbf{0}$, $\boldsymbol{\Psi}_{\boldsymbol{\theta}} = \lambda^{-1}\mathbf{I}$, and $\boldsymbol{\Gamma}_{\boldsymbol{\theta}}$ is a function of $\{\boldsymbol{\phi}, \kappa\}$.

Complexity Comparison. Using matrix inversion, exact-gradient EM will require $\mathcal{O}(D^3)$-time and $\mathcal{O}(D^2)$-space. In comparison, probabilistic unrolling scales with the time $\tau_{\boldsymbol{\theta}}$ and space $\omega_{\boldsymbol{\theta}}$ needed for matrix-vector multiplication with the posterior inverse-covariance matrix $\mathbf{A}_{\boldsymbol{\theta}}$ (6). For the noisy AR model of order P, $\mathbf{A}_{\boldsymbol{\theta}}$ is a banded matrix with $2P+1$ non-zero bands (derivation given in Appendix F.1.1). As a result, $\tau_{\boldsymbol{\theta}} = \mathcal{O}(DP + P^3)$ and $\omega_{\boldsymbol{\theta}} = \mathcal{O}(DP + P^2)$,

[1] Using the fact that the ℓ_2-norm is an upper-bound on the ℓ_∞-norm, we can bound the overall error (19) in ℓ_∞ norm (with probability $1 - 1/N$) by adding the results of Prop. 5.1 and 5.2.

Table 2. Comparing percent error in noisy AR parameter recovery and computation time for EM and probabilistic unrolling (PU).

D	$r(\boldsymbol{\phi}^{\text{EM}}, \boldsymbol{\phi}^\star)$	$r(\boldsymbol{\phi}^{\text{PU}}, \boldsymbol{\phi}^\star)$	$r(\kappa^{\text{EM}}, \kappa^\star)$	$r(\kappa^{\text{PU}}, \kappa^\star)$	$r(\lambda^{\text{EM}}, \lambda^\star)$	$r(\lambda^{\text{PU}}, \lambda^\star)$	EM Time (Best)	PU Time
1,000	7.5±4.7 %	6.8±3.1 %	3.6±4.4 %	3.1±1.5 %	5.0±2.9 %	6.0±3.7 %	41±0 s	**8±0 s**
3,000	3.0±2.5 %	3.7±2.1 %	3.1±2.2 %	3.6±2.6 %	2.8±3.6 %	3.0±3.5 %	413±2 s	**10±0 s**
10,000	1.8±1.1 %	2.5±2.2 %	1.3±0.8 %	1.5±0.7 %	1.3±0.3 %	1.0±0.5 %	1361±36 s	**29±0 s**
30,000	0.5±0.2 %	0.4±0.2 %	0.4±0.1 %	0.4±0.2 %	0.7±0.1 %	0.8±0.2 %	4139±49 s	**87±1 s**

Table 3. Averaged CS results (see Appendix F.2.4 for breakdown by digit type). Without Woodbury identity, EM time is 4725±61 s.

	$r(\boldsymbol{\mu}^{\text{EM}}, \tilde{\boldsymbol{z}})$	$r(\boldsymbol{\mu}^{\text{PU}}, \tilde{\boldsymbol{z}})$	EM Time	PU Time
Avg.	4.8±1.0 %	4.7±1.4 %	1481±19 s	**21±0 s**

which is much more efficient than EM.[2]

Setup and Results. We compare the accuracy and speed of exact-gradient EM and probabilistic unrolling in parameter recovery for noisy AR models of order $P = 5$. First, we randomly sample a set of true parameters $\{\boldsymbol{\phi}^\star, \lambda^\star, \kappa^\star\}$, generate $N = 5$ time series according to (26), and randomly mask out 10% of the observations from each time series to create $\tilde{\boldsymbol{y}}^{(1)}, \ldots, \tilde{\boldsymbol{y}}^{(5)}$. Then, we perform maximum likelihood estimation using either gradient EM or probabilistic unrolling to produce parameter estimates $\{\hat{\boldsymbol{\phi}}, \hat{\lambda}, \hat{\kappa}\}$. We measure accuracy using the normalized root-mean-square error (NRMSE) $r(\boldsymbol{\theta}, \boldsymbol{\theta}^\star) := \|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^\star\|_2 / \|\boldsymbol{\theta}^\star\|_2 \times 100\%$. For probabilistic unrolling, we use $K = 10$ Monte Carlo samples, unroll $I = 30$ iterations of the conjugate gradient solver, and use the network gradient. Other details can be found in Appendix F.1.3. We report results for different values of D in Table 2. Probabilistic unrolling consistently matches the performance of gradient EM, while being up to 47 times faster. For each D, we report the smaller of the times between EM with matrix inversion and EM with a Kalman smoother (see Appendix F.1.2). Typically, inversion is faster for smaller D while using the Kalman smoother is faster for larger D. Probabilistic unrolling is faster than both of these for all D. We additionally perform comparisons between probabilistic unrolling and variational EM (as implemented through the variational auto-encoder (VAE) (Kingma & Welling, 2013)) in Appendix F.1.4.

6.2. Bayesian Compressed Sensing of Sparse Signals

With applications from radio astronomy (Wiaux et al., 2009) to MRI (Lustig et al., 2008), compressed sensing (CS) is a technique for reconstructing sparse, high-dimensional signals $\tilde{\boldsymbol{z}}^{(n)}$ from measurements $\tilde{\boldsymbol{y}}^{(n)}$. *Bayesian compressed sensing* (Ji et al., 2008b; Bilgic et al., 2011; Lin et al., 2021; 2022a) is an approach to CS that employs the sparse Bayesian learning model (Wipf & Rao, 2004)

$$\boldsymbol{z}^{(n)} \sim \mathcal{N}(\mathbf{0}, \text{diag}(\boldsymbol{\alpha})^{-1}), \qquad n = 1, \ldots, N \quad (27)$$

$$\tilde{\boldsymbol{y}}^{(n)} | \boldsymbol{z}^{(n)} \sim \mathcal{N}(\boldsymbol{\Phi}^{(n)} \boldsymbol{z}^{(n)}, \beta^{-1}\mathbf{I}), \quad n = 1, \ldots, N,$$

where each $\boldsymbol{z}^{(n)} \in \mathbb{R}^D$ is an unknown signal, $\tilde{\boldsymbol{y}}^{(n)} \in \mathbb{R}^M$ is a measurement associated of the signal, and $\boldsymbol{\Phi}^{(n)} \in \mathbb{R}^{M \times D}$ is a so-called measurement matrix. The free parameters $\boldsymbol{\theta}$ of the model are $\boldsymbol{\alpha} \in \mathbb{R}^D$ and $\beta \in \mathbb{R}$. When a common sparsity pattern underlies the observations $\{\tilde{\boldsymbol{y}}^{(n)}\}_{n=1}^N$, maximum likelihood estimation will push many of the entries α_m to adopt large values, tending to ∞, and, thus, encouraging sparsity of samples from the posterior $p(\boldsymbol{z}^{(n)} | \tilde{\boldsymbol{y}}^{(n)}, \boldsymbol{\alpha}, \beta)$ (Yee & Atchadé, 2017). The mean $\boldsymbol{\mu}^{(n)}$ of each posterior is then used as an estimate for $\tilde{\boldsymbol{z}}^{(n)}$ (Ji et al., 2008a).

Complexity Comparison. In several applications (e.g. MRI, astronomy), each $\boldsymbol{\Phi}^{(n)} = \boldsymbol{\Omega}^{(n)} \boldsymbol{\Phi}$, where $\boldsymbol{\Phi} \in \mathbb{C}^{D \times D}$ is the Fourier transform and $\boldsymbol{\Omega}^{(n)} \in \mathbb{R}^{M \times D}$ is a random undersampling mask. Thus, (27) is an instance of the LGM, where $\boldsymbol{\theta} := \{\boldsymbol{\alpha}, \beta\}$. Using gradient EM to fit $\boldsymbol{\theta}$ requires $\mathcal{O}(D^3)$-time and $\mathcal{O}(D^2)$-space. On the other hand, probabilistic unrolling scales with the complexity needed to apply $\mathbf{A}_{\boldsymbol{\theta}}$ (6) to vectors; this is dominated by the Fourier transform $\boldsymbol{\Phi}$, which only requires $\mathcal{O}(D \log D)$-time and $\mathcal{O}(D)$-space.

Setup and Results. We perform CS experiments on NIST (Grother, 1995), a dataset of handwritten digits. For each digit type (i.e. 0 through 9), we sample $N = 10$ images $\tilde{\boldsymbol{z}}^{(n)}$ of size 128 × 128, which are high-dimensional signals with $D = 16{,}384$ pixels. Each image is naturally sparse because most pixels are zero. For each $\tilde{\boldsymbol{z}}^{(n)}$, we randomly undersample its 2D Fourier transform by 15% (i.e. $M = 0.15D$) and add noise to construct the measurement $\tilde{\boldsymbol{y}}^{(n)}$. Then, we fit a Bayesian compressed sensing model (27) to $\{\tilde{\boldsymbol{y}}^{(n)}\}_{n=1}^N$ to obtain reconstructions $\{\boldsymbol{\mu}^{(n)}\}_{n=1}^N$. We measure success using the NRMSE between $\boldsymbol{\mu}$ and $\tilde{\boldsymbol{z}}$, where $\boldsymbol{\mu}, \tilde{\boldsymbol{z}} \in \mathbb{R}^{ND}$ are the concatenations of $\{\boldsymbol{\mu}^{(n)}\}_{n=1}^N$ and the true signals $\{\tilde{\boldsymbol{z}}^{(n)}\}_{n=1}^N$, respectively. For probabilistic unrolling, we use $K = 30$ samples, $I = 25$ iterations of preconditioned conjugate gradient, and the network gradient. More details can be found in Appendix F.2.2. Results averaged over the 10 different digit types are given in Table 3. We find that

[2] We note that instead of using matrix inversion, we could cast (26) as a state-space model and use a Kalman smoother to run exact-gradient EM in $\mathcal{O}(DP^3)$-time and $\mathcal{O}(DP^2)$-space (see Appendix F.1.2). However, unlike probabilistic unrolling, the Kalman filter is a sequential algorithm and does not parallelize across D.

Table 4. MovieLens results. For timing, a *cycle* is defined as 2,000 gradient steps. EM requires too much memory to run ML-25M.

Dataset	N (users)	M (movies)	EM RMSE	PU RMSE	EM Time/Cycle	PU Time/Cycle	PU Mem	PU Mem
ML-1M	6,000	4,000	0.8433	0.8436	54 min, 42 s	**5 min, 50 s**	1.94 GB	**0.17 GB**
ML-10M	72,000	10,000	0.7809	0.7796	78 min, 36 s	**12 min, 8 s**	5.62 GB	**2.64 GB**
ML-25M	162,000	62,000	—	0.7700	—	**31 min, 11 s**	>16 GB	**8.48 GB**

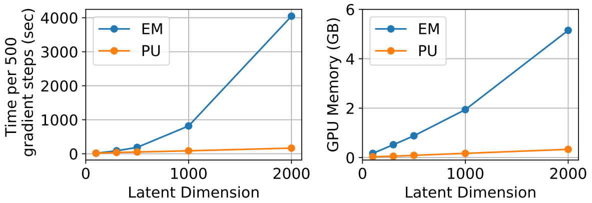

Figure 2. Time and memory versus D for the ML-1M dataset.

probabilistic unrolling and gradient EM have similar error. However, probabilistic unrolling is approximately 70 times faster than gradient EM, even after we accelerate EM using the Woodbury matrix identity (see Appendix F.2.1).

6.3. Collaborative Filtering through Factor Analysis

The goal of recommender systems is to predict user ratings for various items. One common approach is *collaborative filtering*, in which we pool together incomplete ratings data for M items across N users to infer how all users would rate all items. One of the central challenges of collaborative filtering is the inherent sparsity of the data – for every user, we typically only observe ratings for a small fraction of items, leading to large amounts of missing data (Rendle et al., 2020; Wu et al., 2021). In this section, we use *factor analysis* models for collaborative filtering. Factor analysis is a Bayesian analog of matrix factorization, one of the state-of-the-art methods for recommender systems (Koren et al., 2009; Lawrence & Urtasun, 2009; Rendle et al., 2019).

Let $\boldsymbol{y}^{(n)} \in \mathbb{R}^M$ be the ratings for user n across M movies. Only part of this vector is known: $\tilde{\boldsymbol{y}}^{(n)} = \boldsymbol{\Omega}^{(n)}\boldsymbol{y}^{(n)} \in \mathbb{R}^{M_n}$, where $M_n < M$. The factor analysis model is written as

$$\begin{aligned} \boldsymbol{z}^{(n)} &\sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \\ \tilde{\boldsymbol{y}}^{(n)} | \boldsymbol{z}^{(n)} &\sim \mathcal{N}(\boldsymbol{\Omega}^{(n)}(\boldsymbol{\Phi}\boldsymbol{z}^{(n)} + \boldsymbol{\eta}), \boldsymbol{\Omega}^{(n)}\boldsymbol{\Psi}^{-1}(\boldsymbol{\Omega}^{(n)})^\top), \end{aligned} \tag{28}$$

where each $\boldsymbol{z}^{(n)} \in \mathbb{R}^D$ for $D < M$ is a set of latent factors for user n. The free parameters of this model are $\boldsymbol{\theta} := \{\boldsymbol{\Phi}, \boldsymbol{\eta}, \boldsymbol{\Psi}\}$, where $\boldsymbol{\Phi} \in \mathbb{R}^{M \times D}$, $\boldsymbol{\eta} \in \mathbb{R}^M$, and $\boldsymbol{\Psi}$ is a diagonal $M \times M$ matrix. After estimating $\boldsymbol{\theta}$, we can predict any unknown rating $y_m^{(n)} \notin \tilde{\boldsymbol{y}}^{(n)}$ using the mean of the distribution $p(y_m^{(n)} | \tilde{\boldsymbol{y}}^{(n)}, \boldsymbol{\theta})$ (i.e. $\hat{y}_m^{(n)} = \boldsymbol{\phi}_m^\top \boldsymbol{\mu}_{\boldsymbol{\theta}}^{(n)} + \eta_m$, where $\boldsymbol{\mu}_{\boldsymbol{\theta}}^{(n)}$ is defined by (7) and $\boldsymbol{\phi}_m$ is the m-th row of $\boldsymbol{\Phi}$).

Setup and Results. We perform collaborative filtering experiments on MovieLens (Harper & Konstan, 2015), a group of successively larger datasets with $R = 1$ million, 10 million, and 25 million ratings of thousands of movies, by thousands of users. For each dataset, we perform a 90%-10% train-test split of the ratings data (Sedhain et al., 2015). Then, we fit a factor analysis model to the training set using mini-batch gradient descent, where the gradients are calculated using either gradient EM or probabilistic unrolling. For probabilistic unrolling, we use $K = 10$ Monte Carlo samples, $I = 10$ unrolled iterations of conjugate gradient, and the output gradient to reduce memory consumption. After convergence, we calculate the root-mean-square error between all ratings in the test set $y_m^{(n)}$ and the fitted model's predictions $\hat{y}_m^{(n)}$. Further experimental details can be found in Appendix F.3.1. The results for the three MovieLens datasets are given in Table 4. We also report processing time and GPU memory utilized by EM and probabilistic unrolling. All of the results in Table 4 are for $D = 1{,}000$ latent factors. Figure 2 shows a plot of time/memory vs. D for other values of D. An additional VAE baseline is provided in Appendix F.3.2.

7. Conclusion

We introduced *probabilistic unrolling*, a computational framework for accelerating gradient-based maximum likelihood estimation for a large class of latent variable models with Gaussian prior and Gaussian likelihood. Our method combines Monte Carlo sampling with iterative solvers and unrolled optimization, leading to a novel means of backpropagating through a sampling algorithm. Our theoretical analyses demonstrated that this can accelerate gradient estimation and, hence, maximum likelihood estimation. Our analyses provide insight into the relationship between the number of solver iterations, i.e. network depth, the number of Monte Carlo samples, and the gradient approximation error. In the future, we will consider extensions of probabilistic unrolling to other classes of probabilistic latent variable models.

Acknowledgements

This work was supported by a National Defense Science and Engineering Graduate Fellowship, and grants PHY-2019786, DMS-2015485, and DMS-2210664 from the National Science Foundation. The authors also thank the anonymous reviewers, whose comments greatly improved this paper.

References

Ablin, P., Moreau, T., Massias, M., and Gramfort, A. Learning step sizes for unfolded sparse coding. *Advances in Neural Information Processing Systems*, 32, 2019.

Ablin, P., Peyré, G., and Moreau, T. Super-efficiency of automatic differentiation for functions defined as a minimum. In *International Conference on Machine Learning*, pp. 32–41. PMLR, 2020.

Balakrishnan, S., Wainwright, M. J., and Yu, B. Statistical guarantees for the EM algorithm: From population to sample-based analysis. *The Annals of Statistics*, 45(1): 77–120, 2017.

Basilevsky, A. T. *Statistical factor analysis and related methods: theory and applications*. John Wiley & Sons, 2009.

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. Automatic differentiation in machine learning: a survey. *Journal of Marchine Learning Research*, 18:1–43, 2018.

Bertrand, Q., Klopfenstein, Q., Massias, M., Blondel, M., Vaiter, S., Gramfort, A., and Salmon, J. Implicit differentiation for fast hyperparameter selection in non-smooth convex learning. *The Journal of Machine Learning Research*, 23(1):6680–6722, 2022.

Bilgic, B., Goyal, V. K., and Adalsteinsson, E. Multi-contrast reconstruction with Bayesian compressed sensing. *Magnetic resonance in medicine*, 66(6):1601–1615, 2011.

