THE UNIVERSITY OF CHICAGO

STRUCTURED LOW-COMPLEXITY MATRIX AND TENSOR APPROXIMATION WITH APPLICATIONS

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

COMMITTEE ON COMPUTATIONAL AND APPLIED MATHEMATICS

BY
YIAN CHEN

CHICAGO, ILLINOIS
JUNE 2023

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to express my heartfelt gratitude to my two advisors, Professor Mihai Anitescu and Professor Yuehaw Khoo, for their invaluable guidance, support, and encouragement throughout my PhD journey. Their mentorship and expertise have been instrumental in shaping my research and personal growth.

I'm grateful to Professor Mihai Anitescu for leading me to the exciting research field of fast algorithms, providing extremely insightful feedback and instilling in me the enthusiasm for research and life. I would also like to thank him for supporting me with the research internship at Argonne National Laboratory and holding several in-depth discussions over the years to guide me through the unprecedented times during the pandemic and help establish the direction for my future career path. I'm thankful to Professor Yuehaw Khoo for constantly challenging me to think critically, generosity with his time and resources, and introducing me to so many interesting problems which greatly expanded my arsenal. His constant availability, willingness to go above and beyond the call of duty have been an inspiration to me and have made a significant impact on my academic and personal growth.

It is also my great honor to have Professor Jeremy Hoskins and Professor Michael Lindsey in my thesis committee. I thank them for providing critical insights and constructive feedback which have helped me refine my research questions and methodologies. I am forever grateful to have had the opportunity to work with them.

Finally, it has been my pleasure to meet all my friends and colleagues and receive their support and encouragement during this journey. I am fortunate enough to have had the opportunity to work with such outstanding individuals and to have been part of such an exceptional academic community. The year 2023 marks the official end of the pandemic era. I would like to express my deepest gratitude to my family and my two beloved cats for their constant companionship and trust through the times of hardship. Time may be sad and fleeting, but I find solace in the enduring presence of love and happiness by my side.

ABSTRACT

Matrix and tensor operations play a vital role in diverse fields such as machine learning, numerical analysis, computational physics and optimization. As data sizes continue to grow and high-dimensional problems become more challenging, low-complexity matrix and tensor approximations are becoming increasingly crucial. These approximations are essential in providing efficient and robust numerical solutions to these problems.

This thesis focuses on two types of efficient low-complexity matrix and tensor formats, hierarchical matrices for efficient matrix compression and tensor-train (TT) for efficient high-dimensional tensor compression. Chapter 3 utilizes hierarchical matrix techniques and proposes a scalable algorithm for traditional tasks in Gaussian processes (GPs) applications, including maximum likelihood estimations (MLEs) for parameter identification, computing confidence intervals for uncertainty quantification, and regressions. In Chapter 5 and 4, we propose a novel approach for compressing Gibbs-Boltzmann distributions in statistical mechanics into TT formats and their downstream applications in computing committor functions.

Low-complexity formats provide a compact representation of matrices and tensors by exploring their numerical structure. This reduces memory usage and enables fast algorithms that exploit the inherent low-rankness of the problem and data. Numerical evidence is provided to demonstrate the scaling of the algorithms, as well as the efficiency/accuracy trade-offs.

CHAPTER 1

INTRODUCTION

The thesis comprises four projects on low-complexity matrix and tensor approximations and their applications in statistical modeling, computational physics, and more. In recent years, there has been a significant increase in the amount of data generated from various sources, and this trend is expected to continue. The growth in data volume and the need to work with high-dimensional models has led to a surge in the use of algorithms that rely on linear algebra operations, including matrix and tensor operations. However, traditional linear algebra methods can become prohibitively expensive when dealing with large-scale datasets, making them impractical for many applications.

To tackle this challenge, the development and implementation of low-complexity matrix and tensor approximations, such as sparse [183, 63], low-rank [18, 51, 116], hierarchical [4, 31, 32, 212, 77], and tensor-train representations [141, 1, 143, 204], have been demonstrated to significantly reduce computational and storage complexity without compromising the accuracy of the results. These techniques have widespread applications in various fields, such as scientific computing, machine learning, computational physics, statistical mechanics, etc.

Despite their advantages, low-complexity matrix and tensor approximations pose some challenges. For instance, the trade-off between compression and accuracy is a significant consideration when selecting a representation format. Moreover, there is no universal solution for choosing the optimal format for a particular problem. Hence, the development of efficient algorithms for manipulating these formats is crucial.

The aim of this thesis is to highlight the significance of low-complexity matrix and tensor approximations in developing fast algorithms and handling large-scale datasets. Specifically, we leverage these efficient numerical linear algebra tools to design efficient new algorithms or improve the scalability of traditional algorithms to make them suitable for modern applications. We provide concrete numerical examples to demonstrate the effectiveness and efficiency of the proposed algorithms.

In this thesis, we concentrate on multivariate GPs with physics-based outputs in Chapters 2 and 3. For single-output processes, the typical GP modeling workflow involves specifying the covariance function, fitting the unknown parameters using MLEs, and performing GP regression with the estimated covariance model. However, with multiple-output processes of physical relations, it is difficult to describe the process in order to correctly structure the outputs and ensure positive definitiveness. The physics-based covariance model, based on the work in [57], typically takes the form of a sequential product of Jacobians of physical operators and common covariance models. For large-scale physical operators, the exact covariance model is usually prohibitively large to evaluate. Conversely, GPs have notoriously bad scaling ($O(n^3)$) in nearly all related tasks, where n is the size of the covariance matrix. Various approximation methods can be employed to reduce the computational and storage complexities of traditional GP applications, e.g. low-rank methods [170, 135], matrix tapering [68, 93, 44], hierarchical matrices [33, 131, 122, 5, 47]. In Chapter 2, we assume that the latent covariance model is smooth and globally low-rank. The physics-based covariance model for the output process can then be approximated by an easily invertible matrix plus a low-rank matrix. Based on the low-rank structure of the covariance model, a range of matrix operations can be accelerated. As a result, we can perform GP tasks, including MLEs, Fisher information estimations, and regressions, in quasilinear time.

Low-rank approximations are effective when the data contain limited information about the parameters, but become prohibitive as the data become more informative [3]. In Chapter 3, we aim to relax the assumptions on the covariance model and physical operators to make the approach more widely applicable. Instead of low-rank, we assume the output covariance matrix has a hierarchical structure. We compress the covariance matrix into a special hierarchical matrix structure, namely the hierarchically off-diagonally low-rank

(HODLR) matrices [5, 6], to accelerate the numerical linear algebra operations. Unlike conventional approaches used in literature [5, 71, 122], we propose a method to construct the HODLR approximation of the covariance matrix based on randomized sketching, which avoids the expensive step of evaluating the full covariance matrix. As a result, we can perform all downstream GP tasks, including MLEs, Fisher information estimations, and regressions, in quasilinear time with a much milder assumption. In both chapters, we demonstrate the numerical scaling, approximation accuracy, and MLE trajectories of the proposed algorithms with exact computations using the full unstructured covariance matrix.

In Chapters 4 and 5, we explore the applications of another popular low-complexity tensor structure, the tensor train (TT) approximation, in computational physics and statistical mechanics [141, 1, 143, 204]. In Chapter 4, we address the challenge of computing the committor function in complex systems. Mathematically, this involves solving the high-dimensional backward Kolmogorov equation, which can scale exponentially with the number of dimensions when using traditional finite difference or finite element approaches. To overcome this issue, we propose a novel approach that parameterizes the committor function as a TT, which significantly speeding up the computations. Furthermore, we assume that the Boltzmann distribution of the system can be efficiently compressed in a TT representation. With these assumptions, we can solve the committor function in computational and memory complexity that both scale linearly with the number of dimensions.

The performance of the approach presented in Chapter 4 critically depends on the ability of efficiently approximating the Boltzmann distribution of the system in TT format. Several methods are available for constructing the TT approximation, such as TT cross approximation and recent works on TT density estimation from empirical samples. In Chapter 5, we propose a novel approach that combines the auxiliary-field Monte Carlo (AFMC) method [14, 12, 13] in statistical mechanics with TT. A key feature of our approach is that the Monte Carlo sampling and TT recompression processes can be fully parallelized, enabling the algorithm to take full advantage of modern computing hardware. The total computational complexity of the algorithm is linear in both the number of dimensions and number of samples. With Chapters 4 and 5, we have a complete workflow for computing the committor function of a many-body system.

This thesis contains material from two published papers by the author [49, 50] one accepted paper and one unpublished draft. In particular, Chapter 2 is based on [49], coauthored with Prof. Mihai Anitescu; Chapter 3 is based on an accepted paper, coauthored with Prof. Mihai Anitescu; Chapter 4 is based on [50], coauthored with Prof. Yuehaw Khoo; Chapter 5 is based on an unpublished draft, coauthored with Prof. Yuehaw Khoo. Some material from each of these papers has also been incorporated into this introductory chapter. In the following, we provide more detailed overviews for each chapter.

1.1 Overview of Chapter 2

In Chapter 2, we consider traditional GP applications with physics-based covariance models proposed in [57]. We further assume the covariance model of the latent process is smooth, which results in a low-rank latent covariance matrix. Consequently, we can approximate the observable covariance matrix as $(K_\epsilon + LK_zL^T)$, where L is the Jacobian of some physical operator, K_z is the low-rank latent covariance matrix, and K_ϵ is the covariance for the observation noise, which we assume is easily invertible. With these assumptions, we parameterize the observable covariance model as an easily-invertible matrix plus globally low-rank components. The downstream GP tasks rely heavily on numerical linear algebra operations, e.g. computing the log-determinant of the matrix, solving matrix linear systems, and performing matrix-vector products. We leverage the Woodbury matrix identity [210], Sylvester's determinant identity [180], and Hutchinson's trace estimator [10] to perform all GP tasks efficiently, including evaluating the log-likelihood, score equations, Fisher information matrix, and conditional expectations, in quasilinear $(O(n \log^2 n))$ time.

The physics-based covariance model, in its simplest form, is obtained by truncating the Taylor series up to the second order, which is equivalent to approximating the physical dynamics by linear approximations. In addition, we propose a way to efficiently compute higher-order terms and include up to fourth order term in the Taylor expansion to account for nonlinearities. The resulting covariance model has the same structure with only slightly inflated ranks. Nevertheless, all operations with the covariance matrix remain in the same order of complexity. We demonstrate the algorithm's numerical scaling, parameter estimation accuracy, and MLE trajectories using a synthetic example with an $1D$ Burger's equation and a real-world example with a tropical climate model.

1.2 Overview of Chapter 3

In Chapter 3, we use the same physics-based covariance model, but with a much milder assumption. Instead of assuming a low-rank latent covariance, we assume the covariance of the observable process has a hierarchical structure. Recently, hierarchical matrices have been used to improve the scaling of common matrix operations by exploiting the decay of correlations with physical distances [4, 31, 32, 212, 77]. These matrices originated from the fast multipole method (FMM) in physics [153] and are based on the assumption that different clusters of observations with long pairwise distances have low-rank cross-covariance structure. Though the entire covariance matrix could still be full-rank, some local blocks, mostly off-diagonal blocks, preserve low-rank structure. Many well-designed numerical algorithms can effectively exploit the locally low-rank structure and achieve better scaling in most linear algebra operations, including matrix-vector products, matrix-matrix products, determinants, factorizations, matrix inverse and more. We refer readers to [77] for a detailed discussion. The technique has also been proposed for Gaussian process computations and parameter estimations in several works [33, 122, 5, 47, 160]. However, current approaches rely on an efficient $O(1)$ evaluator of the covariance matrix and cannot directly deal with derivatives.

In this chapter, we focus on the application of HODLR matrices and propose a novel approach to compress the physics-based covariance model using randomized sketching. We design patterned random sketching matrices to sketch specific local blocks of the covariance matrix. The entire procedure doesn't require the explicit evaluation of the entire covariance matrix and takes only $O(\log n)$ matrix-vector products.

Moreover, we leverage the differentiability of randomized sketching and matrix decompositions to design an efficient algorithm to construct the derivatives of the covariance matrix with respect to unknown parameters. The derivatives of the covariance matrix also retain the same HODLR structure, providing computational advantages in the same manner. Using the efficient HODLR factorization algorithm developed in [6], we can evaluate the log-likelihood, the score equations, and the empirical Fisher information matrix all in $O(n \log^2 n)$ time complexity and $O(n \log n)$ memory complexity. We provide two synthetic numerical examples based on the Mat'ern kernel to demonstrate the numerical scaling, accuracy of the estimated quantities, and MLE performances.

1.3 Overview of Chapter 4

In Chapter 4, we consider the problem of solving the committor function for a high-dimensional stochastic dynamic system. This requires us to numerically solve the backward Kolmogorov equation,

$$-\beta^{-1}\Delta q(\boldsymbol{x}) + \nabla V(\boldsymbol{x}) \cdot \nabla q(\boldsymbol{x}) = 0 \text{ in } \Omega \backslash (A \cup B),\ q(\boldsymbol{x})|_{\partial A}= 0,\ q(\boldsymbol{x})|_{\partial B}= 1. \tag{1.3.1}$$

where q is the unknown committor function to solve and V is the potential field of the system. Recently, researchers propose to solve the variational optimization problem in the form

$$\arg\min_{q} \int_{\Omega} |\nabla q(\boldsymbol{x})|^2 p(\boldsymbol{x})\, d\boldsymbol{x}, \quad q(\boldsymbol{x})|_{\partial A} = 0,\ q(\boldsymbol{x})|_{\partial B} = 1, \tag{1.3.2}$$

and parametrize the committor function with deep neural networks [95, 115, 114]. The objective function (1.3.2) is then approximated by Monte Carlo integration with training samples from the equilibrium distribution corresponding to the potential field V. Inspired by this idea, we leverage the tensor network methods in modern quantum physics and parameterize the unknown committor function as a TT. We further assume the equilibrium density of the system can be well-approximated in TT format. These assumptions allow the high-dimensional integration in the variational objective to reduce to simple tensor contractions. We finally solve the optimization problem via alternatively minimizing each TT tensor core. The total computational and storage complexity of the algorithm is linear with the number of dimensions. We provide extensive numerical experiments on several potential models in the committor function community, including the double-well potential, Ginzburg-Landau potential [84] and Gaussian mixture models.

1.4 Overview of Chapter 5

In Chapter 5, we address the problem of compressing the Boltzmann distribution of a many-body system into a low-complexity tensor format, which can significantly simplify all downstream computations. In the classical setting, the key objective is to compute the partition function of the density. We aim to find a TT approximation of the Boltzmann distribution that can enable us to perform these computations efficiently. Once the approximation is obtained, computing the partition function can be achieved by taking the dot product of the TT with an all-one tensor in $O(d)$ time, where d is the number of dimensions. Similarly, other related tasks, such as drawing i.i.d samples, drawing conditional samples, and evaluating marginal distributions, can also be done in $O(d)$ time.

In the quantum setting, the goal is to find the ground state energy of the Hamiltonian. To achieve this, we use Trotter decomposition and approximate the imaginary-time propagator,

$$(I - \beta H) \approx \exp(-\beta H), \tag{1.4.1}$$

where H is the Hamiltonian of the system and β is the infinitesimal time step. We aim to compress the propagator into an matrix product operator (MPO) format, which is a higher-dimensional analogy of TT. Once this is done, we can compute the ground state energy by efficiently and successively applying the propagator to a test wavefunction.

The main tool we use to make the approximation is the AFMC method [14, 12, 13], along with its quantum counterpart, the auxiliary-field quantum Monte Carlo (AFQMC) method [220, 168, 28, 45]. These methods decouple the correlations between pairs of particles by introducing an auxiliary field and approximating the Boltzmann distribution with Monte Carlo samples from the auxiliary field. In our work, we take this sampling procedure one step further by compressing each sample as a TT in the classical setting or an MPO in the quantum setting. We then apply randomized sketching to reduce the effective rank. The rank of the final approximation is independent of the number of samples used and is solely determined by the nature of the system. Additionally, the most time-consuming parts of the algorithm (sampling and sketching) are fully parallelizable, making the algorithm suitable for modern computational hardware.