Bishop, C. M. and Nasrabadi, N. M. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

Blondel, M., Berthet, Q., Cuturi, M., Frostig, R., Hoyer, S., Llinares-López, F., Pedregosa, F., and Vert, J.-P. Efficient and modular implicit differentiation. *Advances in Neural Information Processing Systems*, 35:5230–5242, 2022.

Çayır, Ö. and Candan, Ç. Maximum likelihood autoregressive model parameter estimation with noise corrupted independent snapshots. *Signal Processing*, 186:108118, 2021.

Chen, X., Liu, J., Wang, Z., and Yin, W. Theoretical linear convergence of unfolded ISTA and its practical weights and thresholds. *Advances in Neural Information Processing Systems*, 31, 2018.

Deledalle, C.-A., Vaiter, S., Fadili, J., and Peyré, G. Stein unbiased gradient estimator of the risk (SUGAR) for multiple parameter selection. *SIAM Journal on Imaging Sciences*, 7(4):2448–2487, 2014.

Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

Domke, J. Generic methods for optimization-based modeling. In Lawrence, N. D. and Girolami, M. (eds.), *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, volume 22 of *Proceedings of Machine Learning Research*, pp. 318–326, La Palma, Canary Islands, 21–23 Apr 2012. PMLR.

Dong, K., Eriksson, D., Nickisch, H., Bindel, D., and Wilson, A. G. Scalable log determinants for Gaussian process kernel learning. *Advances in Neural Information Processing Systems*, 30, 2017.

Durbin, J. and Koopman, S. J. *Time series analysis by state space methods*, volume 38. OUP Oxford, 2012.

Foo, C.-s., Ng, A., et al. Efficient multiple hyperparameter learning for log-linear models. *Advances in neural information processing systems*, 20, 2007.

Franceschi, L., Frasconi, P., Salzo, S., Grazzi, R., and Pontil, M. Bilevel programming for hyperparameter optimization and meta-learning. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1568–1577. PMLR, 10–15 Jul 2018.

Gardner, J., Pleiss, G., Weinberger, K. Q., Bindel, D., and Wilson, A. G. GPytorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. *Advances in neural information processing systems*, 31, 2018.

Gregor, K. and LeCun, Y. Learning fast approximations of sparse coding. In *Proceedings of the 27th international conference on international conference on machine learning*, pp. 399–406, 2010.

Grother, P. J. NIST special database 19-hand-printed forms and characters database. *Technical Report, National Institute of Standards and Technology*, 1995.

Halko, N., Martinsson, P.-G., and Tropp, J. A. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

Harper, F. M. and Konstan, J. A. The MovieLens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

Hestenes, M. R. and Stiefel, E. Methods of conjugate gradients for solving. *Journal of research of the National Bureau of Standards*, 49(6):409, 1952.

Hutchinson, M. F. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

Ji, S., Dunson, D., and Carin, L. Multitask compressive sensing. *IEEE Transactions on Signal Processing*, 57(1): 92–106, 2008a.

Ji, S., Xue, Y., and Carin, L. Bayesian compressive sensing. *IEEE Transactions on signal processing*, 56(6):2346–2356, 2008b.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Koren, Y., Bell, R., and Volinsky, C. Matrix factorization techniques for recommender systems. *Computer*, 42(8): 30–37, 2009.

Lanczos, C. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. 1950.

Lange, K. A gradient algorithm locally equivalent to the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 57(2):425–437, 1995.

Lawrence, N. D. and Urtasun, R. Non-linear matrix factorization with Gaussian processes. In *Proceedings of the 26th annual international conference on machine learning*, pp. 601–608, 2009.

Lin, A., Ba, D., and Bilgic, B. Accelerating Bayesian compressed sensing for fast multi-contrast reconstruction. In *Proceedings of the 30th Annual Meeting of ISMRM*, 2021.

Lin, A., Ba, D., and Bilgic, B. Bayesian sensitivity encoding enables parameter-free, highly accelerated joint multi-contrast reconstruction. In *Proceedings of the 31st Annual Meeting of ISMRM*, 2022a.

Lin, A., Song, A. H., Bilgic, B., and Ba, D. Covariance-free sparse Bayesian learning. *IEEE Transactions on Signal Processing*, 70:3818–3831, 2022b.

Lin, A., Song, A. H., Bilgic, B., and Ba, D. High-dimensional sparse Bayesian learning without covariance matrices. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1511–1515. IEEE, 2022c.

Luo, Q., Misaki, M., Mulyana, B., Wong, C.-K., and Bodurka, J. Improved autoregressive model for correction of noise serial correlation in fast fMRI. *Magnetic resonance in medicine*, 84(3):1293–1305, 2020.

Lustig, M., Donoho, D. L., Santos, J. M., and Pauly, J. M. Compressed sensing MRI. *IEEE signal processing magazine*, 25(2):72–82, 2008.

Maclaurin, D., Duvenaud, D., and Adams, R. Gradient-based hyperparameter optimization through reversible learning. In *International conference on machine learning*, pp. 2113–2122. PMLR, 2015.

Malézieux, B., Moreau, T., and Kowalski, M. Understanding approximate and unrolled dictionary learning for pattern recovery. In *International Conference on Learning Representations*, 2021.

Metz, L., Poole, B., Pfau, D., and Sohl-Dickstein, J. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*, 2016.

Murphy, K. P. *Probabilistic Machine Learning: Advanced Topics*. MIT Press, 2023. URL http://probml.github.io/book2.

Ober, S. W. and Rasmussen, C. E. Benchmarking the neural linear model for regression. *arXiv preprint arXiv:1912.08416*, 2019.

Rendle, S., Zhang, L., and Koren, Y. On the difficulty of evaluating baselines: A study on recommender systems. *arXiv preprint arXiv:1905.01395*, 2019.

Rendle, S., Krichene, W., Zhang, L., and Anderson, J. Neural collaborative filtering vs. matrix factorization revisited. In *Proceedings of the 14th ACM Conference on Recommender Systems*, pp. 240–248, 2020.

Saad, Y. *Iterative methods for sparse linear systems*. SIAM, 2003.

Sedhain, S., Menon, A. K., Sanner, S., and Xie, L. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th international conference on World Wide Web*, pp. 111–112, 2015.

Shaban, A., Cheng, C.-A., Hatch, N., and Boots, B. Truncated back-propagation for bilevel optimization. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1723–1732. PMLR, 2019.

Shewchuk, J. R. et al. An introduction to the conjugate gradient method without the agonizing pain, 1994.

Skilling, J. The eigenvalues of mega-dimensional matrices. *Maximum Entropy and Bayesian Methods: Cambridge, England, 1988*, pp. 455–466, 1989.

Sun, J., Li, H., Xu, Z., et al. Deep ADMM-net for compressive sensing MRI. *Advances in neural information processing systems*, 29, 2016.

Tipping, M. E. Sparse Bayesian learning and the relevance vector machine. *Journal of machine learning research*, 1 (Jun):211–244, 2001.

Tolooshams, B. and Ba, D. E. Stable and interpretable unrolled dictionary learning. *Transactions on Machine Learning Research*, 2022.

Tolooshams, B., Song, A., Temereanca, S., and Ba, D. Convolutional dictionary learning based auto-encoders for natural exponential-family distributions. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9493–9503. PMLR, 13–18 Jul 2020.

Ubaru, S., Chen, J., and Saad, Y. Fast estimation of tr(f(a)) via stochastic Lanczos quadrature. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1075–1099, 2017.

Wang, K., Pleiss, G., Gardner, J., Tyree, S., Weinberger, K. Q., and Wilson, A. G. Exact Gaussian processes on a million data points. *Advances in Neural Information Processing Systems*, 32, 2019.

Wenger, J., Pleiss, G., Hennig, P., Cunningham, J., and Gardner, J. Preconditioning for scalable Gaussian process hyperparameter optimization. In *International Conference on Machine Learning*, pp. 23751–23780. PMLR, 2022.

Wiaux, Y., Jacques, L., Puy, G., Scaife, A. M., and Vandergheynst, P. Compressed sensing imaging techniques for radio interferometry. *Monthly Notices of the Royal Astronomical Society*, 395(3):1733–1742, 2009.

Wipf, D. P. and Rao, B. D. Sparse Bayesian learning for basis selection. *IEEE Transactions on Signal processing*, 52(8):2153–2164, 2004.

Wu, D., Shang, M., Luo, X., and Wang, Z. An l 1-and-l 2-norm-oriented latent factor model for recommender systems. *IEEE Transactions on Neural Networks and Learning Systems*, 33(10):5775–5788, 2021.

Yee, C. C. and Atchadé, Y. F. On the sparse Bayesian learning of linear models. *Communications in Statistics-Theory and Methods*, 46(15):7672–7691, 2017.

A. Examples of LGMs

Factor Analysis This model sets $\boldsymbol{\nu}_{\boldsymbol{\theta}} \leftarrow \mathbf{0}$ and $\boldsymbol{\Gamma}_{\boldsymbol{\theta}} \leftarrow \mathbf{I}$ to constant values. Given data, it learns the entries of $\boldsymbol{\Phi}_{\boldsymbol{\theta}}, \boldsymbol{\eta}_{\boldsymbol{\theta}}$ and $\boldsymbol{\Psi}_{\boldsymbol{\theta}}$ as free parameters (Basilevsky, 2009).

Probabilistic PCA Probabilistic principal components analysis is a slight variation of factor analysis in which $\boldsymbol{\Psi}_{\boldsymbol{\theta}} = \beta\mathbf{I}$ has a single free parameter β that determines its constant diagonal values (**?**).

Sparse Bayesian Learning This model assumes that $\boldsymbol{\nu}_{\boldsymbol{\theta}} \leftarrow \mathbf{0}$ and $\boldsymbol{\eta}_{\boldsymbol{\theta}} \leftarrow \mathbf{0}$. The matrix $\boldsymbol{\Phi}_{\boldsymbol{\theta}}$ is a known and typically overcomplete (i.e. $D > M$) dictionary. The free parameters are $\boldsymbol{\theta} = \{\boldsymbol{\alpha}, \beta\}$, where $\boldsymbol{\alpha} \in \mathbb{R}^D$ determines the prior precision $\boldsymbol{\Gamma}_{\boldsymbol{\theta}} := \text{diag}(\boldsymbol{\alpha})$ and $\beta \in \mathbb{R}$ determines the likelihood precision $\boldsymbol{\Psi}_{\boldsymbol{\theta}} := \beta\mathbf{I}$. (Tipping, 2001; Wipf & Rao, 2004).

State-Space Model This is a popular time series model that generalizes other popular variants (e.g. auto-regressive processes, moving average processes) (Durbin & Koopman, 2012). For a single time series n, it is typically written as:

$$\begin{aligned} \boldsymbol{z}_m^{(n)} &= \mathbf{A}\boldsymbol{z}_{m-1}^{(n)} + \boldsymbol{w}_m^{(n)}, & \boldsymbol{w}_m^{(n)} &\sim \mathcal{N}(\mathbf{0}, \mathbf{Q}), \\ y_m^{(n)} &= \boldsymbol{c}^\top \boldsymbol{z}_m^{(n)} + v_m^{(n)}, & v_m^{(n)} &\sim \mathcal{N}(0, \sigma^2). \end{aligned} \tag{29}$$

At time step m, $\boldsymbol{z}_m^{(n)} \in \mathbb{R}^S$ is a latent state vector and $y_m^{(n)} \in \mathbb{R}$ is the observed data point. We can write (29) in the form of (1) by defining $\boldsymbol{z}^{(n)}$ (with length $D = S \cdot M$) as the concatentation of all $\boldsymbol{z}_m^{(n)}$ across m. The canonical parameters can then be written as functions of the free parameters $\boldsymbol{\theta} := \{\mathbf{A}, \boldsymbol{c}, \mathbf{Q}, \sigma^2\}$. Note that multiple time series can share these parameters in an LGM framework.

Bayesian Linear Regression Given a dataset of covariates and response variables $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_M, y_M)$ where each $\boldsymbol{x}_m \in \mathbb{R}^D$, Bayesian linear regression posits the model

$$y_m = \boldsymbol{x}_m^\top \boldsymbol{z} + \varepsilon_m, \tag{30}$$

where each regression weight $z_d \sim \mathcal{N}(0, 1/\alpha)$ and each noise variable $\varepsilon_m \sim \mathcal{N}(0, 1/\beta)$ for parameters $\alpha \in \mathbb{R}, \beta \in \mathbb{R}$. This is an LGM in which $N = 1$, $\boldsymbol{\nu}_{\boldsymbol{\theta}} \leftarrow \mathbf{0}$, and the rows of $\boldsymbol{\Phi}_{\boldsymbol{\theta}}$ are comprised of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M$. The free parameters $\boldsymbol{\theta}$ are $\{\alpha, \beta\}$ with $\boldsymbol{\Gamma}_{\boldsymbol{\theta}} := \alpha\mathbf{I}$ and $\boldsymbol{\Psi}_{\boldsymbol{\theta}} := \beta\mathbf{I}$ (Bishop & Nasrabadi, 2006) (Section 9.3.4).

Neural Linear Model One modern instance of Bayesian linear regression is the *neural linear model* (NLM):

$$y_m = \boldsymbol{f}_{\boldsymbol{\phi}}(\boldsymbol{x}_m)^\top \boldsymbol{z} + \varepsilon_m, \tag{31}$$

where $\boldsymbol{f}_{\boldsymbol{\phi}}$ is a neural network featurizer with weights $\boldsymbol{\phi}$ (**?**Ober & Rasmussen, 2019). Thus, different from traditional Bayesian linear regression, the canonical parameter $\boldsymbol{\Phi}_{\boldsymbol{\theta}}$ is now learned through $\boldsymbol{\phi}$. Therefore, the free parameteters are $\{\alpha, \beta, \boldsymbol{\phi}\}$. NLMs can learn very complicated, non-linear relationships (e.g. see Figure 1c in **?** and Figure 1 in Ober & Rasmussen (2019)). The LGM also covers multi-task versions of the NLM, in which we may have multiple target vectors $\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(N)}$ for $N > 1$ (**?**).

B. Derivation of q-Function for the LGM (5)

Taking into account missing data with the data mask $\boldsymbol{\Omega}^{(n)}$, the model in (1) becomes:

$$\begin{aligned} \boldsymbol{z}^{(n)} &\sim \mathcal{N}(\boldsymbol{\nu}_{\boldsymbol{\theta}}, \boldsymbol{\Gamma}_{\boldsymbol{\theta}}^{-1}), \\ \tilde{\boldsymbol{y}}^{(n)} | \boldsymbol{z}^{(n)} &\sim \mathcal{N}(\boldsymbol{\Omega}^{(n)}(\boldsymbol{\Phi}_{\boldsymbol{\theta}}\boldsymbol{z}^{(n)} + \boldsymbol{\eta}_{\boldsymbol{\theta}}), \boldsymbol{\Omega}^{(n)}\boldsymbol{\Psi}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{\Omega}^{(n)})^\top). \end{aligned} \tag{32}$$

From the definition of q in (3), we have (dropping the index n for notational convenience)

$$q(\boldsymbol{\theta}_1 | \boldsymbol{\theta}_2) := \mathbb{E}_{p(\boldsymbol{z}|\tilde{\boldsymbol{y}}, \boldsymbol{\Omega}, \boldsymbol{\theta}_2)}[-\log p(\boldsymbol{z}, \tilde{\boldsymbol{y}} | \boldsymbol{\Omega}, \boldsymbol{\theta}_1)]. \tag{33}$$

For any $\boldsymbol{\theta} \in \Theta$, the log-posterior can be written as

$$\log p(\boldsymbol{z} | \tilde{\boldsymbol{y}}, \boldsymbol{\Omega}, \boldsymbol{\theta}) = \log p(\boldsymbol{z}, \tilde{\boldsymbol{y}} | \boldsymbol{\Omega}, \boldsymbol{\theta}) - \log p(\tilde{\boldsymbol{y}} | \boldsymbol{\Omega}, \boldsymbol{\theta}), \tag{34}$$

where the second term is constant with respect to $\boldsymbol{z}$. Expanding the first term using the probability density function for multivariate Gaussians, we have

$$\begin{aligned}
&\log p(\boldsymbol{z}, \tilde{\boldsymbol{y}} | \boldsymbol{\Omega}, \boldsymbol{\theta}) = \log p(\tilde{\boldsymbol{y}} | \boldsymbol{z}, \boldsymbol{\Omega}, \boldsymbol{\theta}) + \log p(\boldsymbol{z} | \boldsymbol{\theta}) \\
&\cong -\frac{1}{2}(\tilde{\boldsymbol{y}} - \boldsymbol{\Omega}\boldsymbol{\Phi}_{\boldsymbol{\theta}}\boldsymbol{z} - \boldsymbol{\Omega}\boldsymbol{\eta}_{\boldsymbol{\theta}})^\top (\boldsymbol{\Omega}\boldsymbol{\Psi}_{\boldsymbol{\theta}}^{-1}\boldsymbol{\Omega}^\top)^{-1}(\tilde{\boldsymbol{y}} - \boldsymbol{\Omega}\boldsymbol{\Phi}_{\boldsymbol{\theta}}\boldsymbol{z} - \boldsymbol{\Omega}\boldsymbol{\eta}_{\boldsymbol{\theta}}) + \frac{1}{2}\log\det(\boldsymbol{\Omega}\boldsymbol{\Psi}_{\boldsymbol{\theta}}^{-1}\boldsymbol{\Omega}^\top)^{-1} \\
&\quad - \frac{1}{2}(\boldsymbol{z} - \boldsymbol{\nu}_{\boldsymbol{\theta}})^\top \boldsymbol{\Gamma}_{\boldsymbol{\theta}}(\boldsymbol{z} - \boldsymbol{\nu}_{\boldsymbol{\theta}}) + \frac{1}{2}\log\det\boldsymbol{\Gamma}_{\boldsymbol{\theta}},
\end{aligned} \tag{35}$$

where $\cong$ denotes equality up to additive constants with respect to $\boldsymbol{z}$ and $\boldsymbol{\theta}$. Note that $(\boldsymbol{\Omega}\boldsymbol{\Psi}_{\boldsymbol{\theta}}^{-1}\boldsymbol{\Omega}^\top)^{-1} = \boldsymbol{\Omega}\boldsymbol{\Psi}_{\boldsymbol{\theta}}\boldsymbol{\Omega}^\top$ because $\boldsymbol{\Psi}_{\boldsymbol{\theta}}$ is a diagonal matrix. This leads to the simplification

$$\begin{aligned}
\log p(\boldsymbol{z}, \tilde{\boldsymbol{y}} | \boldsymbol{\Omega}, \boldsymbol{\theta}) &\cong -\frac{1}{2}(\tilde{\boldsymbol{y}} - \boldsymbol{\Omega}\boldsymbol{\Phi}_{\boldsymbol{\theta}}\boldsymbol{z} - \boldsymbol{\Omega}\boldsymbol{\eta}_{\boldsymbol{\theta}})^\top \boldsymbol{\Omega}\boldsymbol{\Psi}_{\boldsymbol{\theta}}\boldsymbol{\Omega}^\top (\tilde{\boldsymbol{y}} - \boldsymbol{\Omega}\boldsymbol{\Phi}_{\boldsymbol{\theta}}\boldsymbol{z} - \boldsymbol{\Omega}\boldsymbol{\eta}_{\boldsymbol{\theta}}) + \frac{1}{2}\log\det(\boldsymbol{\Omega}\boldsymbol{\Psi}_{\boldsymbol{\theta}}\boldsymbol{\Omega}^\top) \\
&\quad - \frac{1}{2}(\boldsymbol{z} - \boldsymbol{\nu}_{\boldsymbol{\theta}})^\top \boldsymbol{\Gamma}_{\boldsymbol{\theta}}(\boldsymbol{z} - \boldsymbol{\nu}_{\boldsymbol{\theta}}) + \frac{1}{2}\log\det\boldsymbol{\Gamma}_{\boldsymbol{\theta}}.
\end{aligned} \tag{36}$$

We can then combine terms based on whether they are quadratic, linear, or constant functions of $\boldsymbol{z}$ to obtain the following simplified quadratic form:

$$\log p(\boldsymbol{z}, \tilde{\boldsymbol{y}} | \boldsymbol{\Omega}, \boldsymbol{\theta}) \cong -\frac{1}{2}\boldsymbol{z}^\top \mathbf{A}_{\boldsymbol{\theta}}\boldsymbol{z} + \boldsymbol{b}_{\boldsymbol{\theta}}^\top \boldsymbol{z} - c_{\boldsymbol{\theta}}, \tag{37}$$

where

$$\begin{aligned}
\mathbf{A}_{\boldsymbol{\theta}} &:= \boldsymbol{\Gamma}_{\boldsymbol{\theta}} + \boldsymbol{\Phi}_{\boldsymbol{\theta}}^\top \boldsymbol{\Omega}^\top \boldsymbol{\Omega}\boldsymbol{\Psi}_{\boldsymbol{\theta}}\boldsymbol{\Omega}^\top \boldsymbol{\Omega}\boldsymbol{\Phi}_{\boldsymbol{\theta}}, \\
\boldsymbol{b}_{\boldsymbol{\theta}} &:= \boldsymbol{\Gamma}_{\boldsymbol{\theta}}\boldsymbol{\nu}_{\boldsymbol{\theta}} + \boldsymbol{\Phi}_{\boldsymbol{\theta}}^\top \boldsymbol{\Omega}^\top \boldsymbol{\Omega}\boldsymbol{\Psi}_{\boldsymbol{\theta}}\boldsymbol{\Omega}^\top (\tilde{\boldsymbol{y}} - \boldsymbol{\Omega}\boldsymbol{\eta}_{\boldsymbol{\theta}}), \\
c_{\boldsymbol{\theta}} &:= \frac{1}{2}(\tilde{\boldsymbol{y}} - \boldsymbol{\Omega}\boldsymbol{\eta}_{\boldsymbol{\theta}})^\top \boldsymbol{\Omega}\boldsymbol{\Psi}_{\boldsymbol{\theta}}\boldsymbol{\Omega}^\top (\tilde{\boldsymbol{y}} - \boldsymbol{\Omega}\boldsymbol{\eta}_{\boldsymbol{\theta}}) + \frac{1}{2}\boldsymbol{\nu}_{\boldsymbol{\theta}}^\top \boldsymbol{\Gamma}_{\boldsymbol{\theta}}\boldsymbol{\nu}_{\boldsymbol{\theta}} - \frac{1}{2}\log\det\boldsymbol{\Omega}\boldsymbol{\Psi}_{\boldsymbol{\theta}}\boldsymbol{\Omega}^\top - \frac{1}{2}\log\det\boldsymbol{\Gamma}_{\boldsymbol{\theta}}.
\end{aligned} \tag{38}$$

By Gaussian prior-Gaussian likelihood conjugacy in (1), we know that the posterior $p(\boldsymbol{z}|\tilde{\boldsymbol{y}}, \boldsymbol{\Omega}, \boldsymbol{\theta})$ is also Gaussian with some mean $\boldsymbol{\mu}_{\boldsymbol{\theta}}$ and some covariance $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}$. From the derivations above, the log-pdf of this posterior (up to an additive constant) is given by (37). By matching this log-pdf to that of a $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}})$, we can conclude that $\boldsymbol{\Sigma}_{\boldsymbol{\theta}} = \mathbf{A}_{\boldsymbol{\theta}}^{-1}$ and $\boldsymbol{\mu}_{\boldsymbol{\theta}} = \mathbf{A}_{\boldsymbol{\theta}}^{-1}\boldsymbol{b}_{\boldsymbol{\theta}}$. In conclusion, we have

$$\begin{aligned}
q(\boldsymbol{\theta}_1 | \boldsymbol{\theta}_2) &= \mathbb{E}_{p(\boldsymbol{z}|\tilde{\boldsymbol{y}}, \boldsymbol{\Omega}, \boldsymbol{\theta}_2)}[-\log p(\boldsymbol{z}, \tilde{\boldsymbol{y}} | \boldsymbol{\Omega}, \boldsymbol{\theta}_1)] \cong \mathbb{E}_{\boldsymbol{z}\sim\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}_2}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}_2})}\left[\frac{1}{2}\boldsymbol{z}^\top \mathbf{A}_{\boldsymbol{\theta}_1}\boldsymbol{z} - \boldsymbol{b}_{\boldsymbol{\theta}_1}^\top \boldsymbol{z} + c_{\boldsymbol{\theta}_1}\right] \\
&= \frac{1}{2}\boldsymbol{\mu}_{\boldsymbol{\theta}}^\top \mathbf{A}_{\boldsymbol{\theta}'}\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{b}_{\boldsymbol{\theta}'}^\top \boldsymbol{\mu}_{\boldsymbol{\theta}} + c_{\boldsymbol{\theta}'} + \frac{1}{2}\mathrm{Tr}(\mathbf{A}_{\boldsymbol{\theta}'}\boldsymbol{\Sigma}_{\boldsymbol{\theta}}).
\end{aligned} \tag{39}$$

C. Derivation of Network Gradient Limit (17)

We provide a short derivation, showing that the limit of the network gradient (17) is the desired Monte Carlo gradient (10).

$$\begin{aligned}
&\widetilde{\boldsymbol{g}}^{\langle I\rangle}(\boldsymbol{\theta}) := \frac{\partial}{\partial\boldsymbol{\theta}}\left[q^{\langle I\rangle}(\boldsymbol{\theta}|\boldsymbol{\theta}) - \frac{1}{K}\sum_{k=1}^{K}\boldsymbol{\delta}_k^\top \boldsymbol{\sigma}_{k,\boldsymbol{\theta}}^{\langle I\rangle}\right] \\
&= \underbrace{\nabla_1 q^{\langle I\rangle}(\boldsymbol{\theta}|\boldsymbol{\theta})}_{\widehat{\boldsymbol{g}}^{\langle I\rangle}(\boldsymbol{\theta}) \text{ by (16)}} + \nabla_2 q^{\langle I\rangle}(\boldsymbol{\theta}|\boldsymbol{\theta}) - \frac{\partial}{\partial\boldsymbol{\theta}}\left[\frac{1}{K}\sum_{k=1}^{K}\boldsymbol{\delta}_k^\top \boldsymbol{\sigma}_{k,\boldsymbol{\theta}}^{\langle I\rangle}\right] \\
&= \widehat{\boldsymbol{g}}^{\langle I\rangle}(\boldsymbol{\theta}) + \frac{\partial\boldsymbol{\mu}_{\boldsymbol{\theta}}^{\langle I\rangle}}{\partial\boldsymbol{\theta}} \cdot \underbrace{\frac{\partial}{\partial\boldsymbol{\mu}_{\boldsymbol{\theta}}^{\langle I\rangle}}\left[\frac{1}{2}(\boldsymbol{\mu}_{\boldsymbol{\theta}}^{\langle I\rangle})^\top \mathbf{A}_{\boldsymbol{\theta}}\boldsymbol{\mu}_{\boldsymbol{\theta}}^{\langle I\rangle} - \boldsymbol{b}_{\boldsymbol{\theta}}^\top \boldsymbol{\mu}_{\boldsymbol{\theta}}^{\langle I\rangle}\right]}_{\text{converges to } \mathbf{0} \text{ as } I \to \infty} + \frac{1}{K}\sum_{k=1}^{K}\frac{\partial\boldsymbol{\sigma}_{k,\boldsymbol{\theta}}^{\langle I\rangle}}{\partial\boldsymbol{\theta}} \cdot \underbrace{\frac{\partial}{\partial\boldsymbol{\sigma}_{k,\boldsymbol{\theta}}^{\langle I\rangle}}\left[\frac{1}{2}(\boldsymbol{\sigma}_{k,\boldsymbol{\theta}}^{\langle I\rangle})^\top \mathbf{A}_{\boldsymbol{\theta}}\boldsymbol{\sigma}_{k,\boldsymbol{\theta}}^{\langle I\rangle} - \boldsymbol{\delta}_{k,\boldsymbol{\theta}}^\top \boldsymbol{\sigma}_{k,\boldsymbol{\theta}}^{\langle I\rangle}\right]}_{\text{converges to } \mathbf{0} \text{ as } I \to \infty}
\end{aligned} \tag{40}$$

The indicated derivatives converge to $\mathbf{0}$ because $\lim_{I\to\infty}\boldsymbol{\mu}_{\boldsymbol{\theta}}^{\langle I\rangle}, \lim_{I\to\infty}\boldsymbol{\sigma}_{k,\boldsymbol{\theta}}^{\langle I\rangle}$ minimize the respective functions in brackets. It follows that $\lim_{I\to\infty}\widetilde{\boldsymbol{g}}^{\langle I\rangle}(\boldsymbol{\theta}) = \lim_{I\to\infty}\widehat{\boldsymbol{g}}^{\langle I\rangle}(\boldsymbol{\theta}) = \boldsymbol{g}^{\#}(\boldsymbol{\theta})$.

D. Examples of Iterative Linear Solvers

In this appendix, we provide a few examples of iterative linear solvers for the system $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$; see Saad (2003) for other examples. The first example is gradient descent on the quadratic form $\frac{1}{2}\boldsymbol{x}^\top \mathbf{A}\boldsymbol{x} - \boldsymbol{b}^\top \boldsymbol{x}$, with step size parameter $\alpha \in \mathbb{R}$. The next example is steepest descent, which learns the optimal step size $\alpha^{\langle i \rangle}$ at each iteration i. Finally, we have conjugate gradient, which enforces orthogonality in the residuals $\boldsymbol{r}^{\langle i \rangle}$ and conjugacy (with respect to $\mathbf{A}$) in the search directions $\boldsymbol{d}^{\langle i \rangle}$.

Algorithm 2 GRADIENTDESCENT

```
$\boldsymbol{x}^{\langle 0 \rangle} \leftarrow \mathbf{0}$
for $i = 1, 2, \dots, I$ do
    $\boldsymbol{r}^{\langle i \rangle} \leftarrow \boldsymbol{b} - \mathbf{A}\boldsymbol{x}^{\langle i \rangle}$
    $\boldsymbol{x}^{\langle i \rangle} \leftarrow \boldsymbol{x}^{\langle i-1 \rangle} + \alpha \cdot \boldsymbol{r}^{\langle i \rangle}$
end for
```

Algorithm 3 STEEPESTDESCENT

```
$\boldsymbol{x}^{\langle 0 \rangle} \leftarrow \mathbf{0}$
for $i = 1, 2, \dots, I$ do
    $\boldsymbol{r}^{\langle i \rangle} \leftarrow \boldsymbol{b} - \mathbf{A}\boldsymbol{x}^{\langle i \rangle}$
    $\alpha^{\langle i \rangle} \leftarrow \dfrac{\langle \boldsymbol{r}^{\langle i \rangle}, \boldsymbol{r}^{\langle i \rangle} \rangle}{\langle \boldsymbol{r}^{\langle i \rangle}, \mathbf{A}\boldsymbol{r}^{\langle i \rangle} \rangle}$
    $\boldsymbol{x}^{\langle i \rangle} \leftarrow \boldsymbol{x}^{\langle i-1 \rangle} + \alpha^{\langle i \rangle} \cdot \boldsymbol{r}^{\langle i \rangle}$
end for
```

Algorithm 4 CONJUGATEGRADIENT

```
$\boldsymbol{x}^{\langle 0 \rangle} \leftarrow \mathbf{0}$
$\boldsymbol{r}^{\langle 0 \rangle} \leftarrow \boldsymbol{b} - \mathbf{A}\boldsymbol{x}^{\langle 0 \rangle}$
$\boldsymbol{d}^{\langle 0 \rangle} \leftarrow \boldsymbol{r}^{\langle 0 \rangle}$
for $i = 1, 2, \dots, I$ do
    $\alpha^{\langle i \rangle} \leftarrow \dfrac{\langle \boldsymbol{r}^{\langle i-1 \rangle}, \boldsymbol{r}^{\langle i-1 \rangle} \rangle}{\langle \boldsymbol{d}^{\langle i-1 \rangle}, \mathbf{A}\boldsymbol{d}^{\langle i-1 \rangle} \rangle}$
    $\boldsymbol{x}^{\langle i \rangle} \leftarrow \boldsymbol{x}^{\langle i-1 \rangle} + \alpha^{\langle i \rangle} \cdot \boldsymbol{d}^{\langle i-1 \rangle}$
    $\boldsymbol{r}^{\langle i \rangle} \leftarrow \boldsymbol{r}^{\langle i-1 \rangle} + \alpha^{\langle i \rangle} \cdot \mathbf{A}\boldsymbol{d}^{\langle i-1 \rangle}$
    $\beta^{\langle i \rangle} \leftarrow \dfrac{\langle \boldsymbol{r}^{\langle i \rangle}, \boldsymbol{r}^{\langle i \rangle} \rangle}{\langle \boldsymbol{r}^{\langle i-1 \rangle}, \boldsymbol{r}^{\langle i-1 \rangle} \rangle}$
    $\boldsymbol{d}^{\langle i \rangle} \leftarrow \boldsymbol{r}^{\langle i \rangle} + \beta^{\langle i \rangle} \cdot \boldsymbol{d}^{\langle i-1 \rangle}$
end for
```

E. Proofs for Section 5

E.1. Proposition 5.1

Proof. We will write $\boldsymbol{h}^\star - \boldsymbol{h}^\#$ as a short for $\boldsymbol{h}^\star(\boldsymbol{\theta}) - \boldsymbol{h}^\#(\boldsymbol{\theta})$. Recall from definitions that

$$h_\ell^\# - h_\ell^\star = \frac{1}{N}\sum_{n=1}^{N} g_\ell^{\#,(n)} - g_\ell^{\star,(n)} = \frac{1}{N}\sum_{n=1}^{N} [\nabla_1 q^{\#,(n)}(\boldsymbol{\theta}|\boldsymbol{\theta}) - \nabla_1 q^{(n)}(\boldsymbol{\theta}|\boldsymbol{\theta})]_\ell \tag{41}$$

where q is given by (5) and $q^\#$ is given by (9). Observe that the only difference between these two objectives $q, q^\#$ is the trace term in q versus the Monte Carlo approximation in $q^\#$. It therefore follows that for each n,

$$g_\ell^{\#,(n)} - g_\ell^{\star,(n)} = \left\langle \frac{\partial}{\partial \theta_\ell}\mathbf{A}_{\boldsymbol{\theta}}^{(n)}, \frac{1}{K}\sum_{k=1}^{K} \boldsymbol{x}_{k,\boldsymbol{\theta}}^{(n)}(\boldsymbol{x}_{k,\boldsymbol{\theta}}^{(n)})^\top - \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{(n)} \right\rangle, \tag{42}$$

where for two matrices $\mathbf{Q}, \mathbf{R}$ we define $\langle \mathbf{Q}, \mathbf{R} \rangle := \mathrm{Tr}(\mathbf{Q}\mathbf{R}^\top)$. Recall that for all k, each $\boldsymbol{x}_{k,\boldsymbol{\theta}}^{(n)} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{(n)})$. Thus, we can equivalently write $\boldsymbol{x}_{k,\boldsymbol{\theta}}^{(n)} = (\boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{(n)})^{1/2} \boldsymbol{\epsilon}_k^{(n)}$, where each $\boldsymbol{\epsilon}_k^{(n)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. It then follows that

$$g_\ell^{\#,(n)} - g_\ell^{\star,(n)} = \left\langle \frac{\partial}{\partial \theta_\ell} \mathbf{A}_{\boldsymbol{\theta}}^{(n)}, (\boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{(n)})^{1/2} \left(\frac{1}{K} \sum_{k=1}^{K} \boldsymbol{\epsilon}_k^{(n)} (\boldsymbol{\epsilon}_k^{(n)})^\top - \mathbf{I} \right) (\boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{(n)})^{1/2} \right\rangle \tag{43}$$

$$= \left\langle \underbrace{(\boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{(n)})^{1/2} \left(\frac{\partial}{\partial \theta_\ell} \mathbf{A}_{\boldsymbol{\theta}}^{(n)} \right) (\boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{(n)})^{1/2}}_{\mathbf{M}^{(n,\ell)}}, \frac{1}{K} \sum_{k=1}^{K} \boldsymbol{\epsilon}_k^{(n)} (\boldsymbol{\epsilon}_k^{(n)})^\top - \mathbf{I} \right\rangle . \tag{44}$$

Our proof strategy is now similar to that of the Hanson-Wright inequality (Theorem 6.2.1 in **?**).