CHAPTER 2

SCALABLE GAUSSIAN PROCESS ANALYSIS FOR IMPLICIT PHYSICS-BASED COVARIANCE MODELS

2.1 Introduction

GPs have been widely applied in different communities for several tasks, benefiting from their nonparametric nature and accessible likelihood expression. GPs are considered standard methods of Bayesian inference [148]. For overviews of their usage, one can refer to several extensive monographs [148, 125, 209]. Gaussian process regression, or kriging, is used for interpolating or predicting a range of natural phenomena in geophysics and biology [176, 209, 101], taking advantage of its ability to provide fully probabilistic estimates of the uncertainty of the predictions. Extending the technique to multiple target output variables is known as cokriging [176, 54] or multikriging [35]. In this chapter, we focus on multivariate cokriging where the variables can be extracted from spatial processes or spatiotemporal processes.

The main computational expense of GP regression generally consists of solving linear systems with the kernel matrix. For an $n \times n$ dense kernel matrix, however, the general Cholesky decomposition approach scales as $O(n^3)$, which can easily become problematic with increasing size. Over the past decades, devising approximations to reduce the computational load has become an active research field. The approaches include active set methods [146] and partitioning methods [187, 134] that separate the training data to induce sparse representation of the full problem. More significant efforts rely on efficiently obtaining low-rank approximations of the dense covariance matrix. These include global low-rank methods [170, 214], which find a compressed low-rank matrix by selecting or constructing a set of landmark points, and local low-rank approximation methods, which apply the framework of hierarchical matrices [5, 33] to obtain better time and storage complexity for certain common operations.

Another important feature of GP-based approaches is that the related probability densities can be fully characterized by their mean and covariance functions. Correct and accurate covariance models play an essential role in setting up meaningful GP regression problems. Many efforts aim to incorporate prior knowledge about the physics of the processes into the covariance model structure, which may improve the efficiency of uncertainty quantification and provide physically consistent results. A typical strategy of including prior knowledge of a real system governed by known physical laws is hierarchical Bayesian modeling [69, 20]. Examples include atmospheric modeling [207, 20, 156, 53] and environmental sciences [20, 21, 22, 208, 55]. In [57], one of the authors of this work proposed a systematic framework of assembling consistent auto- and cross-covariance models based on known physical processes. The endeavor showed that significant improvements in the forecast efficiency can be achieved by using physics-based covariance models. But some of the computational issues mentioned above for GP analysis that may appear for large datasets have nonetheless not been fully addressed.

In this chapter, we combine the ideas of low-rank approximation and physics-based covariance models to propose a computationally efficient, physically consistent method for identifying GP covariance parameters and carrying out the regression. We assume that the physical model is implicit, in the sense that all that is available for interacting with it is a mapping between the unknown fields (initial conditions, boundary conditions, and stochastic-noise type forcing) that can be characterized by parameterized Gaussian processes and the output. We propose an efficient approach for estimating the covariance model parameters by maximum likelihood estimation based on a low-rank approximation of this mapping. Our approach achieves quasilinear complexity in the number of data points both for evaluating the log-likelihood and for carrying out the GP regression. The approach can be directly applied to linear processes. For nonlinear processes, we propose additional operations to approximate the nonlinearity with little additional cost. In this work, we aim to increase the flexibility of carrying out GP analysis by minimizing the required prior knowledge of the physical processes, providing more choices of solvers, and obtaining more control of the tradeoff between computational speed and accuracy.

The rest of the chapter is organized as follows. The implicit physics-based GP regression problem is formulated and several useful techniques are reviewed in Section 2.2. Then scalable GP regression algorithms are developed in Section 2.3. Scalable maximum likelihood parameter estimations and uncertainty quantification are proposed in Section 2.4. Extensive numerical tests for both model simulated data and real climate data are presented in Section 2.5 to evaluate the effectiveness of the proposed algorithms. We conclude with a summary and discussion in Section 2.6.

2.2 Implicit Physics-Based Covariance Models and Low-Rank Approximation

To facilitate the theoretical development of our approach, we first assume a general physical model structure to work on. Consider a general form of a deterministic model for physical processes:

$$y = F(z),\ z = (z_{x_1}, z_{x_2}, \cdots, z_{x_n})^T, \tag{2.2.1}$$

where y is an m-dimensional random field, z is an n-dimensional random field, and $F \in \mathbb{R}^n \to \mathbb{R}^m$ is a sufficiently regular mapping. To distinguish the two quantities, we call y the output process (i.e., the process we can partially observe and want to predict) and z the latent process (i.e., the process that is usually not directly observed but is related to the output). We assume that the observations of y denoted by y_o and observations of z denoted by z_o are affected by observation noise

$$y_o = y + \epsilon_y, \ z_o = z + \epsilon_z, \tag{2.2.2}$$

where ϵ_y, ϵ_z are independent Gaussian processes with covariance $K_{\epsilon_y} = \text{Cov}(\epsilon_y, \epsilon_y)$ and $K_{\epsilon_z} = \text{Cov}(\epsilon_z, \epsilon_z)$, respectively. Note that our framework allows multiple output components and multiple components in the latent process. Each component may have different levels of observation noise due to their possibly different physical nature, measurements, units, and conditions. We assume the observation noise covariance K_{ϵ_y} and K_{ϵ_z} can be efficiently inverted with at most quasilinear time cost. For example, in a lot of cases, such as when each component is measured by a different instrument, the noise processes ϵ_y and ϵ_z are componentwise independent and can many times be assumed to be Gaussian [216]. Then their covariance matrix is only a diagonal matrix with entries representing different noise magnitudes, and it is easily invertible with linear time cost. We thus find that our assumptions include a broad set of models.

We also note that many models can be adapted or converted to this form. For example, many physical processes can be modeled by using differential equations or stochastic differential equations. Additional constraints such as initial conditions and boundary conditions are needed in order to solve these equations. In this case, the model can be expressed by

$$y = G(B, I, \omega), \tag{2.2.3}$$

where B denotes the boundary condition, I denotes the initial condition, and ω is the random process component of the stochastic differential equation. To convert the model into general form, we can concatenate the latent variables into a column vector of latent process $z^T = (B^T, I^T, \omega^T)$. Note that we will use the general form (2.2.1) in the theoretical derivation of this study; however, form (2.2.3) may be simpler to use in applicable cases since separating independent variables may provide a more compact solution and reduce the size of the problem.

2.2.1 Physics-Based Covariance Models

Gaussian process regression requires good covariance models that can accurately capture the joint variability of the processes. By incorporating information from the physical model (2.2.1), we use physics-based covariance models proposed in [57]. Note that although [57] provides covariance models only for spatial processes, they can be extended to the spatiotemporal context with little extra effort.

Second-Order Covariance Model

Denote the expectation $\mathrm{E}(z) = \bar{z}$ and the small perturbation around the mean value by $\delta z = z - \bar{z}$. If two processes satisfy the physical constraint given by (2.2.1),(2.2.2) for $F \in \mathcal{C}^2$, the covariance matrix formed by y_o and z satisfies

$$\operatorname{Cov}\left(\begin{bmatrix} y_o \\ z \end{bmatrix}\right) = \begin{bmatrix} L\operatorname{Cov}(z,z)L^T + K_{\epsilon_y} & L\operatorname{Cov}(z,z) \\ \operatorname{Cov}(z,z)L^T & \operatorname{Cov}(z,z) \end{bmatrix} + O(||\delta z||^3), \tag{2.2.4}$$

where L is the Jacobian matrix of F evaluated at $\bar{z}$. If we used this matrix as the estimate of the joint covariance, we would incur a third-order error in δz. If the function F is linear, the error term vanishes, and the covariance model is exact. This covariance model can be applied when the corresponding physical process can be exactly or approximately well represented by its linearization. If the function is highly nonlinear, extra higher-order terms can be added to the covariance model to account for the nonlinearity and lead to more accurate results.