Let $M_{d,d}^{(n,\ell)}$ denote the d-th diagonal element of the $D \times D$ matrix $\mathbf{M}^{(n,\ell)}$ and let $\widetilde{\mathbf{M}}^{(n,\ell)}$ denote the matrix $\mathbf{M}^{(n,\ell)}$ with its diagonal set to zero. Then,

$$h_\ell^{\#} - h_\ell^{\star} = \frac{1}{NK} \sum_{n=1}^{N} \sum_{k=1}^{K} \sum_{d=1}^{D} M_{d,d}^{(n,\ell)} \left((\epsilon_{k,d}^{(n)})^2 - 1 \right) + \frac{1}{NK} \sum_{n=1}^{N} \sum_{k=1}^{K} (\boldsymbol{\epsilon}_k^{(n)})^\top \widetilde{\mathbf{M}}^{(n,\ell)} \boldsymbol{\epsilon}_k^{(n)}. \tag{45}$$

We then write

$$\Pr\left(\left| NK \cdot (h_\ell^{\#} - h_\ell^{\star}) \right| > t \right) \leq \Pr\left(\left| \sum_{n=1}^{N} \sum_{k=1}^{K} \sum_{d=1}^{D} M_{d,d}^{(n,\ell)} \left((\epsilon_{k,d}^{(n)})^2 - 1 \right) \right| > \frac{t}{2} \right)$$

$$+ \Pr\left(\left| \sum_{n=1}^{N} \sum_{k=1}^{K} (\boldsymbol{\epsilon}_k^{(n)})^\top \widetilde{\mathbf{M}}^{(n,\ell)} \boldsymbol{\epsilon}_k^{(n)} \right| > \frac{t}{2} \right), \tag{46}$$

and bound each of these two terms separately.

(I) Diagonal Elements. We recall the definition of sub-exponential norm for a random variable X,

$$\|X\|_{\psi_1} := \inf \left\{ t > 0 : \mathbb{E}\left(\exp \frac{|X|}{t} \right) \leq 2 \right\}. \tag{47}$$

For $\epsilon_{k,d}^{(n)} \sim \mathcal{N}(0, 1)$, we have

$$\left\| M_{d,d}^{(n,\ell)} \left((\epsilon_{k,d}^{(n)})^2 - 1 \right) \right\|_{\psi_1} \leq \left| M_{d,d}^{(n,\ell)} \right| \left\| (\epsilon_{k,d}^{(n)})^2 - 1 \right\|_{\psi_1} \leq c_0 |M_{d,d}^{(n,\ell)}|, \tag{48}$$

for some absolute constant c_0. Then, by Bernstein's inequality, we have for all $t > 0$,

$$\Pr\left(\left| \sum_{n=1}^{N} \sum_{k=1}^{K} \sum_{d=1}^{D} M_{d,d}^{(n,\ell)} \left((\epsilon_{k,d}^{(n)})^2 - 1 \right) \right| > \frac{t}{2} \right) \leq 2 \exp\left(-c_1 \min\left(\frac{t^2}{K \sum_n \sum_{d=1}^{D} |M_{d,d}^{(n,\ell)}|^2}, \frac{t}{\max_n \max_d |M_{d,d}^{(n,\ell)}|} \right) \right)$$

$$\leq 2 \exp\left(-c_1 \min\left(\frac{t^2}{K \sum_n \|\mathbf{M}^{(n,\ell)}\|_F^2}, \frac{t}{\max_n \|\mathbf{M}^{(n,\ell)}\|_2} \right) \right), \tag{49}$$

for some absolute constant c_1, and where $\|\cdot\|_F$ denotes Frobenius norm and $\|\cdot\|_2$ denotes spectral norm.

(II) Off-diagonal Elements. We can bound one side of the off-diagonal term in (46) as follows: For all $\lambda \in \mathbb{R}$ and by Markov's inequality, we have

$$\Pr\left(\sum_{n=1}^{N} \sum_{k=1}^{K} (\boldsymbol{\epsilon}_k^{(n)})^\top \widetilde{\mathbf{M}}^{(n,\ell)} \boldsymbol{\epsilon}_k^{(n)} > \frac{t}{2} \right) \leq \left(\exp \frac{-\lambda t}{2} \right) \prod_{n=1}^{N} \prod_{k=1}^{K} \mathbb{E}\left[\exp\left(\lambda \cdot (\boldsymbol{\epsilon}_k^{(n)})^\top \widetilde{\mathbf{M}}^{(n,\ell)} \boldsymbol{\epsilon}_k^{(n)} \right) \right]. \tag{50}$$

We now apply decoupling (Theorem 6.1.1 in **?**) to have

$$\mathbb{E}\left[\exp\left(\lambda \cdot (\boldsymbol{\epsilon}_k^{(n)})^\top \widetilde{\mathbf{M}}^{(n,\ell)} \boldsymbol{\epsilon}_k^{(n)} \right) \right] \leq \mathbb{E}\left[\exp\left(4\lambda \cdot (\boldsymbol{\epsilon}_k^{(n)})^\top \widetilde{\mathbf{M}}^{(n,\ell)} \widetilde{\boldsymbol{\epsilon}}_k^{(n)} \right) \right], \tag{51}$$

where $\{\widetilde{\boldsymbol{\epsilon}}_k^{(n)}\}$ is an independent copy of $\{\boldsymbol{\epsilon}_k^{(n)}\}$. Then, by a known bound on the moment generating function of Gaussian chaos (Lemma 6.2.2 in **?**), we can find an absolute constant c_2, such that for $\lambda \max_n \|\widetilde{\mathbf{M}}^{(n,\ell)}\|_2 \leq c_2$,

$$\mathbb{E}\left[\exp\left(4\lambda \cdot (\boldsymbol{\epsilon}_k^{(n)})^\top \widetilde{\mathbf{M}}^{(n,\ell)} \widetilde{\boldsymbol{\epsilon}}_k^{(n)}\right)\right] \leq \exp\left(c_2 \lambda^2 \|\widetilde{\mathbf{M}}^{(n,\ell)}\|_F^2\right). \tag{52}$$

Then,

$$\Pr\left(\sum_{n=1}^N \sum_{k=1}^K (\boldsymbol{\epsilon}_k^{(n)})^\top \widetilde{\mathbf{M}}^{(n,\ell)} \boldsymbol{\epsilon}_k^{(n)} > \frac{t}{2}\right) \leq \inf_{\substack{\lambda > 0, \\ \lambda \max_n \|\widetilde{\mathbf{M}}^{(n,\ell)}\|_2 \leq c_2}} \exp\left(-\frac{\lambda t}{2} + c_2 K \lambda^2 \sum_{n=1}^N \|\widetilde{\mathbf{M}}^{(n,\ell)}\|_F^2\right). \tag{53}$$

We can then optimize over λ to get

$$\Pr\left(\sum_{n=1}^N \sum_{k=1}^K (\boldsymbol{\epsilon}_k^{(n)})^\top \widetilde{\mathbf{M}}^{(n,\ell)} \boldsymbol{\epsilon}_k^{(n)} > \frac{t}{2}\right) \leq \exp\left(-c_2 \min\left(\frac{t^2}{K \sum_{n=1}^N \|\mathbf{M}^{(n,\ell)}\|_F^2}, \frac{t}{\max_n \|\mathbf{M}^{(n,\ell)}\|_2}\right)\right). \tag{54}$$

Bounding both sides then yields

$$\Pr\left(\left|\sum_{n=1}^N \sum_{k=1}^K (\boldsymbol{\epsilon}_{k,d}^{(n)})^\top \widetilde{\mathbf{M}}^{(n,\ell)} \boldsymbol{\epsilon}_{k,d}^{(n)}\right| > \frac{t}{2}\right) \leq 2\exp\left(-c_2 \min\left(\frac{t^2}{K \sum_{n=1}^N \|\mathbf{M}^{(n,\ell)}\|_F^2}, \frac{t}{\max_n \|\mathbf{M}^{(n,\ell)}\|_2}\right)\right). \tag{55}$$

Returning to our original objective in (46), we can take $C := \min\{c_1, c_2\}$ and conclude

$$\Pr\left(\left|NK \cdot (h_\ell^\# - h_\ell^\star)\right| > t\right) \leq 4\exp\left(-C \min\left(\frac{t^2}{K \sum_{n=1}^N \|\mathbf{M}^{(n,\ell)}\|_F^2}, \frac{t}{\max_n \|\mathbf{M}^{(n,\ell)}\|_2}\right)\right). \tag{56}$$

Now, if we choose $t = \sqrt{t_0 (K/C) \sum_n \|\mathbf{M}^{(n,\ell)}\|_F^2}$ for some $t_0 > 0$, we have

$$\Pr\left(\left|NK \cdot (h_\ell^\# - h_\ell^\star)\right| > t\right) \leq 4\exp\left(-C \min\left(\frac{t_0}{C}, \sqrt{\frac{\sum_n \|\mathbf{M}^{(n,\ell)}\|_F^2}{\max_n \|\mathbf{M}^{(n,\ell)}\|_2^2} \frac{t_0 K}{C}}\right)\right) \leq 4e^{-t_0}, \tag{57}$$

as long as

$$\frac{t_0}{C} \leq \sqrt{\frac{\sum_n \|\mathbf{M}^{(n,\ell)}\|_F^2}{\max_n \|\mathbf{M}^{(n,\ell)}\|_2^2} \frac{t_0 K}{C}} \implies K \geq \frac{t_0}{C}\left(\frac{\max_n \|\mathbf{M}^{(n,\ell)}\|_2^2}{\sum_n \|\mathbf{M}^{(n,\ell)}\|_F^2}\right). \tag{58}$$

Since for all ℓ, $\sum_n \|\mathbf{M}^{(n,\ell)}\|_F^2 \leq N\xi^2$, we have

$$\text{for all } \ell: \quad \Pr\left(\left|NK \cdot (h_\ell^\# - h_\ell^\star)\right| > \xi \sqrt{\frac{t_0 NK}{C}}\right) \leq 4\exp(-t_0), \tag{59}$$

as long as

$$K \geq \frac{t_0}{C} \max_\ell \left(\frac{\max_n \|\mathbf{M}^{(n,\ell)}\|_2^2}{\sum_n \|\mathbf{M}^{(n,\ell)}\|_F^2}\right).$$

By union bound and taking $t_0 = \log(4NL)$, we can conclude that for some absolute constant C,

$$\Pr\left(\|\boldsymbol{h}^\star - \boldsymbol{h}^\#\|_\infty > \xi \sqrt{\frac{\log(4NL)}{CNK}}\right) \leq 4Le^{-t_0} \leq \frac{1}{N},$$

under the condition

$$K \geq \frac{\log(4NL)}{C} \max_\ell \left(\frac{\max_n \|\mathbf{M}^{(n,\ell)}\|_2^2}{\sum_n \|\mathbf{M}^{(n,\ell)}\|_F^2}\right). \tag{60}$$

Clearly, $\frac{\max_n \|\mathbf{M}^{(n,\ell)}\|_2^2}{\sum_n \|\mathbf{M}^{(n,\ell)}\|_F^2} \leq 1$. And if we can find $\kappa \geq 0$ such that for all ℓ, n, n',

$$\frac{\|\mathbf{M}^{(n,\ell)}\|_2^2}{\|\mathbf{M}^{(n',\ell)}\|_2^2} \geq \kappa,$$

then

$$\frac{\sum_n \|\mathbf{M}^{(n,\ell)}\|_F^2}{\max_n \|\mathbf{M}^{(n,\ell)}\|_2^2} = \sum_n \frac{\|\mathbf{M}^{(n,\ell)}\|_F^2}{\|\mathbf{M}^{(n,\ell)}\|_2^2} \frac{\|\mathbf{M}^{(n,\ell)}\|_2^2}{\max_n \|\mathbf{M}^{(n,\ell)}\|_2^2} \geq \kappa N.$$

Hence the condition in (60) is implied by the condition

$$K \geq \frac{\log(4NL)}{C \max(1, N\kappa)}.$$

□

E.2. Proposition 5.2

Proof. First, observe that

$$\|\widehat{\boldsymbol{h}}^{\langle I \rangle} - \boldsymbol{h}^{\#}\|_2 \leq \frac{1}{N} \sum_{n=1}^{N} \|\widehat{\boldsymbol{g}}^{\langle I \rangle,(n)} - \boldsymbol{g}^{\#,(n)}\|_2, \qquad \|\widetilde{\boldsymbol{h}}^{\langle I \rangle} - \boldsymbol{h}^{\#}\|_2 \leq \frac{1}{N} \sum_{n=1}^{N} \|\widetilde{\boldsymbol{g}}^{\langle I \rangle,(n)} - \boldsymbol{g}^{\#,(n)}\|_2. \tag{61}$$

Thus, if we show that for each n,

$$\|\widehat{\boldsymbol{g}}^{\langle I \rangle,(n)} - \boldsymbol{g}^{\#,(n)}\|_2 = \mathcal{O}(\rho^I), \qquad \|\widetilde{\boldsymbol{g}}^{\langle I \rangle,(n)} - \boldsymbol{g}^{\#,(n)}\|_2 = \mathcal{O}(I \cdot \rho^{2I}), \tag{62}$$

then the same convergence rates also hold for the population-level quantities $\|\widehat{\boldsymbol{h}}^{\langle I \rangle} - \boldsymbol{h}^{\#}\|_2$ and $\|\widetilde{\boldsymbol{h}}^{\langle I \rangle} - \boldsymbol{h}^{\#}\|_2$.

The rest of the proof is dedicated to proving (62). At a high level, we will reinterpret probabilistic unrolling as solving a bilevel optimization problem. We will then leverage new results that we prove for general bilevel optimization (i.e. Lemmas 5.3 and 5.4) to draw conclusions about the gradients' optimization error.

We define the bivariate function $r : \Theta \times \mathbb{R}^{(K+1)\times D} \to \mathbb{R}$,

$$r(\boldsymbol{\theta}, \{\boldsymbol{\mu}, \boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_K\}) := \frac{1}{2}\boldsymbol{\mu}^\top \mathbf{A}_{\boldsymbol{\theta}} \boldsymbol{\mu} - \boldsymbol{b}_{\boldsymbol{\theta}}^\top \boldsymbol{\mu} + \frac{1}{2K} \sum_{k=1}^{K} \boldsymbol{\sigma}_k^\top \mathbf{A}_{\boldsymbol{\theta}} \boldsymbol{\sigma}_k + c_{\boldsymbol{\theta}}. \tag{63}$$

We now re-express the gradients $\boldsymbol{g}^{\#}, \widehat{\boldsymbol{g}}^{\langle I \rangle}, \widetilde{\boldsymbol{g}}^{\langle I \rangle}$ in terms of r to analyze the relationships between them: For example, the Monte Carlo EM gradient (10) can be written as

$$\boldsymbol{g}^{\#}(\boldsymbol{\theta}) = \nabla_1 r(\boldsymbol{\theta}, \{\boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\sigma}_{1,\boldsymbol{\theta}}, \ldots, \boldsymbol{\sigma}_{K,\boldsymbol{\theta}}\}). \tag{64}$$

Here, $\nabla_1 r$ denotes the gradient of r with respect to its first argument $\boldsymbol{\theta}$ and

$$\begin{aligned} \boldsymbol{\mu}_{\boldsymbol{\theta}} &:= \mathbf{A}_{\boldsymbol{\theta}}^{-1} \boldsymbol{b}_{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\mu}} \left[s_0(\boldsymbol{\theta}, \boldsymbol{\mu}) := \tfrac{1}{2}\boldsymbol{\mu}^\top \mathbf{A}_{\boldsymbol{\theta}} \boldsymbol{\mu} - \boldsymbol{b}_{\boldsymbol{\theta}}^\top \boldsymbol{\mu} \right], \\ \boldsymbol{\sigma}_{k,\boldsymbol{\theta}} &:= \mathbf{A}_{\boldsymbol{\theta}}^{-1} \boldsymbol{u}_{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\sigma}_k} \left[s_k(\boldsymbol{\theta}, \boldsymbol{\sigma}_k) := \tfrac{1}{2}\boldsymbol{\sigma}_k^\top \mathbf{A}_{\boldsymbol{\theta}} \boldsymbol{\sigma}_k - \boldsymbol{\delta}_k^\top \boldsymbol{\sigma}_k \right], \forall k, \end{aligned} \tag{65}$$

are minimizers of inner problems $\{s_k\}_{k=0}^K$ parameterized by $\boldsymbol{\theta}$. Linear solvers, unrolled for I iterations, approximate the minimizers with $(\boldsymbol{\mu}_{\boldsymbol{\theta}}^{\langle I \rangle}, \boldsymbol{\sigma}_{k,\boldsymbol{\theta}}^{\langle I \rangle})$. Using r and $\{s_k\}_{k=0}^K$, we can write the output (16) and network (17) gradients as

$$\widehat{\boldsymbol{g}}^{\langle I \rangle}(\boldsymbol{\theta}) = \nabla_1 r(\boldsymbol{\theta}, \{\boldsymbol{\mu}_{\boldsymbol{\theta}}^{\langle I \rangle}, \boldsymbol{\sigma}_{1,\boldsymbol{\theta}}^{\langle I \rangle}, \ldots, \boldsymbol{\sigma}_{K,\boldsymbol{\theta}}^{\langle I \rangle}\}), \tag{66}$$

$$\widetilde{\boldsymbol{g}}^{\langle I \rangle}(\boldsymbol{\theta}) = \nabla_1 r(\boldsymbol{\theta}, \{\boldsymbol{\mu}_{\boldsymbol{\theta}}^{\langle I \rangle}, \boldsymbol{\sigma}_{1,\boldsymbol{\theta}}^{\langle I \rangle}, \ldots, \boldsymbol{\sigma}_{K,\boldsymbol{\theta}}^{\langle I \rangle}\}) + \frac{\partial \boldsymbol{\mu}_{\boldsymbol{\theta}}^{\langle I \rangle}}{\partial \boldsymbol{\theta}} \cdot \nabla_2 s_0(\boldsymbol{\theta}, \boldsymbol{\mu}_{\boldsymbol{\theta}}^{\langle I \rangle}) + \frac{1}{K} \sum_{k=1}^{K} \frac{\partial \boldsymbol{\sigma}_{k,\boldsymbol{\theta}}^{\langle I \rangle}}{\partial \boldsymbol{\theta}} \cdot \nabla_2 s_k(\boldsymbol{\theta}, \boldsymbol{\sigma}_{k,\boldsymbol{\theta}}^{\langle I \rangle}). \tag{67}$$

By Lemma 5.3, the output gradient (also called the "analytic gradient" by Ablin et al. (2020)) and the network gradient (also called the "automatic gradient" by Ablin et al. (2020)[3]) converge with rates

$$\|\boldsymbol{g}^{\#} - \widehat{\boldsymbol{g}}^{\langle I\rangle}\|_2 = \mathcal{O}(\|\boldsymbol{\beta}^{\#} - \boldsymbol{\beta}^{\langle I\rangle}\|_2), \qquad \|\boldsymbol{g}^{\#} - \widetilde{\boldsymbol{g}}^{\langle I\rangle}\|_2 = \mathcal{O}(\|\boldsymbol{J}^{\langle I\rangle} - \boldsymbol{J}^{\#}\|_2\|\boldsymbol{\beta}^{\langle I\rangle} - \boldsymbol{\beta}^{\#}\|_2). \tag{68}$$

where

$$\boldsymbol{\beta}^{\#} := \begin{bmatrix} \boldsymbol{\mu}_{\boldsymbol{\theta}} \\ \boldsymbol{\sigma}_{1,\boldsymbol{\theta}} \\ \vdots \\ \boldsymbol{\sigma}_{K,\boldsymbol{\theta}} \end{bmatrix}, \qquad \boldsymbol{\beta}^{\langle I\rangle} := \begin{bmatrix} \boldsymbol{\mu}_{\boldsymbol{\theta}}^{\langle I\rangle} \\ \boldsymbol{\sigma}_{1,\boldsymbol{\theta}}^{\langle I\rangle} \\ \vdots \\ \boldsymbol{\sigma}_{K,\boldsymbol{\theta}}^{\langle I\rangle} \end{bmatrix}, \qquad \boldsymbol{J}^{\#} := \frac{\partial \boldsymbol{\beta}^{\#}}{\partial \boldsymbol{\theta}}, \qquad \boldsymbol{J}^{\langle I\rangle} := \frac{\partial \boldsymbol{\beta}^{\langle I\rangle}}{\partial \boldsymbol{\theta}}. \tag{69}$$

Lemma 5.4 shows that for gradient descent (GD) and steepest descent (SD) as the linear solver,

$$\|\boldsymbol{\beta}^{\langle I\rangle} - \boldsymbol{\beta}^{\#}\| = \mathcal{O}(\rho^I), \tag{70}$$

$$\|\boldsymbol{J}^{\langle I\rangle} - \boldsymbol{J}^{\#}\| = \mathcal{O}(I \cdot \|\boldsymbol{\beta}^{\langle I\rangle} - \boldsymbol{\beta}^{\#}\|) = \mathcal{O}(I \cdot \rho^I), \tag{71}$$

where $\rho < 1$ is the solver's convergence rate. These rates are known as (Saad, 2003)

$$\rho_{\text{GD}} := \frac{\iota - 1}{\iota}, \qquad \rho_{\text{SD}} := \frac{\iota - 1}{\iota + 1}, \tag{72}$$

where ι is the condition number of $\mathbf{A}_{\boldsymbol{\theta}}$. □

E.3. Lemma 5.3

Proof. We denote the Lipschitz constant of the gradients $\nabla_1 r(\boldsymbol{\theta}, \boldsymbol{\beta})$ and $\nabla_2 s(\boldsymbol{\theta}, \boldsymbol{\beta})$ with respect to $\boldsymbol{\beta}$ by L_1^r and L_2^s, respectively. Similarly, we let the second derivatives $\nabla_{22}^2 s(\boldsymbol{\theta}, \boldsymbol{\beta})$ and $\nabla_{12}^2 r(\boldsymbol{\theta}, \boldsymbol{\beta})$ be L_{22}^s-Lipschitz and L_{12}^r-Lipschitz with respect to $\boldsymbol{\beta}$, respectively (note that for a function $f(\boldsymbol{a}_1, \boldsymbol{a}_2)$ of two variables $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$, the notation $\nabla_{ij}^2 f$ denotes the second derivative of f with respect to $\boldsymbol{a}_i$ and $\boldsymbol{a}_j$ for $i, j \in \{1, 2\}$).

Recall the target gradient is

$$\boldsymbol{g}^{\#} := \nabla_1 r(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#}). \tag{73}$$

The analytic gradient is defined as

$$\widehat{\boldsymbol{g}}^{\langle I\rangle} := \nabla_1 r(\boldsymbol{\theta}, \boldsymbol{\beta}^{\langle I\rangle}). \tag{74}$$

By Lipschitz continuity, the automatic gradient satisfies

$$\|\boldsymbol{g}^{\#} - \widehat{\boldsymbol{g}}^{\langle I\rangle}\|_2 = \|\nabla_1 r(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#}) - \nabla_1 r(\boldsymbol{\theta}, \boldsymbol{\beta}^{\langle I\rangle})\|_2 \leq L_1^r \|\boldsymbol{\beta}^{\#} - \boldsymbol{\beta}^{\langle I\rangle}\|_2. \tag{75}$$

Hence, the error of the analytic gradient is on the order of the approximation error of the optimizer, i.e.

$$\|\boldsymbol{g}^{\#} - \widehat{\boldsymbol{g}}^{\langle I\rangle}\|_2 = \mathcal{O}(\|\boldsymbol{\beta}^{\#} - \boldsymbol{\beta}^{\langle I\rangle}\|_2). \tag{76}$$

Next, we bound the error of the automatic gradient,

$$\widetilde{\boldsymbol{g}}^{\langle I\rangle} := \nabla_1 r(\boldsymbol{\theta}, \boldsymbol{\beta}^{\langle I\rangle}) + \boldsymbol{J}^{\langle I\rangle} \cdot \nabla_2 s(\boldsymbol{\theta}, \boldsymbol{\beta}^{\langle I\rangle}), \tag{77}$$

where $\boldsymbol{J}^{\langle I\rangle} := \frac{\partial \boldsymbol{\beta}^{\langle I\rangle}}{\partial \boldsymbol{\theta}}$, which is assumed to be bounded $\|\boldsymbol{J}^{\langle I\rangle}\|_2 \leq J_M$ for some constant J_M.

We begin by establishing some identities that will be useful for constructing our bound. From the inner problem, we have

$$\nabla_2 s(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#}) = \mathbf{0}. \tag{78}$$

[3] Note that the analytic and automatic gradients we define here are technically more general than those defined by Ablin et al. (2020), since Ablin et al. (2020) assume that the outer and inner optimization problems are the same (i.e. $s = r$).

In addition, by the implicit function theorem, the following also holds,

$$\boldsymbol{J}^{\#} := \frac{\partial \boldsymbol{\beta}^{\#}}{\partial \boldsymbol{\theta}} = -\nabla_{12}^2 s(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#}) \left[\nabla_{22}^2 s(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#})\right]^{-1} \implies \boldsymbol{J}^{\#} \cdot \nabla_{22}^2 s(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#}) + \nabla_{12}^2 s(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#}) = \mathbf{0}. \tag{79}$$

We have

$$\begin{aligned} \boldsymbol{g}^{\#} - \widetilde{\boldsymbol{g}}^{\langle I \rangle} = {} & \nabla_1 r(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#}) - \nabla_1 r(\boldsymbol{\theta}, \boldsymbol{\beta}^{\langle I \rangle}) - \boldsymbol{J}^{\langle I \rangle} \cdot \nabla_2 s(\boldsymbol{\theta}, \boldsymbol{\beta}^{\langle I \rangle}) && \\ & + (\nabla_{12}^2 r(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#}) - \nabla_{12}^2 r(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#}))(\boldsymbol{\beta}^{\#} - \boldsymbol{\beta}^{\langle I \rangle}) && = \mathbf{0} \\ & + (\boldsymbol{J}^{\langle I \rangle} \cdot \nabla_{22}^2 s(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#}) - \boldsymbol{J}^{\langle I \rangle} \cdot \nabla_{22}^2 s(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#}))(\boldsymbol{\beta}^{\#} - \boldsymbol{\beta}^{\langle I \rangle}) && = \mathbf{0} \\ & + \boldsymbol{J}^{\langle I \rangle} \cdot \nabla_2 s(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#}) && = \mathbf{0} \text{ by (78)} \\ & - (\boldsymbol{J}^{\#} \cdot \nabla_{22}^2 s(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#}) + \nabla_{12}^2 s(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#}))(\boldsymbol{\beta}^{\#} - \boldsymbol{\beta}^{\langle I \rangle}) && = \mathbf{0} \text{ by (79)} \end{aligned} \tag{80}$$

Rearranging terms, we have

$$\begin{aligned} \boldsymbol{g}^{\#} - \widetilde{\boldsymbol{g}}^{\langle I \rangle} = {} & (\nabla_1 r(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#}) - \nabla_1 r(\boldsymbol{\theta}, \boldsymbol{\beta}^{\langle I \rangle}) - \nabla_{12}^2 r(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#})(\boldsymbol{\beta}^{\#} - \boldsymbol{\beta}^{\langle I \rangle})) && \text{(Term A)} \\ & + \boldsymbol{J}^{\langle I \rangle} \left(\nabla_2 s(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#}) - \nabla_2 s(\boldsymbol{\theta}, \boldsymbol{\beta}^{\langle I \rangle}) - \nabla_{22}^2 s(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#})(\boldsymbol{\beta}^{\#} - \boldsymbol{\beta}^{\langle I \rangle})\right) && \text{(Term B)} \\ & + \left(\boldsymbol{J}^{\langle I \rangle} \nabla_{22}^2 s(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#}) + \nabla_{12}^2 r(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#}) - \boldsymbol{J}^{\#} \nabla_{22}^2 s(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#}) - \nabla_{12}^2 s(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#})\right)(\boldsymbol{\beta}^{\#} - \boldsymbol{\beta}^{\langle I \rangle}) && \text{(Term C)} \end{aligned}$$

We bound each term below. From Lipschitz-continuity of the second derivative $\nabla_{12} r$, we have the quadratic bound

$$\|\text{(Term A)}\|_2 = \|\nabla_1 r(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#}) - \nabla_1 r(\boldsymbol{\theta}, \boldsymbol{\beta}^{\langle I \rangle}) - \nabla_{12}^2 r(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#})(\boldsymbol{\beta}^{\#} - \boldsymbol{\beta}^{\langle I \rangle})\|_2 \leq \frac{L_{12}^r}{2} \|\boldsymbol{\beta}^{\#} - \boldsymbol{\beta}^{\langle I \rangle}\|_2^2. \tag{81}$$

Similarly, from Lipschitz-continuity of the second derivative $\nabla_{22} s$, we have another quadratic bound

$$\|\text{(Term B)}\|_2 = \|\boldsymbol{J}^{\langle I \rangle} \left(\nabla_2 s(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#}) - \nabla_2 s(\boldsymbol{\theta}, \boldsymbol{\beta}^{\langle I \rangle}) - \nabla_{22}^2 s(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#})(\boldsymbol{\beta}^{\#} - \boldsymbol{\beta}^{\langle I \rangle})\right)\|_2 \leq J_M \cdot \frac{L_{22}^s}{2} \|\boldsymbol{\beta}^{\#} - \boldsymbol{\beta}^{\langle I \rangle}\|_2^2. \tag{82}$$

Finally,

$$\begin{aligned} \|\text{(Term C)}\|_2 &= \|\left(\boldsymbol{J}^{\langle I \rangle} \nabla_{22}^2 s(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#}) + \nabla_{12}^2 r(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#}) - \boldsymbol{J}^{\#} \nabla_{22}^2 s(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#}) - \nabla_{12}^2 s(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#})\right)(\boldsymbol{\beta}^{\#} - \boldsymbol{\beta}^{\langle I \rangle})\|_2 \\ &= \|((\boldsymbol{J}^{\langle I \rangle} - \boldsymbol{J}^{\#}) \nabla_{22}^2 s(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#}) + (\nabla_{12}^2 r(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#}) - \nabla_{12}^2 s(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#})))(\boldsymbol{\beta}^{\#} - \boldsymbol{\beta}^{\langle I \rangle})\|_2 \\ &\leq L_2^s \|\boldsymbol{J}^{\langle I \rangle} - \boldsymbol{J}^{\#}\|_2 \|\boldsymbol{\beta}^{\#} - \boldsymbol{\beta}^{\langle I \rangle}\|_2 + \|\nabla_{12}^2 r(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#}) - \nabla_{12}^2 s(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#})\|_2 \|\boldsymbol{\beta}^{\#} - \boldsymbol{\beta}^{\langle I \rangle}\|_2 \end{aligned} \tag{83}$$

Hence, under the sufficient condition of s and r sharing the same second order derivatives, i.e. $\nabla_{12}^2 r(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#}) = \nabla_{12}^2 s(\boldsymbol{\theta}, \boldsymbol{\beta}^{\#})$, the automatic gradient converges as

$$\|\boldsymbol{g}^{\#} - \widetilde{\boldsymbol{g}}^{\langle I \rangle}\|_2 = \mathcal{O}(\|\boldsymbol{J}^{\langle I \rangle} - \boldsymbol{J}^{\#}\|_2 \|\boldsymbol{\beta}^{\langle I \rangle} - \boldsymbol{\beta}^{\#}\|_2). \tag{84}$$

□

E.4. Lemma 5.4

This section proves the convergence rate of the Jacobian $\boldsymbol{J}^{\langle I \rangle}$ to $\boldsymbol{J}^{\#}$ for gradient descent and steepest descent. See Appendix D for a summary of these algorithms.

E.4.1. Solver Convergence

For completeness, we begin by proving the convergence rate of the linear solver (i.e. how fast $\boldsymbol{\beta}^{\langle I \rangle}$ converges to $\boldsymbol{\beta}^{\langle \# \rangle}$).