Higher-Order Covariance Model

To use higher-order expansions, we will need to compute tensor operations. In particular, note that for H, the Hessian tensor of $F(z)$ defined in (2.2.1), a vector $v \in \mathbb{R}^n$, we have that $v^T H v$ is a column vector, the transpose of which is the row vector $v^T H^T v$. We give details on the tensor algebra we used to obtain our derivations in A.2.

With these conventions, the covariance model under higher-order closure is given by [57]

$$\operatorname{Cov}\left(\begin{bmatrix} y_o \\ z \end{bmatrix}\right) = \begin{bmatrix} K_{11} + K_{\epsilon_y} & L\operatorname{Cov}(z,z) \\ \operatorname{Cov}(z,z)L^T & \operatorname{Cov}(z,z) \end{bmatrix} + O(||\delta z||^4), \tag{2.2.5}$$

where L is the Jacobian matrix, H is the Hessian tensor of $F(z)$ evaluated at $\bar{z}$, and from [57]

$$K_{11} = L\operatorname{Cov}(z,z)L^T + \frac{1}{4}\overline{\delta z^T H \delta z \delta z^T H^T \delta z} - \frac{1}{4}\overline{\delta z^T H \delta z}\;\overline{\delta z^T H^T \delta z}. \tag{2.2.6}$$

This covariance model includes terms up to order four. If the function F is quadratic, the error term vanishes, and the covariance model is exact. For more complicated processes, the truncated covariance model (2.2.5) can be used as a good approximation of the exact covariance matrix. We note that the higher-order terms appear only in the auto-covariance. The cross-covariance remains the same under both second-order and higher-order closure assumption.

2.2.2 Low-Rank Approximation of Kernel Using Chebyshev Interpolation

Carrying out GP regression in the general covariance case requires computing the inverse of, or at least solving linear systems with, large, dense covariance matrices. Conventional methods based on Cholesky decomposition are expensive since the computational cost scales as $O(n^3)$ for an $n \times n$ dense matrix. To obtain a computationally efficient method for solving the GP regression problem, we use low-rank approximation methods that facilitate linear algebra with better scaling. In this study, we approximate the covariance matrix of the latent process $\mathrm{Cov}(z, z)$ by Chebyshev interpolation. Note that low-rank kernel approximation using other choices of interpolation methods and nodes is also possible; we refer readers to [214] for a discussion. If the Gaussian process describing z is smooth enough, the Chebyshev interpolation is sufficient for our purpose. In some circumstances, particularly at high sample density, using smooth covariance models may not be a suitable representation of the uncertainty [176], and in these cases our approach would not produce a good approximation. Our approach can be relatively easily extended to a block diagonal plus low-rank structure of $\mathrm{Cov}(z, z)$ if we make some assumptions about the resulting covariance or if we use the fact that for many applications L itself is low rank [2] to allow for cases where the covariance is rougher. For this paper we restrict ourselves to the smooth $\mathrm{Cov}(z, z)$, which we will show can have a positive impact for some observed data problems in Section 2.5.

Chebyshev nodes are roots of Chebyshev polynomials of the first kind. Using Chebyshev nodes as interpolation points in polynomial interpolation (Chebyshev interpolation) can help minimize the effect of Runge's phenomenon. Given the number of interpolation points N, in the interval $(-1, 1)$, the Chebyshev nodes are defined by

$$\tilde{x}_k = \cos\left(\frac{2k-1}{2N}\pi\right), \; k = 1, 2, \cdots, N. \tag{2.2.7}$$

For a 1D smooth function $f : (-1, 1) \to \mathbb{R}$, given function evaluations at the set of interpolation points $\{(\tilde{x}_1, f(\tilde{x}_1)), (\tilde{x}_2, f(\tilde{x}_2)), \cdots, (\tilde{x}_N, f(\tilde{x}_N))\}$, then for any $x \in (-1, 1)$, $f(x)$ can be approximated by using Lagrange polynomials:

$$\tilde{f}(x) \approx \sum_{i=1}^{N} \left(\prod_{j \neq i} \frac{x - \tilde{x}_j}{\tilde{x}_i - \tilde{x}_j} \right) f(\tilde{x}_i). \tag{2.2.8}$$

The approach can be generalized to functions over arbitrary interval domains by affine transformation. Now our goal is to construct a low-rank compressed kernel matrix K from which the kernel matrix $\mathrm{Cov}(z, z)$ can be approximately reconstructed by Chebyshev interpolation. Assume $z(x) = (z(x_1), z(x_2), \cdots, z(x_n))^T$ is a random field over an n-dimensional location set $(x_1, x_2, \cdots, x_n)$. We then can find N Chebyshev nodes $(\tilde{x}_1, \tilde{x}_2, \cdots, \tilde{x}_N)$ over the location set. Define the compressed covariance matrix $K \in \mathbb{R}^{N \times N}$ by

$$K = (K_{ij}),\ K_{ij} = \mathrm{Cov}(z(\tilde{x}_i), z(\tilde{x}_j)),\ i, j = 1, 2, \cdots, N. \tag{2.2.9}$$

In addition, define the coefficient vector associated with x by

$$c(x) = \left(\prod_{j \neq 1} \frac{x - \tilde{x}_j}{\tilde{x}_1 - \tilde{x}_j}, \prod_{j \neq 2} \frac{x - \tilde{x}_j}{\tilde{x}_2 - \tilde{x}_j}, \cdots, \prod_{j \neq N} \frac{x - \tilde{x}_j}{\tilde{x}_N - \tilde{x}_j} \right)^T \in \mathbb{R}^N. \tag{2.2.10}$$

Construct $C_z = (c(x_1), c(x_2), \cdots, c(x_n)) \in \mathbb{R}^{N \times n}$. Then we can approximate $\mathrm{Cov}(z, z)$ by $C_z^T K C_z$, which is of rank N. If we take $N < n$, it becomes a valid low-rank approximation of $\mathrm{Cov}(z, z)$. Note that the same Chebyshev interpolation can be easily extended to multiple dimensions by using tensor products. It can be useful when the latent process has more than one dimension.

A practical question on the way to applying the method is how to choose the effective rank N to balance accuracy and complexity. As a guideline, the smoother the kernel function f, the faster the decay of its numerical rank. In other words, the error in polynomial approximation decreases rapidly as the rank increases. Specifically, let $\|\cdot\|_\infty$ denote the supremum norm on $(-1, 1)$. If f is analytic in some open region of the complex plane containing $(-1, 1)$, then there exist $\rho > 1$ and a positive constant C such that $\|\tilde{f} - f\|_\infty < C\rho^{-N}$ [186]. This means that for analytic f, which is the case for typical kernel functions, the approximation error decays with N at a geometric rate. This provides us a way to estimate the required rank to get desired accuracy, though it is highly dependent on the choice of kernel function. In this work, we assume that the kernel function is smooth enough and can be well-approximated as a block bivariate function by a rank $N = O(\log n)$ Chebyshev interpolated matrix.

2.2.3 Approximation of Jacobian and Hessian

As we have noted, the physics-based covariance model requires computing the Jacobian and Hessian matrices of the function F at given points. Since the physical model is implicit, we have no algebraic expression of F, and the information of the model is available only via some black-box forward solvers. To avoid direct estimation and storage of the Jacobian and Hessian, we use a central difference scheme to efficiently approximate the Jacobian-vector product and the vector-Hessian-vector product for any vectors. While automatic differentiation is always a possibility, when one works with complex models such as climate codes, getting automatic differentiation to work is a complex and time-consuming endeavor [126]. Our simple approach has downsides, including introducing additional approximation errors, but it is highly parallelizable. Here are the approximations we use.