Gradient Descent For gradient descent, we have

$$\begin{aligned}
\boldsymbol{\beta}^{\langle I+1\rangle} &= \boldsymbol{\beta}^{\langle I\rangle} - \alpha(\mathbf{A}_{\boldsymbol{\theta}}\boldsymbol{\beta}^{\langle I\rangle} - \boldsymbol{u}_{\boldsymbol{\theta}}) \\
&= \boldsymbol{\beta}^{\langle I\rangle} - \alpha(\mathbf{A}_{\boldsymbol{\theta}}\boldsymbol{\beta}^{\langle I\rangle} - \mathbf{A}_{\boldsymbol{\theta}}\boldsymbol{\beta}^{\#}) \\
\boldsymbol{\beta}^{\langle I+1\rangle} - \boldsymbol{\beta}^{\#} &= (\mathbf{I} - \alpha\mathbf{A}_{\boldsymbol{\theta}})(\boldsymbol{\beta}^{\langle I\rangle} - \boldsymbol{\beta}^{\#}) \\
\|\boldsymbol{\beta}^{\langle I+1\rangle} - \boldsymbol{\beta}^{\#}\| &= \|(\mathbf{I} - \alpha\mathbf{A}_{\boldsymbol{\theta}})(\boldsymbol{\beta}^{\langle I\rangle} - \boldsymbol{\beta}^{\#})\|_2 \\
&\leq \|\mathbf{I} - \alpha\mathbf{A}_{\boldsymbol{\theta}}\|_2\|\boldsymbol{\beta}^{\langle I\rangle} - \boldsymbol{\beta}^{\#}\|_2 \\
&\leq \rho_{\text{GD}}\|\boldsymbol{\beta}^{\langle I\rangle} - \boldsymbol{\beta}^{\#}\|_2,
\end{aligned} \tag{85}$$

where $\alpha \in \mathbb{R}$ is the constant step size of gradient descent. To ensure convergence, we require $\alpha \leq \frac{1}{\lambda_{\max}(\mathbf{A}_{\boldsymbol{\theta}})}$, where $\lambda_{\max}(\mathbf{A}_{\boldsymbol{\theta}})$ is the largest eigenvalue of matrix $\mathbf{A}_{\theta}$. Assuming $\alpha = \frac{1}{\lambda_{\max}(\mathbf{A}_{\boldsymbol{\theta}})}$, the spectral norm ρ_{GD} (or largest eigenvalue) of the symmetric positive definite matrix $\mathbf{I} - \alpha\mathbf{A}_{\boldsymbol{\theta}}$ is

$$\rho_{\text{GD}} = 1 - \frac{\lambda_{\min}(\mathbf{A}_{\boldsymbol{\theta}})}{\lambda_{\max}(\mathbf{A}_{\boldsymbol{\theta}})} = \frac{\iota - 1}{\iota}, \tag{86}$$

where $\iota := \frac{\lambda_{\max}(\mathbf{A}_{\boldsymbol{\theta}})}{\lambda_{\min}(\mathbf{A}_{\boldsymbol{\theta}})}$ is defined as the condition number of $\mathbf{A}_{\theta}$. This leads to the following rate of convergence:

$$\|\boldsymbol{\beta}^{\langle I\rangle} - \boldsymbol{\beta}^{\#}\|_2 = \mathcal{O}(\rho_{\text{GD}}^{I}). \tag{87}$$

Steepest Descent For steepest descent, let $\{\boldsymbol{v}_j\}_j$ be the set of eigenvectors of $\mathbf{A}_{\boldsymbol{\theta}}$, with $\|\boldsymbol{v}_j\|_2 = 1$ and corresponding eigenvalues of $\lambda_1 > \lambda_2 > \cdots > \lambda_p$. We define the error at iteration I by $\boldsymbol{e}^{\langle I\rangle} := \boldsymbol{\beta}^{\langle I\rangle} - \boldsymbol{\beta}^{\#}$. We express this error as a linear combination of the eigenvectors,

$$\boldsymbol{e}^{\langle I\rangle} = \sum_j \zeta_j^{\langle I\rangle}\boldsymbol{v}_j \tag{88}$$

for some coefficients $\{\boldsymbol{\zeta}_j^{\langle I\rangle}\}_{j=1}^{D}$. Now, we define the following residual

$$\boldsymbol{r}^{\langle I\rangle} = \boldsymbol{u}_{\boldsymbol{\theta}} - \mathbf{A}_{\boldsymbol{\theta}}\boldsymbol{\beta}^{\langle I\rangle} = -\mathbf{A}_{\boldsymbol{\theta}}\boldsymbol{e}^{\langle I\rangle} = -\sum_j \zeta_j^{\langle I\rangle}\lambda_j\boldsymbol{v}_j. \tag{89}$$

This gives us $\|\boldsymbol{r}^{\langle I\rangle}\|_2^2 = \sum_j \zeta_j^{\langle I\rangle 2}\lambda_j^2$ and $\boldsymbol{r}^{\langle I\rangle\top}\mathbf{A}_{\boldsymbol{\theta}}\boldsymbol{r}^{\langle I\rangle} = \sum_j \zeta_j^{\langle I\rangle 2}\lambda_j^3$. Hence, we can express the optimal step-size $\alpha^{\langle I\rangle}$ as

$$\alpha^{\langle I\rangle} = \frac{\sum_j \zeta_j^{\langle I\rangle 2}\lambda_j^2}{\sum_j \zeta_j^{\langle I\rangle 2}\lambda_j^3}. \tag{90}$$

Now, we show that the updates of steepest descent are contractive,

$$\begin{aligned}
\|\boldsymbol{\beta}^{\langle I+1\rangle} - \boldsymbol{\beta}^{\#}\|_2 &= \|(\mathbf{I} - \alpha^{\langle I\rangle}\mathbf{A}_{\boldsymbol{\theta}})(\boldsymbol{\beta}^{\langle I\rangle} - \boldsymbol{\beta}^{\#})\|_2 \\
&= \|(\mathbf{I} - \frac{\sum_j \zeta_j^{\langle I\rangle 2}\lambda_j^2}{\sum_j \zeta_j^{\langle I\rangle 2}\lambda_j^3}\mathbf{A}_{\boldsymbol{\theta}})\sum_b \zeta_b^{\langle I\rangle}\boldsymbol{v}_b\|_2 \\
&= \|\sum_b(1 - \frac{\sum_j \zeta_j^{\langle I\rangle 2}\lambda_j^2}{\sum_j \zeta_j^{\langle I\rangle 2}\lambda_j^3}\lambda_b)\zeta_b^{\langle I\rangle}\boldsymbol{v}_b\|_2 \\
&= \|\sum_b(1 - \frac{\sum_j \zeta_j^{\langle I\rangle 2}\lambda_j^2}{\sum_j \zeta_j^{\langle I\rangle 2}\lambda_j^3}\lambda_b)\zeta_b^{\langle I\rangle}\boldsymbol{v}_b\|_2 \\
&= \omega^{\langle I\rangle}\|\boldsymbol{\beta}^{\langle I\rangle} - \boldsymbol{\beta}^{\#}\|_2,
\end{aligned} \tag{91}$$

where

$$\omega^{\langle I\rangle} = \frac{\|\sum_b(1 - \frac{\sum_j \zeta_j^{\langle I\rangle 2}\lambda_j^2}{\sum_j \zeta_j^{\langle I\rangle 2}\lambda_j^3}\lambda_b)\zeta_b^{\langle I\rangle}\boldsymbol{v}_b\|_2}{\|\sum_a \zeta_a^{\langle I\rangle}\boldsymbol{v}_a\|_2}. \tag{92}$$

We are now left to find an upper bound on $\omega^{\langle I\rangle}$ that goes to zero as I increases. We derive results for $D = 2$. The condition number of $\mathbf{A}_{\boldsymbol{\theta}}$ is $\iota := \frac{\lambda_1}{\lambda_2} > 1$. We denote $\tau^{\langle I\rangle} :== \frac{\zeta_2^{\langle I\rangle}}{\zeta_1^{\langle I\rangle}}$. We have

$$\begin{aligned}
\omega^{\langle I\rangle} &= \frac{\|\sum_b (1 - \frac{\sum_j \zeta_j^{\langle I\rangle 2}\lambda_j^2}{\sum_j \zeta_j^{\langle I\rangle 2}\lambda_j^3}\lambda_b)\zeta_b^{\langle I\rangle}\boldsymbol{v}_b\|_2}{\|\sum_a \zeta_a^{\langle I\rangle}\boldsymbol{v}_a\|_2} \\
&= \frac{\|\sum_b (1 - \frac{\zeta_1^{\langle I\rangle 2}\lambda_1^2 + \zeta_2^{\langle I\rangle 2}\lambda_2^2}{\zeta_1^{\langle I\rangle 2}\lambda_1^3 + \zeta_2^{\langle I\rangle 2}\lambda_2^3}\lambda_b)\zeta_b^{\langle I\rangle}\boldsymbol{v}_b\|_2}{\|\zeta_1^{\langle I\rangle}\boldsymbol{v}_1 + \zeta_2^{\langle I\rangle}\boldsymbol{v}_2\|_2} \\
&= \frac{\|\sum_b (1 - \frac{\iota^2 + \tau^{\langle I\rangle 2}}{\lambda_2(\iota^3 + \tau^{\langle I\rangle 2})}\lambda_b)\zeta_b^{\langle I\rangle}\boldsymbol{v}_b\|_2}{\|\zeta_1^{\langle I\rangle}\boldsymbol{v}_1 + \zeta_2^{\langle I\rangle}\boldsymbol{v}_2\|_2} \\
&= \frac{\|(1 - \frac{\iota^2 + \tau^{\langle I\rangle 2}}{\lambda_2(\iota^3 + \tau^{\langle I\rangle 2})}\lambda_1)\zeta_1^{\langle I\rangle}\boldsymbol{v}_1 + (1 - \frac{\iota^2 + \tau^{\langle I\rangle 2}}{\lambda_2(\iota^3 + \tau^{\langle I\rangle 2})}\lambda_2)\zeta_2^{\langle I\rangle}\boldsymbol{v}_2\|_2}{\|\zeta_1^{\langle I\rangle}\boldsymbol{v}_1 + \zeta_2^{\langle I\rangle}\boldsymbol{v}_2\|_2} \\
&= \frac{\|(\zeta_1^{\langle I\rangle}\boldsymbol{v}_1 + \zeta_2^{\langle I\rangle}\boldsymbol{v}_2) - (\iota\frac{\iota^2 + \tau^{\langle I\rangle 2}}{(\iota^3 + \tau^{\langle I\rangle 2})})\zeta_1^{\langle I\rangle}\boldsymbol{v}_1 - (\frac{\iota^2 + \tau^{\langle I\rangle 2}}{(\iota^3 + \tau^{\langle I\rangle 2})})\zeta_2^{\langle I\rangle}\boldsymbol{v}_2\|_2}{\|\zeta_1^{\langle I\rangle}\boldsymbol{v}_1 + \zeta_{i2}\boldsymbol{v}_2\|_2} \\
&= \frac{\|(\zeta_1^{\langle I\rangle}\boldsymbol{v}_1 + \zeta_2^{\langle I\rangle}\boldsymbol{v}_2) - \frac{\iota^2 + \tau^{\langle I\rangle 2}}{(\iota^3 + \tau^{\langle I\rangle 2})}(\iota\zeta_1^{\langle I\rangle}\boldsymbol{v}_1 + \zeta_2^{\langle I\rangle}\boldsymbol{v}_2)\|_2}{\|\zeta_1^{\langle I\rangle}\boldsymbol{v}_1 + \zeta_2^{\langle I\rangle}\boldsymbol{v}_2\|_2} \\
&= \frac{\|(\boldsymbol{v}_1 + \tau^{\langle I\rangle}\boldsymbol{v}_2) - \frac{\iota^2 + \tau^{\langle I\rangle 2}}{(\iota^3 + \tau^{\langle I\rangle 2})}(\iota\boldsymbol{v}_1 + \tau^{\langle I\rangle}\boldsymbol{v}_2)\|_2}{\|\boldsymbol{v}_1 + \tau^{\langle I\rangle}\boldsymbol{v}_2\|_2}.
\end{aligned} \tag{93}$$

The worst convergence (i.e. an upper bound) is achieved when $\iota = \tau^{\langle I\rangle}$. Hence, we write the upper bound on $\omega^{\langle I\rangle}$ as follows:

$$\begin{aligned}
\omega^{\langle I\rangle} &= \frac{\|(\boldsymbol{v}_1 + \tau^{\langle I\rangle}\boldsymbol{v}_2) - \frac{\iota^2 + \tau^{\langle I\rangle 2}}{(\iota^3 + \tau^{\langle I\rangle 2})}(\iota\boldsymbol{v}_1 + \tau^{\langle I\rangle}\boldsymbol{v}_2)\|_2}{\|\boldsymbol{v}_1 + \tau^{\langle I\rangle}\boldsymbol{v}_2\|_2} \\
&\leq \frac{\|(\boldsymbol{v}_1 + \iota\boldsymbol{v}_2) - \frac{\iota^2 + \iota^2}{(\iota^3 + \iota^2)}(\iota\boldsymbol{v}_1 + \iota\boldsymbol{v}_2)\|_2}{\|\boldsymbol{v}_1 + \iota\boldsymbol{v}_2\|_2} \\
&= \frac{\|(\boldsymbol{v}_1 + \iota\boldsymbol{v}_2) - \frac{2\iota^2}{\iota^2(1+\iota)}(\iota\boldsymbol{v}_1 + \iota\boldsymbol{v}_2)\|_2}{\|\boldsymbol{v}_1 + \iota\boldsymbol{v}_2\|_2} \\
&= \frac{\|(\boldsymbol{v}_1 + \iota\boldsymbol{v}_2) - \frac{2\iota}{1+\iota}(\boldsymbol{v}_1 + \boldsymbol{v}_2)\|_2}{\|\boldsymbol{v}_1 + \iota\boldsymbol{v}_2\|_2} \\
&= \frac{\|\frac{1-\iota}{\iota+1}\boldsymbol{v}_1 + \frac{\iota(\iota-1)}{\iota+1}\boldsymbol{v}_2\|_2}{\|\boldsymbol{v}_1 + \iota\boldsymbol{v}_2\|_2} \\
&= \frac{\frac{\iota-1}{\iota+1}\|\boldsymbol{v}_1 + \iota\boldsymbol{v}_2\|_2}{\|\boldsymbol{v}_1 + \iota\boldsymbol{v}_2\|_2} \\
&= \frac{\iota - 1}{\iota + 1}.
\end{aligned} \tag{94}$$

Hence,

$$\|\boldsymbol{\beta}^{\langle I+1\rangle} - \boldsymbol{\beta}^{\#}\|_2 \leq \left(\frac{\iota - 1}{\iota + 1}\right)\|\boldsymbol{\beta}^{\langle I\rangle} - \boldsymbol{\beta}^{\#}\|_2. \tag{95}$$

We denote $\rho_{\text{SD}} := \frac{\iota-1}{\iota+1}$ (which is a faster rate than ρ_{GD}), and write

$$\|\boldsymbol{\beta}^{\langle I\rangle} - \boldsymbol{\beta}^{\#}\|_2 = \mathcal{O}(\rho_{\text{SD}}^I). \tag{96}$$

This bound also holds for $D > 2$; the proof steps are similar (e.g. see Section 9.2 of Shewchuk et al. (1994)).

E.4.2. Jacobian Convergence

We now analyze the convergence rate of the Jacobian. Convergence studies in this context date back to the seminal work of **?**. We drop the subscript $\boldsymbol{\theta}$ when referring to (j,k)-entry of $\mathbf{A}_{\boldsymbol{\theta}}$ for ease of notation. Given matrix $\mathbf{A}$, we have

$$\partial\lambda_j = \boldsymbol{v}_j^\top \partial\mathbf{A}_{\boldsymbol{\theta}}\boldsymbol{v}_j \tag{97}$$

$$\partial\boldsymbol{v}_j = (\lambda_j\mathbf{I} - \mathbf{A}_{\boldsymbol{\theta}})^\dagger \partial\mathbf{A}_{\boldsymbol{\theta}}\boldsymbol{v}_j \tag{98}$$

$$\frac{\partial\mathbf{A}_{\boldsymbol{\theta}}}{\partial\mathbf{A}_{jk}} = \mathbf{H}^{jk}, \tag{99}$$

where † denotes pseudo inverse and $\mathbf{H}$ is a zero matrix except at the (j,k)-entry, which is 1. The above holds when the eigenvalues and vectors are distinct. We denote the Jacobian error by $\mathbf{B}^{\langle I\rangle} := \boldsymbol{J}^{\langle I\rangle} - \boldsymbol{J}^{\#} = \frac{\partial\boldsymbol{\beta}^{\langle I\rangle}}{\partial\boldsymbol{\theta}} - \frac{\partial\boldsymbol{\beta}^{\#}}{\partial\boldsymbol{\theta}}$. Given the expression $\boldsymbol{\beta}^{\langle I\rangle} - \boldsymbol{\beta}^{\#} = \sum_b \zeta_b^{\langle I\rangle}\boldsymbol{v}_b$, we first focus on writing the Jacobian error with respect to $\mathbf{A}_{jk}$, denoted by $\mathbf{B}_{\mathbf{A}_{jk}}^{\langle I\rangle} = \frac{\partial\boldsymbol{e}^{\langle I\rangle}}{\partial\mathbf{A}_{jk}}$. Then, we use this Jacobian error to find the bound on the Jacobian with respect to $\boldsymbol{\theta}$ as $\mathbf{B}^{\langle I\rangle} = \sum_{jk}\frac{\partial\mathbf{A}_{jk}}{\partial\boldsymbol{\theta}}\frac{\partial\boldsymbol{e}^{\langle I\rangle}}{\partial\mathbf{A}_{jk}}$.

Gradient Descent For gradient descent, we start with the recursion below,

$$\boldsymbol{\beta}^{\langle I+1\rangle} - \boldsymbol{\beta}^{\#} = (\mathbf{I} - \alpha\mathbf{A}_{\boldsymbol{\theta}})(\boldsymbol{\beta}^{\langle I\rangle} - \boldsymbol{\beta}^{\#}). \tag{100}$$

Then, we take the derivative with respect to $\mathbf{A}_{jk}$,

$$\begin{aligned}\mathbf{B}_{\mathbf{A}_{jk}}^{\langle I+1\rangle} &= (\mathbf{I} - \alpha\mathbf{A}_{\boldsymbol{\theta}})\mathbf{B}_{\mathbf{A}_{jk}}^{\langle I\rangle} + \frac{\partial(\mathbf{I} - \alpha\mathbf{A}_{\boldsymbol{\theta}})}{\partial\mathbf{A}_{jk}}\boldsymbol{e}^{\langle I\rangle} \\ &= (\mathbf{I} - \alpha\mathbf{A}_{\boldsymbol{\theta}})\mathbf{B}_{\mathbf{A}_{jk}}^{\langle I\rangle} - \alpha\mathbf{H}^{jk}\boldsymbol{e}^{\langle I\rangle}.\end{aligned} \tag{101}$$

We use the eigendecomposition of $\mathbf{A}_{\boldsymbol{\theta}} = \mathbf{V}\boldsymbol{\Sigma}\mathbf{V}^\top$ (i.e. $\mathbf{V}$ is the eigenvectors of $\mathbf{A}_{\boldsymbol{\theta}}$, and $\boldsymbol{\Sigma}$ is a diagonal matrix of $\mathbf{A}$ eigenvalues.This leads to the bound

$$\begin{aligned}\|\mathbf{B}_{\mathbf{A}_{jk}}^{\langle I+1\rangle}\|_2 &\le \|\mathbf{I} - \alpha\mathbf{A}_{\boldsymbol{\theta}}\|_2\|\mathbf{B}_{\mathbf{A}_{jk}}^{\langle I\rangle}\|_2 + \|\alpha\mathbf{H}^{jk}\boldsymbol{e}^{\langle I\rangle}\|_2 \\ &\le \rho_{\text{GD}}\|\mathbf{B}_{\mathbf{A}_{jk}}^{\langle I\rangle}\|_2 + \|\alpha\mathbf{H}^{jk}\|_2\|\boldsymbol{e}^{\langle I\rangle}\|_2.\end{aligned} \tag{102}$$

Unrolling the recursion gives us

$$\|\mathbf{B}_{\mathbf{A}_{jk}}^{\langle I\rangle}\|_2 = \mathcal{O}(I\rho_{\text{GD}}^I). \tag{103}$$

Given $\mathbf{B}^{\langle I\rangle} = \sum_{jk}\frac{\partial\mathbf{A}_{jk}}{\partial\boldsymbol{\theta}}\mathbf{B}_{\mathbf{A}_{jk}}^{\langle I\rangle}$, for gradient descent, we have

$$\|\mathbf{B}^{\langle I\rangle}\|_2 = \mathcal{O}(I\rho_{\text{GD}}^I). \tag{104}$$

Steepest Descent For steepest descent, we analyze Jacobian convergence under two different scenarios: (a) the gradient with respect to $\boldsymbol{\theta}$ is *not* propagated through the adaptive step size $\alpha^{\langle I\rangle}$, and (b) the gradient is propagated through $\alpha^{\langle I\rangle}$ (i.e. a more sophisticated scenario). In both of these scenarios, we show that the Jacobian converges at the same asymptotic rate. We express the Jacobian error as

$$\begin{aligned}\mathbf{B}_{\mathbf{A}_{jk}}^{\langle I\rangle} &= \sum_b \zeta_b^{\langle I\rangle}\frac{\partial\boldsymbol{v}_b}{\partial\mathbf{A}_{jk}} \\ &= \sum_b \zeta_b^{\langle I\rangle}(\lambda_b\boldsymbol{I} - \mathbf{A}_{\boldsymbol{\theta}})^\dagger\mathbf{H}^{jk}\boldsymbol{v}_b \\ &= \sum_b \zeta_b^{\langle I\rangle}(\lambda_b\boldsymbol{I} - \mathbf{V}\boldsymbol{\Sigma}\mathbf{V}^\top)^\dagger\mathbf{H}^{jk}\boldsymbol{v}_b \\ &= \sum_b \zeta_b^{\langle I\rangle}(\mathbf{V}\boldsymbol{\Lambda}_b\mathbf{V}^\top)\mathbf{H}^{jk}\boldsymbol{v}_b \\ &= \sum_b\sum_d \zeta_b^{\langle I\rangle}(q_{bd}\boldsymbol{v}_{jd}\boldsymbol{v}_{bk})\boldsymbol{v}_d,\end{aligned} \tag{105}$$

where $\mathbf{\Lambda}_b$ is a diagonal matrix with $q_{bd} := \frac{1}{\lambda_b - \lambda_d}$ is its d-th diagonal entry, and 0 on b-th diagonal entry. We substitute this expression into the Jacobian recursion,

$$\mathbf{B}_{\partial \mathbf{A}_{jk}}^{\langle I+1\rangle} = (\mathbf{I} - \alpha^{\langle I\rangle}\mathbf{A}_{\boldsymbol{\theta}})\mathbf{B}_{\mathbf{A}_{jk}}^{\langle I\rangle} + \frac{\partial(\mathbf{I} - \alpha^{\langle I\rangle}\mathbf{A}_{\boldsymbol{\theta}})}{\partial \mathbf{A}_{jk}}\boldsymbol{e}^{\langle I\rangle}. \tag{106}$$

Scenario (a).

$$\begin{aligned}
\mathbf{B}_{\mathbf{A}_{jk}}^{\langle I+1\rangle} &= (\mathbf{I} - \alpha^{\langle I\rangle}\mathbf{A}_{\boldsymbol{\theta}})\mathbf{B}_{\mathbf{A}_{jk}}^{\langle I\rangle} - \alpha^{\langle I\rangle}\mathbf{H}^{jk}\boldsymbol{e}^{\langle I\rangle} \\
&= (\mathbf{I} - \alpha^{\langle I\rangle}\mathbf{A}_{\boldsymbol{\theta}})\sum_b\sum_d \zeta_b^{\langle I\rangle}(q_{bd}\boldsymbol{v}_{jd}\boldsymbol{v}_{bk})\boldsymbol{v}_d - \alpha^{\langle I\rangle}\mathbf{H}^{jk}\boldsymbol{e}^{\langle I\rangle} \\
&= \sum_b\sum_d \zeta_b^{\langle I\rangle}(q_{bd}\boldsymbol{v}_{jd}\boldsymbol{v}_{bk})(\mathbf{I} - \alpha^{\langle I\rangle}\mathbf{A}_{\boldsymbol{\theta}})\boldsymbol{v}_d - \alpha^{\langle I\rangle}\mathbf{H}^{jk}\boldsymbol{e}^{\langle I\rangle} \\
&= \sum_b\sum_d \zeta_b^{\langle I\rangle}(q_{bd}\boldsymbol{v}_{jd}\boldsymbol{v}_{bk})(1 - \alpha^{\langle I\rangle}\lambda_d)\boldsymbol{v}_d - \alpha^{\langle I\rangle}\mathbf{H}^{jk}\boldsymbol{e}^{\langle I\rangle} \\
&= \sum_b\sum_d \zeta_b^{\langle I\rangle}(q_{bd}\boldsymbol{v}_{jd}\boldsymbol{v}_{bk})(1 - \frac{\sum_o \zeta_o^{\langle I\rangle 2}\lambda_o^2}{\sum_o \zeta_o^{\langle I\rangle 2}\lambda_o^3}\lambda_d)\boldsymbol{v}_d - \alpha^{\langle I\rangle}\mathbf{H}^{jk}\boldsymbol{e}^{\langle I\rangle}.
\end{aligned} \tag{107}$$

Similar to the analysis for the solver error, we consider the case when $D = 2$. We have

$$\begin{aligned}
\mathbf{B}_{\mathbf{A}_{jk}}^{\langle I+1\rangle} &= \sum_b\sum_d \zeta_b^{\langle I\rangle}(q_{bd}\boldsymbol{v}_{jd}\boldsymbol{v}_{bk})\boldsymbol{v}_d(1 - \frac{\sum_o \zeta_o^{\langle I\rangle 2}\lambda_o^2}{\sum_o \zeta_o^{\langle I\rangle 2}\lambda_o^3}\lambda_d) - \alpha^{\langle I\rangle}\mathbf{H}^{jk}\boldsymbol{e}^{\langle I\rangle} \\
&= \sum_b\sum_d \zeta_b^{\langle I\rangle}(q_{bd}\boldsymbol{v}_{jd}\boldsymbol{v}_{bk})\boldsymbol{v}_d(1 - \frac{\sum_o \zeta_o^{\langle I\rangle 2}\lambda_o^2}{\sum_o \zeta_o^{\langle I\rangle 2}\lambda_o^3}\lambda_d) - \alpha^{\langle I\rangle}\mathbf{H}^{jk}\boldsymbol{e}^{\langle I\rangle} \\
&= \sum_b \left(\zeta_b^{\langle I\rangle}(q_{b1}\boldsymbol{v}_{j1}\boldsymbol{v}_{bk})\boldsymbol{v}_1(1 - \frac{\zeta_1^{\langle I\rangle 2}\lambda_1^2 + \zeta_2^{\langle I\rangle 2}\lambda_2^2}{\zeta_1^{\langle I\rangle 2}\lambda_1^3 + \zeta_2^{\langle I\rangle 2}\lambda_2^3}\lambda_1) + \zeta_b^{\langle I\rangle}(q_{b2}\boldsymbol{v}_{j2}\boldsymbol{v}_{bk})\boldsymbol{v}_2(1 - \frac{\zeta_1^{\langle I\rangle 2}\lambda_1^2 + \zeta_2^{\langle I\rangle 2}\lambda_2^2}{\zeta_1^{\langle I\rangle 2}\lambda_1^3 + \zeta_2^{\langle I\rangle 2}\lambda_2^3}\lambda_2)\right) - \alpha^{\langle I\rangle}\mathbf{H}^{jk}\boldsymbol{e}^{\langle I\rangle} \\
&= \sum_b \left(\zeta_b^{\langle I\rangle}(q_{b1}\boldsymbol{v}_{j1}\boldsymbol{v}_{bk})\boldsymbol{v}_1(1 - \frac{\iota^2 + \tau^{\langle I\rangle 2}}{\lambda_2(\iota^3 + \tau^{\langle I\rangle 2})}\lambda_1) + \zeta_b^{\langle I\rangle}(q_{b2}\boldsymbol{v}_{j2}\boldsymbol{v}_{bk})\boldsymbol{v}_2(1 - \frac{\iota^2 + \tau^{\langle I\rangle 2}}{\lambda_2(\iota^3 + \tau^{\langle I\rangle 2})}\lambda_2)\right) - \alpha^{\langle I\rangle}\mathbf{H}^{jk}\boldsymbol{e}^{\langle I\rangle} \\
&= \sum_b \left(\zeta_b^{\langle I\rangle}(q_{b1}\boldsymbol{v}_{j1}\boldsymbol{v}_{bk})\boldsymbol{v}_1(1 - \frac{\iota^2 + \tau^{\langle I\rangle 2}}{\iota^3 + \tau^{\langle I\rangle 2}}\iota) + \zeta_b^{\langle I\rangle}(q_{b2}\boldsymbol{v}_{j2}\boldsymbol{v}_{bk})\boldsymbol{v}_2(1 - \frac{\iota^2 + \tau^{\langle I\rangle 2}}{\iota^3 + \tau^{\langle I\rangle 2}})\right) - \alpha^{\langle I\rangle}\mathbf{H}^{jk}\boldsymbol{e}^{\langle I\rangle}.
\end{aligned} \tag{108}$$

Now, for the upper bound, we set $\iota = \tau^{\langle I\rangle}$

$$\begin{aligned}
\|\mathbf{B}_{\mathbf{A}_{jk}}^{\langle I+1\rangle}\|_2 &= \|\sum_b \left(\zeta_b^{\langle I\rangle}(q_{b1}\boldsymbol{v}_{j1}\boldsymbol{v}_{bk})\boldsymbol{v}_1(1 - \frac{\iota^2 + \tau^{\langle I\rangle 2}}{\iota^3 + \tau^{\langle I\rangle 2}}\iota) + \zeta_b^{\langle I\rangle}(q_{b2}\boldsymbol{v}_{j2}\boldsymbol{v}_{bk})\boldsymbol{v}_2(1 - \frac{\iota^2 + \tau^{\langle I\rangle 2}}{\iota^3 + \tau^{\langle I\rangle 2}})\right) - \alpha^{\langle I\rangle}\mathbf{H}^{jk}\boldsymbol{e}^{\langle I\rangle}\|_2 \\
&\leq \|\sum_b \left(\zeta_b^{\langle I\rangle}(q_{b1}\boldsymbol{v}_{j1}\boldsymbol{v}_{bk})\boldsymbol{v}_1(1 - \frac{\iota^2 + \tau^{\langle I\rangle 2}}{\iota^3 + \tau^{\langle I\rangle 2}}\iota) + \zeta_b^{\langle I\rangle}(q_{b2}\boldsymbol{v}_{j2}\boldsymbol{v}_{bk})\boldsymbol{v}_2(1 - \frac{\iota^2 + \tau^{\langle I\rangle 2}}{\iota^3 + \tau_i^2})\right)\|_2 + \|\alpha^{\langle I\rangle}\mathbf{H}^{jk}\boldsymbol{e}^{\langle I\rangle}\|_2 \\
&\leq \|\sum_b \left(\zeta_b^{\langle I\rangle}(q_{b1}\boldsymbol{v}_{j1}\boldsymbol{v}_{bk})\boldsymbol{v}_1(1 - \frac{2}{1+\iota}\iota) + \zeta_b^{\langle I\rangle}(q_{b2}\boldsymbol{v}_{j2}\boldsymbol{v}_{bk})\boldsymbol{v}_2(1 - \frac{2}{1+\iota})\right)\|_2 + \|\alpha^{\langle I\rangle}\mathbf{H}^{jk}\boldsymbol{e}^{\langle I\rangle}\|_2 \\
&\leq \|\sum_b \left(\zeta_b^{\langle I\rangle}(q_{b1}\boldsymbol{v}_{j1}\boldsymbol{v}_{bk})\boldsymbol{v}_1(\frac{1-\iota}{\iota+1}) + \zeta_b^{\langle I\rangle}(q_{b2}\boldsymbol{v}_{j2}\boldsymbol{v}_{bk})\boldsymbol{v}_2(\frac{\iota-1}{\iota+1})\right)\|_2 + \|\alpha_i\mathbf{H}^{jk}\boldsymbol{e}^{\langle I\rangle}\|_2 \\
&= \|\frac{\iota-1}{\iota+1}\sum_b \left(\zeta_b^{\langle I\rangle}(q_{b1}\boldsymbol{v}_{j1}\boldsymbol{v}_{bk})\boldsymbol{v}_1 + \zeta_b^{\langle I\rangle}(q_{b2}\boldsymbol{v}_{j2}\boldsymbol{v}_{bk})\boldsymbol{v}_2\right)\|_2 + \|\alpha^{\langle I\rangle}\mathbf{H}^{jk}\boldsymbol{e}^{\langle I\rangle}\|_2 \\
&= \frac{\iota-1}{\iota+1}\|\mathbf{B}_{\mathbf{A}_{jk}}^{\langle I\rangle}\|_2 + \|\alpha^{\langle I\rangle}\mathbf{H}^{jk}\boldsymbol{e}^{\langle I\rangle}\|_2.
\end{aligned} \tag{109}$$

Unrolling the recursion, we have

$$\|\mathbf{B}_{\mathbf{A}_{jk}}^{\langle I\rangle}\| = \mathcal{O}(I\rho_{\text{SD}}^I). \tag{110}$$

Hence,

$$\|\mathbf{B}^{\langle I\rangle}\|_2 = \mathcal{O}(I\rho_{\text{SD}}^{I}). \tag{111}$$

Scenario (b). In scenario (a), we have $\frac{(\mathbf{I}-\alpha^{\langle I\rangle}\mathbf{A}_{\boldsymbol{\theta}})}{\partial \mathbf{A}_{jk}} = -\alpha^{\langle I\rangle}\mathbf{H}^{jk}$ with bounded norm of $\alpha^{\langle I\rangle}$. In scenario (b), we have

$$\frac{\partial(\mathbf{I}-\alpha^{\langle I\rangle}\mathbf{A}_{\boldsymbol{\theta}})}{\partial \mathbf{A}_{jk}} = \frac{\partial \mathbf{A}_{\boldsymbol{\theta}}}{\partial \mathbf{A}_{jk}}\frac{\partial(\mathbf{I}-\alpha^{\langle I\rangle}\mathbf{A}_{\boldsymbol{\theta}})}{\partial \mathbf{A}_{\boldsymbol{\theta}}} + \frac{\partial \alpha^{\langle I\rangle}}{\partial \mathbf{A}_{jk}}\frac{\partial(\mathbf{I}-\alpha^{\langle I\rangle}\mathbf{A}_{\boldsymbol{\theta}})}{\partial \alpha^{\langle I\rangle}} = -\alpha^{\langle I\rangle}\mathbf{H}^{jk} - \mathbf{A}_{\boldsymbol{\theta}}\frac{\partial \alpha^{\langle I\rangle}}{\partial \mathbf{A}_{jk}}. \tag{112}$$

This leads to the following,

$$\mathbf{B}_{\mathbf{A}_{jk}}^{\langle I+1\rangle} = (\mathbf{I}-\alpha^{\langle I\rangle}\mathbf{A}_{\boldsymbol{\theta}})\mathbf{B}_{\mathbf{A}_{jk}}^{\langle I\rangle} - (\alpha^{\langle I\rangle}\mathbf{H}^{jk} + \mathbf{A}\frac{\partial \alpha^{\langle I\rangle}}{\partial \mathbf{A}_{jk}})\boldsymbol{e}^{\langle I\rangle}. \tag{113}$$

It remains to be shown that the norm of $\mathbf{A}\frac{\partial \alpha^{\langle I\rangle}}{\partial \mathbf{A}_{jk}}$ can be upper-bounded. Then, the Jacobian error is upper-bonded by the same order of convergence as in scenario (a),

$$\|\mathbf{A}\frac{\partial \alpha^{\langle I\rangle}}{\partial \mathbf{A}_{jk}}\|_2 \leq \|\mathbf{A}_{\boldsymbol{\theta}}\|_2\|\frac{\partial \alpha^{\langle I\rangle}}{\partial \mathbf{A}_{jk}}\|_2 = \lambda_1\|\frac{\partial \alpha^{\langle I\rangle}}{\partial \mathbf{A}_{jk}}\|_2. \tag{114}$$

We write $\alpha^{\langle I\rangle}$ in terms of eigenvalues of $\mathbf{A}_{\boldsymbol{\theta}}$ (i.e. $\alpha^{\langle I\rangle} = \frac{\sum_o \zeta_o^{\langle I\rangle 2}\lambda_o^2}{\sum_b \zeta_b^{\langle I\rangle 2}\lambda_b^3}$) and take the derivative, i.e.

$$\frac{\partial \alpha^{\langle I\rangle}}{\partial \mathbf{A}_{jk}} = \frac{\left(\sum_o 2\zeta_o^{\langle I\rangle 2}\lambda_o \boldsymbol{v}_o^\top \mathbf{H}^{jk}\boldsymbol{v}_o\right)\left(\sum_b \zeta_b^{\langle I\rangle 2}\lambda_b^3\right) - \left(\sum_o \zeta_o^{\langle I\rangle 2}\lambda_o^2\right)\left(\sum_b 3\zeta_b^{\langle I\rangle 2}\lambda_b^2 \boldsymbol{v}_b^\top \mathbf{H}^{jk}\boldsymbol{v}_b\right)}{(\sum_m \zeta_m^{\langle I\rangle 2}\lambda_m^3)^2}. \tag{115}$$

Given the above, there exist a constant that bounds the norm of this derivative. We denote this constant by $\|\frac{\partial \alpha^{\langle I\rangle}}{\partial \mathbf{A}_{jk}}\|_2 \leq M_{\alpha_{\text{div}}}$.