- Jacobian-vector product Lu for vector $u \in \mathbb{R}^n$

Let s be a sufficiently small positive real number. Then by Taylor expansion, we can approximate Lu by

$$Lu = \frac{1}{2s}(F(\bar{z} + su) - F(\bar{z} - su)) + O(s^2||u||^3). \tag{2.2.11}$$

- Vector-Hessian-vector product $u^T Hv$ for vector $u, v \in \mathbb{R}^n$

Let $L(z)$ be the Jacobian matrix of F evaluated at z. Let s_1 be a sufficiently small positive real number. Consider the Taylor expansion of $L(\bar{z} + s_1 v)$ and $L(\bar{z} - s_1 v)$,

$$L(\bar{z} + s_1 v)u = L(\bar{z})u + s_1 v^T H u + O(s_1^3 ||v||^3), \tag{2.2.12}$$

$$L(\bar{z} - s_1 v)u = L(\bar{z})u - s_1 v^T H u + O(s_1^3 ||v||^3). \tag{2.2.13}$$

Subtracting the two equations, we get the approximation

$$v^T H u = \frac{1}{2s_1}(L(\bar{z} + s_1 v)u - L(\bar{z} - s_1 v)u) + O(s_1^2 ||v||^3). \tag{2.2.14}$$

Combine with the approximation of the Jacobian-vector product (2.2.11). We get

$$\begin{aligned} v^T H u =& \frac{1}{4 s_1 s_2}[F(\bar{z} + s_1 v + s_2 u) - F(\bar{z} + s_1 v - s_2 u) - F(\bar{z} - s_1 v + s_2 u) \\ &+ F(\bar{z} - s_1 v - s_2 u)] + O(s_1^2 ||v||^3 + \frac{s_2^2 ||u||^3}{s_1}). \end{aligned} \tag{2.2.15}$$

For both Jacobian and Hessian approximations, when $s_1, s_2 \to 0$ and $\frac{s_2}{s_1}$ is bounded above, the error tends to zero. In practice, we choose s_1, s_2 small and of the same order to make $s_2 u, s_1 v$ be small perturbations around $\bar{z}$.

2.3 A Scalable Approach for Gaussian Process Regression Using Physics-Based Covariance Models

The GP regression problem aims to interpolate the whole field y by partial observations. Specifically, we can partition the whole field y into an observed part and an unobserved part. Assume the partial observations (with observation noise) of y are denoted by $y_o \in \mathbb{R}^d$ and the remaining unobserved field that we want to predict is denoted by $y_p \in \mathbb{R}^{m-d}$. Assume y satisfies the physical constraint (2.2.1). Then we can correspondingly partition the vector-valued function F by

$$y_o = F_o(z) + \epsilon_y, \tag{2.3.1}$$

$$y_p = F_p(z). \tag{2.3.2}$$

The mappings $F_o(z)$, $F_p(z)$ are defined by the user at application time, and they are typically instantiated by forward simulation software. Examples of how to create them will be provided in Section 2.5.

We consider the following two scenarios: (1) we have observations only of y in Section 2.3.1, and (2) we have additional observations of the latent process z in Section 2.3.2. The name of the resulting tasks, latent process kriging (1) and joint process kriging (2), originate in [57] when applied to the explicit covariance model. We now describe them, and we evaluate their computational effort as a combination of the number of forward model evaluations required by the derivations in Section 2.2.3, the number of system solves with the covariance matrices defining our model (2.2.2), and the (dominant) computational effort of the remaining model forming tasks.

2.3.1 Latent Process Kriging

By the physics-based covariance model (2.2.4), the joint distribution of y_o and y_p is approximated by

$$\mathcal{M}_1 : \begin{pmatrix} y_o \\ y_p \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} m(y_o) \\ m(y_p) \end{pmatrix}, \begin{pmatrix} L_o \mathrm{Cov}(z,z) L_o^T + K_{\epsilon_y} & L_o \mathrm{Cov}(z,z) L_p^T \\ L_p \mathrm{Cov}(z,z) L_o^T & L_p \mathrm{Cov}(z,z) L_p^T \end{pmatrix} \right), \tag{2.3.3}$$

where L_o is the Jacobian matrix of F_o evaluated at $\bar{z}$ and L_p is the Jacobian matrix of F_p evaluated at $\bar{z}$. The corresponding GP regression problem solves the posterior distribution under observations. The predicted posterior mean is given by

$$m(y_p) + (L_p\text{Cov}(z,z)L_o^T)(K_{\epsilon_y} + L_o\text{Cov}(z,z)L_o^T)^{-1}(y_o - m(y_o)). \tag{2.3.4}$$

The difficulty of computing the expression is that $(K_{\epsilon_y} + L_o\text{Cov}(z,z)L_o^T)$ is a $d \times d$ matrix. Solving the inverse by the conventional Cholesky decomposition method requires a computational cost of order $O(d^3)$. Since we assume K_{ϵ_y} possesses sparsity and can be efficiently inverted, the Sherman-Morrison-Woodbury (SMW) formula can then be applied, and the computational cost of solving the inverse is then determined by the rank of the $L_o\text{Cov}(z,z)L_o^T$ term, in other words, $\max(d,n)$. For simplicity of our discussion, we assume $d = O(m)$, which models many circumstances where the dimension of the vector to be forecast is no larger in order than the number of observation points required. Then the computational cost can be reduced by the proposed low-rank approximation, which replaces $L_o\text{Cov}(z,z)L_o^T$ by a rank N approximation. We illustrate the scalable approach for solving the GP regression posterior mean with a computational cost analysis of each step in Algorithm 1.

In summary, based on our assumption of observation noise covariance K_{ϵ_y}, the entire approach takes $O(N)$ forward solves and $O(N)$ K_{ϵ_y} linear system solves, and the dominant computational cost takes $O(mN^2+nN)$ time, although the workflow is highly parallelizable. The cost is quasilinear with n if $N = log(n)$.

2.3.2 Joint Process Kriging

In this subsection, we consider the case where we can partially observe the latent process. The additional information about the latent process helps us get more accurate predictions of the unobserved output. Denote the observed part of the latent process by $z_o \in \mathbb{R}^D$ and the unobserved part of latent process by $z_p \in \mathbb{R}^{n-D}$. The observation noise covariance is

Algorithm 1: Latent Process Kriging $\mathcal{M}_1$

Use the low-rank approximation discussed in Section 2.2.2 with $N = O(\log n)$, and then solve the approximate GP regression,

$$m(y_p) + (L_p C_z^T K C_z L_o^T)(K_{\epsilon_y} + L_o C_z^T K C_z L_o^T)^{-1}(y_o - m(y_o)). \tag{2.3.5}$$

Step 1. Compute $A_1 = L_o C_z^T \in \mathbb{R}^{d\times N}$, $A_2 = L_p C_z^T \in \mathbb{R}^{(m-d)\times N}$ by $O(N)$ forward solves.

Step 2. Solve the approximated inverse problem by the SMW formula.

$$\begin{aligned}&(K_{\epsilon_y} + A_1 K A_1^T)^{-1}(y_o - m(y_o))\\ =&K_{\epsilon_y}^{-1}(y_o - m(y_o)) - K_{\epsilon_y}^{-1} A_1 (K^{-1} + A_1^T K_{\epsilon_y}^{-1} A_1)^{-1} A_1^T K_{\epsilon_y}^{-1}(y_o - m(y_o)).\end{aligned} \tag{2.3.6}$$

a. Compute $\alpha = A_1^T K_{\epsilon_y}^{-1}(y_o - m(y_o))$. It takes $O(mN)$ time and one K_{ϵ_y} linear system solve.

b. Solve $(K^{-1} + A_1^T K_{\epsilon_y}^{-1} A_1)^{-1}\alpha$ by

$$\begin{pmatrix} I_N & A_1^T K_{\epsilon_y}^{-1} A_1 \\ K & -I_N \end{pmatrix}\begin{pmatrix}\beta\\ \gamma\end{pmatrix} = \begin{pmatrix}\alpha \\ 0\end{pmatrix}, \tag{2.3.7}$$

where forming $A_1^T K_{\epsilon_y}^{-1} A_1$ takes $O(N)$ K_{ϵ_y} linear system solves and $O(mN^2)$ time. Solving the system takes $O(N^3)$ time.

c. The solution is given by $y = K_{\epsilon_y}^{-1}(y_o - m(y_o)) - K_{\epsilon_y}^{-1} A_1 \gamma$. This takes $O(mN)$ time and two K_{ϵ_y} linear system solves.