F. Additional Experimental Details

F.1. Parameter Recovery for Noisy AR Models

F.1.1. Form of Prior Precision $\boldsymbol{\Gamma}_{\boldsymbol{\theta}}$

From (26), let us define $\boldsymbol{z}_{\leq P} := \{z_1, \dots, z_P\}$, $\boldsymbol{z}_{>P} := \{z_{P+1}, \dots, z_D\}$ and $\boldsymbol{w} := \{w_{P+1}, \dots, w_D\}$. Then,

$$\begin{bmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{H} & \mathbf{L}\end{bmatrix}\begin{bmatrix}\boldsymbol{z}_{\leq P}\\ \boldsymbol{z}_{>P}\end{bmatrix} = \begin{bmatrix}\boldsymbol{z}_{\leq P}\\ \boldsymbol{w}\end{bmatrix}, \tag{116}$$

where $\mathbf{H} \in \mathbb{R}^{(D-P)\times P}$ and $\mathbf{L} \in \mathbb{R}^{(D-P)\times(D-P)}$ such that

$$\mathbf{H} := \begin{bmatrix} -\phi_P & -\phi_{P-1} & \dots & -\phi_1\\ 0 & -\phi_P & \dots & -\phi_2\\ 0 & 0 & \dots & \vdots\\ 0 & 0 & \dots & -\phi_P\\ 0 & 0 & \dots & 0\\ \vdots & \vdots & \vdots & \vdots\\ 0 & 0 & \dots & 0 \end{bmatrix} \qquad \mathbf{L} := \begin{bmatrix} 1 & 0 & 0 & \dots & 0\\ -\phi_1 & 1 & 0 & \dots & 0\\ \vdots & \vdots & \vdots & \dots & \vdots\\ -\phi_{P-1} & -\phi_{P-2} & -\phi_{P-3} & \dots & 0\\ -\phi_P & -\phi_{P-1} & -\phi_{P-2} & \dots & 0\\ \vdots & \vdots & \vdots & \vdots & \vdots\\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}. \tag{117}$$

Observe that $\begin{bmatrix}\boldsymbol{z}_{\leq P}\\ \boldsymbol{z}_{>P}\end{bmatrix}$ is a multivariate Gaussian with mean $\mathbf{0}$ and inverse-covariance $\boldsymbol{\Gamma}_{\boldsymbol{\theta}}$ (i.e. our object of interest). Similarly, $\begin{bmatrix}\boldsymbol{z}_{\leq P}\\ \boldsymbol{w}\end{bmatrix}$ is also a multivariate Gaussian with mean $\mathbf{0}$ and inverse-covariance $\begin{bmatrix}\mathbf{Q}_{\boldsymbol{\phi}}^{-1} & \mathbf{0}\\ \mathbf{0} & \sigma^{-2}\mathbf{I}\end{bmatrix}$. Let $\mathbf{Q}_{\boldsymbol{\phi}}^{-1} = \sigma^{-2}\mathbf{D}$ for some matrix $\mathbf{D}$. The change-of-variables formula for probability distributions then tells us that

$$\boldsymbol{\Gamma}_{\boldsymbol{\theta}} = \begin{bmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{H} & \mathbf{L}\end{bmatrix}^\top \begin{bmatrix}\mathbf{Q}_{\boldsymbol{\phi}}^{-1} & \mathbf{0}\\ \mathbf{0} & \kappa^{-1}\mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{H} & \mathbf{L}\end{bmatrix} = \frac{1}{\kappa}\begin{bmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{H} & \mathbf{L}\end{bmatrix}^\top \begin{bmatrix}\mathbf{D} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{H} & \mathbf{L}\end{bmatrix}. \tag{118}$$

(Note that change-of-variables also tells us that $\log\det \mathbf{\Gamma}_{\boldsymbol{\theta}} = \log\det \mathbf{D} - T\log\kappa$, which we need to compute the term $\boldsymbol{c}_{\theta}$.)

? show that for a *stationary* AR process, we have $\mathbf{D} = \mathbf{L}_P^\top \mathbf{L}_P - \mathbf{H}_P^\top \mathbf{H}_P$, where $\mathbf{H}_P \in \mathbb{R}^{P\times P}$ is the first P rows of $\mathbf{H}$ and $\mathbf{L}_P \in \mathbb{R}^{P\times P}$ is the top-left $P \times P$ block of $\mathbf{L}$.

Next, let $\mathbf{D}$ be factorized as $\mathbf{R}\mathbf{R}^\top = \mathbf{D}$, which we can obtain through Cholesky decomposition. This implies that $\mathbf{\Gamma}_{\boldsymbol{\theta}} = \mathbf{X}^\top \mathbf{X}$, where

$$\mathbf{X} := \frac{1}{\sqrt{\kappa}} \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{H} & \mathbf{L} \end{bmatrix}. \tag{119}$$

We can verify that $\mathbf{X}$ is a lower triangular and banded matrix with $P+1$ non-zero bands below the diagonal. Similarly $\mathbf{X}^\top$ is an upper triangular and banded matrix with $P+1$ non-zero bands above the diagonal. It follows that $\mathbf{\Gamma}_{\boldsymbol{\theta}}$ has $2P+1$ non-zero bands. In turn, this implies that $\mathbf{A}_{\boldsymbol{\theta}}$ in (6) also has $2P+1$ non-zero bands (since the other part of the sum is a diagonal matrix).

F.1.2. Kalman Smoother Implementation of Exact-Gradient EM

Observe that one can easily write (26) as a state-space model by defining the state

$$\boldsymbol{s}_d := \begin{bmatrix} z_d \\ \vdots \\ z_{d+P-1} \end{bmatrix} \tag{120}$$

for $d = 1, \ldots, D$. Then, the noisy AR model of (26) is equivalent to the following state-space model:

$$\boldsymbol{s}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{\boldsymbol{\phi}}) \tag{121}$$

$$\boldsymbol{s}_d = \mathbf{F}\boldsymbol{s}_{d-1} + \boldsymbol{v}_d, \quad \boldsymbol{v}_d \sim \mathcal{N}(\mathbf{0}, \mathbf{V}) \tag{122}$$

$$y_d = \boldsymbol{c}^\top \boldsymbol{s}_d + w_d, \quad w_d \sim \mathcal{N}(0, W) \tag{123}$$

for

$$\mathbf{F} := \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ \phi_P & \phi_{P-1} & \phi_{P-2} & \dots & \phi_1 \end{bmatrix}, \quad \mathbf{V} := \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & \kappa \end{bmatrix}, \quad \boldsymbol{c} := \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad W = \lambda. \tag{124}$$

The Kalman smoother can be used to obtain the distributions $p(\boldsymbol{s}_d|\boldsymbol{y})$ for all d, which we can then convert into our posterior of interest $p(z_d|\boldsymbol{y})$. Kalman will require D steps with complexity $\mathcal{O}(P^3)$ each because the state size is P. We then follow Bishop & Nasrabadi (2006), Chapter 13.3.2 to compute the EM objective (5) using the Kalman smoother outputs.

F.1.3. Experimental Settings

It is not straightforward to directly optimize $\boldsymbol{\phi}$ over the space of *stationary* noisy AR processes. Thus, we parameterize $\boldsymbol{\phi} := f(\boldsymbol{\gamma})$, where $\boldsymbol{\gamma} \in [-1, 1]^P$ are partial auto-correlations and f is the transformation defined by **?**. Gradient descent is performed over $\boldsymbol{\gamma}, \log\kappa$ and $\log\lambda$. For each algorithm (i.e. gradient EM, probabilistic unrolling), we perform 200 iterations of gradient descent with the Adam optimizer and learning rate 0.1.

Each ground-truth value $\gamma_p^\star$ is randomly initialized between $[-1, 1]$. Similarly, $\kappa^\star$ and $\lambda^\star$ are randomly initialized between $[0.1, 10]$ (in log-space).

F.1.4. Comparison with VAEs

To provide another baseline of comparison for probabilistic unrolling, we fit the model in (26) using a variational auto-encoder (VAE) (Kingma & Welling, 2013). The decoder of the VAE is the generative model in (26), and the encoder of the VAE is a black-box deep neural network. To tune the VAE, we search over different architectures (i.e. 1, 2, 3, 4 layer models), different activations (i.e. ReLU, Sigmoid, Tanh, LeakyReLU) and adjust weights for the different parts of the VAE loss (i.e. the weight β in β-VAE (**?**)).

However, we find that VAEs perform poorly in parameter recovery for the noisy AR model. In Table 2, we observe that probabilistic unrolling (like EM) estimates all the true parameters $\{\phi^\star, \kappa^\star, \lambda^\star\}$ to within 1% error for $D = 30{,}000$ time points. On the other hand, the (optimally-tuned) VAE obtains $41.5 \pm 14.1\%$ error for ϕ, $215.93 \pm 223.5\%$ error for κ, and $37.9 \pm 23.75\%$ error for λ. We hypothesize that this poor performance is due to the VAE's biased objective in comparison to EM (and PU), which suffers from mean-field's inability to model covariance in the latent posterior; this is especially detrimental for this problem, because there is a lot of rich covariance structure across time.

F.2. Bayesian Compressed Sensing of Sparse Signals

F.2.1. Woodbury Matrix Identity for Exact-Gradient EM

The Woodbury matrix identity is a property from linear algebra that allows us to compute $\mathbf{\Sigma}_{\boldsymbol{\theta}}$ for Bayesian compressed sensing by inverting a $M \times M$ matrix instead of a $D \times D$ one. Since typically $M < D$ in compressed sensing, this can lead to computational savings in time from a practical standpoint (but perhaps not an asymptotic one, because M often needs to grow linearly with D in compressed sensing applications). Note that it does not lead to computational savings in space, because the full matrix $\mathbf{\Sigma}_{\boldsymbol{\theta}}$ is still computed in the end.

Instead of computing $\mathbf{\Sigma}_{\boldsymbol{\theta}}$ through (7), we equivalently have

$$\mathbf{\Sigma}_{\boldsymbol{\theta}} = \mathbf{\Gamma}_{\boldsymbol{\theta}}^{-1} - \mathbf{\Gamma}_{\boldsymbol{\theta}}^{-1}\mathbf{\Phi}_{\boldsymbol{\theta}}^\top\mathbf{\Omega}^\top(\mathbf{\Omega}\mathbf{\Psi}_{\boldsymbol{\theta}}^{-1}\mathbf{\Omega}^\top + \mathbf{\Omega}\mathbf{\Phi}_{\boldsymbol{\theta}}\mathbf{\Gamma}_{\boldsymbol{\theta}}^{-1}\mathbf{\Phi}_{\boldsymbol{\theta}}^\top\mathbf{\Omega}^\top)^{-1}\mathbf{\Omega}\mathbf{\Phi}_{\boldsymbol{\theta}}\mathbf{\Gamma}_{\boldsymbol{\theta}}^{-1}. \tag{125}$$

Note that for Bayesian compressed sensing, both $\mathbf{\Gamma}_{\boldsymbol{\theta}}^{-1}$ and $\mathbf{\Psi}_{\boldsymbol{\theta}}^{-1}$ can be cheaply computed, because these are both diagonal matrices.

F.2.2. Experimental Settings

The NIST dataset is accessed at `https://www.nist.gov/srd/nist-special-database-19`.

We scale all raw image pixels in NIST from $[0, 255]$ to the range $[0, 1]$. We add independent, pixel-wise Gaussian noise to the undersampled 2D Fourier transform with standard deviation $\sigma = 0.01$. During model fitting, we use the Adam optimizer with learning rate 1.0. We optimize the parameters $\boldsymbol{\alpha}, \beta$ in log-space. Each component of $\log \boldsymbol{\alpha}$ is initialized as randomly drawn from $\mathcal{N}(0, 1)$. The value $\log \beta$ is initialized as 0.

For probabilistic unrolling, we use the preconditioned conjugate gradient algorithm. The preconditioner we employ is the diagonal preconditioner $\mathbf{M}$ introduced in Lin et al. (2022b) for sparse Bayesian learning; $\mathbf{M}^{-1}$ is a diagonal matrix with diagonal $\boldsymbol{m}$, where

$$m_j := \frac{1}{\alpha_j + \beta}. \tag{126}$$

F.2.3. Sample Images

In Figure 3, we provide sample images of the true signal $\tilde{\boldsymbol{z}}^{(n)}$, its 15% undersampled and noisy Fourier transform measurement $\tilde{\boldsymbol{y}}^{(n)}$, and a reconstruction provided by probabilistic unrolling $\boldsymbol{\mu}^{(n)}$.

F.2.4. Results Breakdown by Digit Type

Table 5 presents a breakdown of the compressed sensing results by digit type. The average is given in Table 3.

F.2.5. Super-Efficiency of Network Gradient

In Figure 4, we empirically show that the network gradient (17) converges faster to the Monte Carlo gradient (10) than the output gradient (16) for our preconditioned conjugate gradient solver.

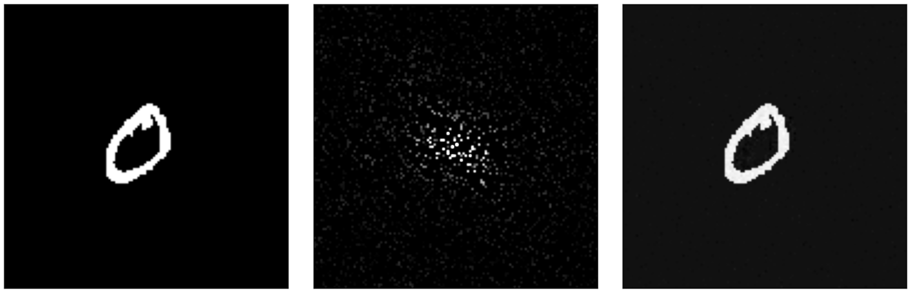

Figure 3. From left to right: the true signal, the Fourier measurement, and the probabilistic unrolling reconstruction.

Table 5. Compressed sensing results broken down by digit type.

Digit Type	$r(\boldsymbol{\mu}^{\text{EM}}, \tilde{\boldsymbol{z}})$	$r(\boldsymbol{\mu}^{\text{PU}}, \tilde{\boldsymbol{z}})$	EM Time	PU Time
0	3.89%	4.01 %	1475 s	21 s
1	4.76%	4.31 %	1483 s	21 s
2	4.61%	4.43 %	1473 s	21 s
3	3.96%	3.90 %	1508 s	21 s
4	7.20%	8.56 %	1485 s	21 s
5	5.22%	4.88 %	1494 s	21 s
6	4.17%	3.94 %	1510 s	21 s
7	4.67%	4.59 %	1467 s	21 s
8	4.27%	4.08 %	1468 s	21 s
9	4.83%	4.46 %	1448 s	21 s

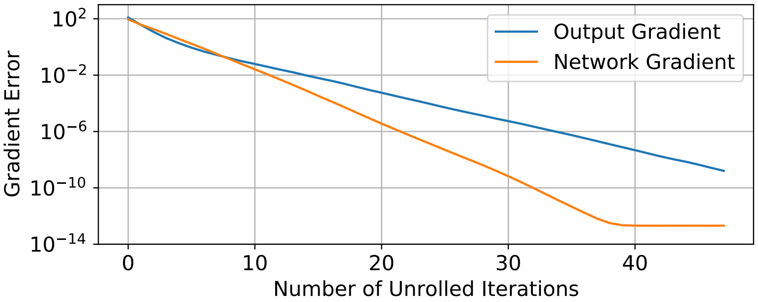

Figure 4. Gradient convergence as a function of iterations I.

F.3. Collaborative Filtering through Factor Analysis

F.3.1. Experimental Settings

The MovieLens datasets are accessed at `https://grouplens.org/datasets/movielens/`.

We follow the general experimental setup of Sedhain et al. (2015). For each dataset, we perform a 90%-10% train-test split of the ratings data. We then set aside 10% of the training set as a validation set. Both models (gradient EM and probabilistic unrolling) are trained with the Adam optimizer, learning rate 0.001, and a user mini-batch size of 25 (with gradient accumulation over four mini-batches for an overall batch size of 100). For the ML-1M dataset, we train the model for 20 epochs and evaluate the model on the validation set every 500 gradient steps. For the ML-10 M and ML-25 M datasets, we train the model for 5 epochs and evaluate the model on the validation set every 2,000 gradient steps. The checkpoint with the best validation set RMSE is used for final evaluation on the test set.

During training, we only train on users with at least one rating in the training set. Thus, there may be users in the validation/test sets that do not appear during training. Following Sedhain et al. (2015), we always predict a rating of 3 for these users. All other model rating predictions are clipped to be in the range [1.0, 5.0] before evaluation of RMSE.

F.3.2. Comparison with VAEs

Similar to Appendix F.1.4, we report results for a VAE baseline on the collaborative filtering task. Searching over architectural options, we find that the best-performing architecture had a two-layer encoder with 3,000 units in the hidden layer and ReLU activations. However, even with our extensive tuning, we observe that VAEs generally perform worse and have higher computational costs in comparison to probabilistic unrolling. The VAE obtains 0.8849 RMSE (compared to 0.8436 for PU) on MovieLens 1-m and 0.8366 RMSE (compared to 0.7796 for PU) on MovieLens 10-m. The VAE also has approximately 1.5x the time cost and 2x the memory cost of PU (due to the separate inference encoder network).