Step 3. **Return** $m(y_p) + (A_2 K A_1^T) y$ by matrix-vector multiplication. This takes $O(2mN + N^2)$ time.

given by K_{ϵ_z}. As discussed after (2.2.2), we assume K_{ϵ_z} is sparse and can be efficiently inverted with at most quasilinear time cost. For estimating the computational effort we assume $D = O(n)$. From our physics-based covariance model (2.2.4), the joint distribution of y_o, z_o, y_p and z_p is given by

$$\mathcal{M}_2 : \begin{pmatrix} y_o \\ z_o \\ y_p \\ z_p \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} m(y_o) \\ m(z_o) \\ m(y_p) \\ m(z_p)\end{pmatrix}, \begin{pmatrix} K_{11} & K_{12} \\ K_{21} & K_{22}\end{pmatrix}\right), \tag{2.3.8}$$

where

$$K_{11} = \begin{pmatrix} K_{\epsilon_y} + L_o \mathrm{Cov}(z,z) L_o^T & L_o \mathrm{Cov}(z, z_o) \\ \mathrm{Cov}(z_o, z) L_o^T & K_{\epsilon_z} + \mathrm{Cov}(z_o, z_o) \end{pmatrix}, \tag{2.3.9}$$

$$K_{21} = \begin{pmatrix} L_p \mathrm{Cov}(z,z) L_o^T & L_p \mathrm{Cov}(z, z_o) \\ \mathrm{Cov}(z_p, z) L_o^T & \mathrm{Cov}(z_p, z_o) \end{pmatrix}. \tag{2.3.10}$$

The corresponding GP regression problem solves the predicted posterior mean,

$$\begin{pmatrix} m(y_p) \\ m(z_p) \end{pmatrix} + (K_{21})(K_{11})^{-1} \begin{pmatrix} y_o - m(y_o) \\ z_o - m(z_o) \end{pmatrix}. \tag{2.3.11}$$

The partitioned covariance matrices $\mathrm{Cov}(z_o, z)$, $\mathrm{Cov}(z_p, z)$, and $\mathrm{Cov}(z_p, z_o)$ are all submatrices of $\mathrm{Cov}(z, z)$ formed by selecting the corresponding rows or columns. Therefore, we can construct $C_{z_o} \in \mathbb{R}^{N\times D}$ and $C_{z_p} \in \mathbb{R}^{N\times(n-D)}$ by selecting the corresponding columns of the interpolation matrix C_z and creating suitable low-rank approximations of these covariance matrices. A scalable approach for solving the GP regression posterior mean problem is summarized in Algorithm 2.

In summary, based on our assumption of observation noise covariance matrices, the entire approach takes $O(N)$ forward solves and $O(N)$ observation noise covariance matrix linear system solves, and the dominant computational cost takes $O((m + n)N^2)$ time. The cost is quasilinear with n if $N = log(n)$.

2.3.3 Correction for Nonlinearity

In the preceding two subsections, we described scalable approaches for solving the GP regression problem with a physics-based second-order covariance model (2.2.4). For highly

---

Algorithm 2: Joint Process Kriging $\mathcal{M}_2$

Use the low-rank approximation index $N = O(\log n)$. Approximate the latent process covariances by interpolation as $\mathrm{Cov}(z_o, z_o) \approx C_{z_o}^T K C_{z_o}$, $\mathrm{Cov}(z_o, z) \approx C_{z_o}^T K C_z$, $\mathrm{Cov}(z_p, z) \approx C_{z_p}^T K C_z$, $\mathrm{Cov}(z_p, z_o) \approx C_{z_p}^T K C_{z_o}$.

Step 1. Compute $A_1 = L_o C_z^T \in \mathbb{R}^{d \times N}$, $A_2 = L_p C_z^T \in \mathbb{R}^{(m-d) \times N}$ by $O(N)$ forward solves.

Step 2. Solve the approximated inverse problem by the SMW formula,

$$\begin{aligned}\alpha =&(K_{11})^{-1}\begin{pmatrix} y_o - m(y_o) \\ z_o - m(z_o)\end{pmatrix}\\ =&\left(\begin{pmatrix} K_{\epsilon_y} & 0 \\ 0 & K_{\epsilon_z}\end{pmatrix} + \begin{pmatrix} A_1 & 0 \\ 0 & C_{z_o}^T\end{pmatrix}\begin{pmatrix} K & K \\ K & K\end{pmatrix}\begin{pmatrix} A_1^T & 0 \\ 0 & C_{z_o}\end{pmatrix}\right)^{-1}\begin{pmatrix} y_o - m(y_o) \\ z_o - m(z_o)\end{pmatrix}.\end{aligned} \tag{2.3.12}$$

This can be efficiently carried out since $\begin{pmatrix} K & K \\ K & K\end{pmatrix} \in \mathbb{R}^{2N \times 2N}$ has dimension much smaller than n and $\begin{pmatrix} K_{\epsilon_y} & 0 \\ 0 & K_{\epsilon_z}\end{pmatrix}$ can be efficiently inverted similarly to Algorithm 1. The time cost is dominated by forming matrix products $(A_1^T K_{\epsilon_y}^{-1} A_1)$ and $(C_{z_o} K_{\epsilon_z}^{-1} C_{z_o}^T)$, which takes $O((n+m)N^2)$ time and $O(N)$ linear system solves with K_{ϵ_y} and K_{ϵ_z}.

Step 3. **Return** the final solution,

$$\begin{pmatrix} m(y_p) \\ m(z_p)\end{pmatrix} + \left(\begin{pmatrix} A_2 & 0 \\ 0 & C_{z_p}^T\end{pmatrix}\begin{pmatrix} K & K \\ K & K\end{pmatrix}\begin{pmatrix} A_1^T & 0 \\ 0 & C_{z_o}\end{pmatrix}\right)\alpha. \tag{2.3.13}$$

The step takes $O(2(n+m)N + 4N^2)$ time.

---

nonlinear functions, we expect that using higher-order covariance models (2.2.5) will provide a more accurate covariance matrix structure. In this subsection, we discuss how to include the higher-order terms in the GP regression problem while maintaining quasilinear time scaling via approximations. To this end, we apply the formalism (2.2.6) to the partitioned case where $y \to (y_o^T, y_p^T)^T$. We denote the Hessian tensor of F_o, F_p evaluated at $\bar{z}$ by H_o, H_p, respectively. From (2.2.6), one important observation is that the higher-order terms occur only in the auto-covariance of y. Our modified latent process kriging with higher-order terms then has the following expression:

$$m(y_p)+\left(L_p\mathrm{Cov}(z,z)L_o^T+\frac{1}{4}\overline{\delta z^T H_p\delta z\delta z^T H_o^T\delta z}-\frac{1}{4}\overline{\delta z^T H_p\delta z}\;\overline{\delta z^T H_o^T\delta z}\right)$$
$$\left(K_{\epsilon_y}+L_o\mathrm{Cov}(z,z)L_o^T+\frac{1}{4}\overline{\delta z^T H_o\delta z\delta z^T H_o^T\delta z}-\frac{1}{4}\overline{\delta z^T H_o\delta z}\;\overline{\delta z^T H_o^T\delta z}\right)^{-1}$$
$$(y_o-m(y_o)). \tag{2.3.14}$$

Likewise, for joint process kriging with higher-order terms, the estimator becomes

$$\begin{pmatrix} m(y_p) \\ m(z_p) \end{pmatrix} + (K_{21}^+)(K_{11}^+)^{-1}\begin{pmatrix} y_o-m(y_o) \\ z_o-m(z_o) \end{pmatrix}, \tag{2.3.15}$$

where

$$K_{11}^+=\begin{pmatrix} L_o\mathrm{Cov}(z,z)L_o^T+\frac{1}{4}\overline{\delta z^T H_o\delta z\delta z^T H_o^T\delta z}-\frac{1}{4}\overline{\delta z^T H_o\delta z}\;\overline{\delta z^T H_o^T\delta z} & L_o\mathrm{Cov}(z,z_o) \\ \mathrm{Cov}(z_o,z)L_o^T & \mathrm{Cov}(z_o,z_o) \end{pmatrix}$$
$$+\begin{pmatrix} K_{\epsilon_y} & 0 \\ 0 & K_{\epsilon_z} \end{pmatrix}, \tag{2.3.16}$$

$$K_{21}^+=\begin{pmatrix} L_p\mathrm{Cov}(z,z)L_o^T+\frac{1}{4}\overline{\delta z^T H_p\delta z\delta z^T H_o^T\delta z}-\frac{1}{4}\overline{\delta z^T H_p\delta z}\;\overline{\delta z^T H_o^T\delta z} & L_p\mathrm{Cov}(z,z_o) \\ \mathrm{Cov}(z_p,z)L_o^T & \mathrm{Cov}(z_p,z_o) \end{pmatrix}. \tag{2.3.17}$$

We now focus on computing the two higher-order terms appearing in auto-covariance. Denote

$$U(H,\hat{H})=\frac{1}{4}\overline{\delta z^T H\delta z\delta z^T\hat{H}^T\delta z}, \tag{2.3.18}$$
$$V(H,\hat{H})=-\frac{1}{4}\overline{\delta z^T H\delta z}\;\overline{\delta z^T\hat{H}^T\delta z}, \tag{2.3.19}$$

where H and $\hat{H}$ both take the value of either H_o or H_p, depending on the covariance components. For multivariate Gaussian random variable δz, we have that

$$
\begin{aligned}
U(H,\hat{H}) + V(H,\hat{H}) \overset{(\text{A.3.9}),(\text{A.3.11})}{=} & \frac{1}{4}\overline{\delta z^T H \delta z \delta z^T \hat{H}^T \delta z} - \frac{1}{4}\overline{\delta z^T H \delta z}\ \overline{\delta z^T \hat{H}^T \delta z} \\
= & \frac{1}{2}\operatorname{tr}(H\operatorname{Cov}(z,z)\hat{H}^T\operatorname{Cov}(z,z)) + \frac{1}{4}\operatorname{tr}(H\operatorname{Cov}(z,z))\operatorname{tr}(\hat{H}\operatorname{Cov}(z,z))^T \\
& - \frac{1}{4}\operatorname{tr}(H\operatorname{Cov}(z,z))\operatorname{tr}(\hat{H}\operatorname{Cov}(z,z))^T \\
= & \frac{1}{2}\operatorname{tr}(H\operatorname{Cov}(z,z)\hat{H}^T\operatorname{Cov}(z,z)). \qquad (2.3.20)
\end{aligned}
$$

Details of the algebra can be found in A.3. We now turn to the issue of approximating the nonlinear corrections efficiently. We will again use the low-rank approximation $\operatorname{Cov}(z,z) \approx C_z^T K C_z$ with K symmetric and positive semi-definite. Then there exists a Cholesky factorization of K such that $K = P^T P$, where P is an upper triangular matrix. Since K is an $N \times N$ matrix, with $N \ll n$, the cost of the factorization can be ignored compared with quasilinear scaling in n. Our approximation becomes

$$
\begin{aligned}
\operatorname{tr}(H\operatorname{Cov}(\mathrm{z},\mathrm{z})\hat{H}^T\operatorname{Cov}(\mathrm{z},\mathrm{z})) &\approx \operatorname{tr}(HC_z^T P^T P C_z \hat{H}^T C_z^T P^T P C_z) \\
&= \operatorname{tr}(PC_z H C_z^T P^T P C_z \hat{H}^T C_z^T P^T) \\
&= \operatorname{tr}(M\hat{M}^T), \qquad (2.3.21)
\end{aligned}
$$

where $M = PC_z H C_z^T P^T$ and $\hat{M} = PC_z \hat{H} C_z^T P^T$ are rank-three tensors, whose size depends on the choice of Hessian tensors $H, \hat{H}$. We further denote $M_o = PC_z H_o C_z^T P^T$, $M_p = PC_z H_p C_z^T P^T$ corresponding to the two choices of Hessians. Let $\{u_i\}, \{v_i\}$ be N-dimensional independent Rademacher vectors (random vectors with independent identical Bernoulli distributed components that take the values $+1$ and -1 with probability 0.5).

Then

$$
\begin{aligned}
\operatorname{tr}(M\hat{M}^T) &= \operatorname{tr}(MI\hat{M}^T I) \\
&= \operatorname{tr}(M\mathrm{E}(uu^T)\hat{M}^T\mathrm{E}(vv^T)) \\
&= \mathrm{E}(\operatorname{tr}(Muu^T\hat{M}^T vv^T)) \\
&= \mathrm{E}(v^T Muu^T\hat{M}^T v) \\
&\approx \frac{1}{N_l}\sum_{i=1}^{N_l}(v_i^T Mu_i)(v_i^T\hat{M}u_i)^T. \qquad (2.3.22)
\end{aligned}
$$

Note that $v_i^T Mu_i \in \mathbf{R}^{m_2}$, where m_2 is the second dimension of M and $v_i^T Mu_i \in \mathbf{R}^{\hat{m}_2}$, where $\hat{m}_2$ is the second dimension of $\hat{M}$. Therefore $(v_i^T Mu_i)(v_i^T\hat{M}u_i)^T$ is a rank-1 $m_2 \times \hat{m}_2$ matrix. The entire stochastic trace estimator (2.3.22) is a matrix of rank-N_l. When $N_l = O(\log n)$, adding the approximated trace term to the auto-covariance matrices becomes an additional low-rank perturbation that can also be handled efficiently by the SMW formula. While there is a concern that N_l may be too small to obtain a high-quality approximation, we note that for well-conditioned blocks $M(:, i, :)$, for $i = 1, 2, \ldots, m_2$ (in Matlab notation), the approximation has provably excellent quality [177]. Since a low-rank approximation of a covariance matrix has a much better condition number than the original, the approach is in the regime of validity from [177].

We illustrate the computational workflow for including nonlinear higher-order terms in the covariance computation and solving the modified GP regression problem scalably in Algorithm 3. A similar workflow for the joint process kriging is in A.1.

In summary, based on our assumption of observation noise covariance matrices, the entire approach takes $O(N+N_l)$ forward solves and $O(N+N_l)$ observation noise covariance matrix linear system solves, and the dominant remaining computational cost takes $O(m(N+N_l)^2)$ time. The cost is quasilinear with n if $N, N_l = log(n)$.

2.4 A Scalable Approach for Gaussian Process Maximum Likelihood Estimation

The proposed physics-based covariance model (2.2.3)(2.2.4) provides explicit covariance structure based on the physical operators and the covariance of the latent process $\mathrm{Cov}(z, z)$. In most applications, a reasonable covariance function of the latent process can be proposed based on past observations or prior knowledge up to some parameters that need to be estimated from data. While in many circumstances, the results of kriging are only weakly dependent on the covariance parameters, this may be not known in advance, and good guidance on how to choose these parameters may not exist, particularly if one does the analysis for the first time. It is useful to give the user the tools to estimate such parameters, which can be seen to be conceptually similar to the point of view of empirical Bayes estimation [43], whereby one marginalizes over the latent variables and first estimates the remaining model parameters from data

Since z is a random field at locations $(x_1, \cdots, x_n)$, assume the covariance matrix $\mathrm{Cov}(z, z)$ can be parameterized by a valid covariance function $k(\cdot, \cdot; \theta)$ depending on unknown parameters $\theta \in \mathbb{R}^r$. Then

$$\mathrm{Cov}(z(x_i), z(x_j); \theta) = k(x_i, x_j; \theta),\ i, j = 1, 2, \cdots, n. \tag{2.4.1}$$

We denote the resulting covariance matrix by $\mathrm{Cov}(z, z; \theta) \in \mathbb{R}^{n \times n}$. The most principled method to estimate the parameters θ is the maximum likelihood estimator (MLE). Following the same setup we distinguish two cases: (1) a latent process MLE, where we have available observations only of output process y and (2) a joint process MLE, where we can observe both processes y and z, which we describe below. Notionally, the same estimation workflow could apply to the nonlinear corrections from Section 2.3.3, but the estimates can no longer be justified by output likelihood considerations, since in the case of nonlinear transformation the output y does not have a Gaussian distribution in general. We thus discuss here only the linear dependence case. Moreover, y_p does not play a role in estimating θ, so we will focus on the joint relationship of y_o and z only.

2.4.1 Latent Process MLE

Given partial observations $y_o \in \mathbb{R}^d$ and based on the covariance model for the latent process (2.3.3), the log-likelihood function (up to an additive constant) is given by

$$\begin{aligned}\log p(y_o|\theta) =& -\frac{1}{2}\log\det\left(K_{\epsilon_y} + L_o\mathrm{Cov}(z,z;\theta)L_o^T\right)\\ &-\frac{1}{2}(y_o-\overline{y_o})^T(K_{\epsilon_y} + L_o\mathrm{Cov}(z,z;\theta)L_o^T)^{-1}(y_o-\overline{y_o}).\end{aligned} \tag{2.4.2}$$

One can avoid the log-determinant computation—which requires a Cholesky factorization of a dense and large matrix, a very expensive computation—by considering the score equations, which are obtained by setting the gradient of log-likelihood to be zero. The gradient components are

$$\begin{aligned}g_i(\theta) =&\frac{1}{2}(y_o-\overline{y_o})^T(K_{\epsilon_y} + L_o\mathrm{Cov}(z,z;\theta)L_o^T)^{-1}(L_o\mathrm{Cov}_i(z,z;\theta)L_o^T)(K_{\epsilon_y} + L_o\mathrm{Cov}(z,z;\theta)L_o^T)^{-1}\\ &(y_o-\overline{y_o}) - \frac{1}{2}\mathrm{tr}((K_{\epsilon_y} + L_o\mathrm{Cov}(z,z;\theta)L_o^T)^{-1}(L_o\mathrm{Cov}_i(z,z;\theta)L_o^T)),\end{aligned} \tag{2.4.3}$$

where $\mathrm{Cov}_i(z,z;\theta) = \frac{\partial}{\partial\theta_i}\mathrm{Cov}(z,z;\theta)$. The score equations become $g_i(\theta)=0$, $i=1,2,\ldots,r$. To solve this problem, one needs to be able to efficiently evaluate the score equation for any θ. When creating low-rank approximations, we exploit the fact that the Chebyshev nodes and coefficients do not depend on the parameters θ. Using the low-rank approximation discussed in Section 2.2.2, we can define the following approximation:

$$\mathrm{Cov}(z, z; \theta) \approx C_z^T K(\theta) C_z, \tag{2.4.4}$$

$$\frac{\partial}{\partial \theta_i} \mathrm{Cov}(z, z; \theta) \approx C_z^T K_i(\theta) C_z, \quad i = 1, 2, \ldots, r, \tag{2.4.5}$$

where $K(\theta)$ is the compressed covariance matrix, explicitly depending on θ and $K_i(\theta)$ its derivative with θ_i. Because of the explicit nature of $K(\theta)$ in an interpolation approach, $K_i(\theta)$ is readily computable. Therefore, the two terms in the gradient component expression (2.4.3) can be approximated separately. For the first term, we apply the SMW formula to sequentially solve the matrix-vector product. For the second term, we use stochastic trace estimator to convert the expensive trace operations to vector-matrix-vector product [177]. A scalable approach for solving the latent process MLE problem is summarized in Algorithm 4.

In summary, the system of score equations takes $O(N)$ forward solves, $O(N)$ linear system solves with the noise covariance matrix K_{ϵ_y}, and an assembly effort of $O(rN^2 N_l + mN^2 + mNN_l)$. Each optimization iteration has quasilinear time complexity with n if $N, N_l = log(n)$.

2.4.2 Joint Process MLE

Given partial observations $y_o \in \mathbb{R}^d$, additional observations of the latent process $z_o \in \mathbb{R}^D$ and based on the covariance model for the joint process (2.3.8), the log-likelihood function of the observations (up to an additive constant) is given by

$$\log p(y_o, z_o | \theta) = -\frac{1}{2} \log \det K_{11} - \frac{1}{2} \begin{pmatrix} y_o - \overline{y_o} \\ z_o - \overline{z_o} \end{pmatrix}^T K_{11} \begin{pmatrix} y_o - \overline{y_o} \\ z_o - \overline{z_o} \end{pmatrix}, \tag{2.4.11}$$

$$K_{11} = \begin{pmatrix} K_{\epsilon_y} + L_o \mathrm{Cov}(z, z; \theta) L_o^T & L_o \mathrm{Cov}(z, z_o; \theta) \\ \mathrm{Cov}(z_o, z; \theta) L_o^T & K_{\epsilon_z} + \mathrm{Cov}(z_o, z_o; \theta) \end{pmatrix}. \tag{2.4.12}$$

The log-likelihood gradient components are given by

$$\begin{aligned} g_i(\theta) =& \frac{1}{2} \begin{pmatrix} y_o - \overline{y_o} \\ z_o - \overline{z_o} \end{pmatrix}^T K_{11}^{-1} \begin{pmatrix} L_o \mathrm{Cov}_i(z, z; \theta) L_o^T & L_o \mathrm{Cov}_i(z, z_o; \theta) \\ \mathrm{Cov}_i(z_o, z; \theta) L_o^T & \mathrm{Cov}_i(z_o, z_o; \theta) \end{pmatrix} K_{11}^{-1} \begin{pmatrix} y_o - \overline{y_o} \\ z_o - \overline{z_o} \end{pmatrix} \\ & - \frac{1}{2} \mathrm{tr} \left(K_{11}^{-1} \begin{pmatrix} L_o \mathrm{Cov}_i(z, z; \theta) L_o^T & L_o \mathrm{Cov}_i(z, z_o; \theta) \\ \mathrm{Cov}_i(z_o, z; \theta) L_o^T & \mathrm{Cov}_i(z_o, z_o; \theta) \end{pmatrix} \right). \end{aligned} \tag{2.4.13}$$

Subsequently, the score equations are defined by $g_i(\theta) = 0$, $i = 1, 2, \ldots, r$.

As in the rest of the paper, we efficiently approximate the components $g_i(\theta)$ by means of low-rank approximations of $\mathrm{Cov}(z, z_o; \theta)$ and $\mathrm{Cov}(z_o, z_o; \theta)$, as well as their derivatives with respect to θ_i (denoted here with the subscript i) by interpolation. A scalable approach for solving the joint process MLE problem by score equations is summarized in Algorithm 5.

In summary, computing the gradient components for the score equations takes $O(N)$ forward solves, $O(N)$ linear system solves with the noise covariance matrix, and $O(rN_lN^2 + (m+n)NN_l + (m+n)N^2)$ operations for the assembly of the various matrices. For any θ, computing these gradient components has a quasilinear time complexity with n if $N, N_l = log(n)$.

2.4.3 Estimation of Fisher Information Matrix

The expected Fisher information matrix is commonly used in providing confidence intervals on the estimates of θ by means of asymptotic analysis. Specifically, denote the solution of the score equations by $\hat{\theta}$ and the Fisher information matrix by $\mathcal{I}(\hat{\theta})$. Then, as the sample size goes to infinity, one expects that

$$(\hat{\theta} - \theta_{\text{true}}) \xrightarrow{d} \mathcal{N}(0, \mathcal{I}(\hat{\theta})^{-1}). \tag{2.4.18}$$

Therefore, for each component of θ, the confidence interval can be approximated by

$$\hat{\theta}_i \pm c\sqrt{(\mathcal{I}(\hat{\theta})^{-1})_{ii}}, \tag{2.4.19}$$

where c is the appropriate critical value. Furthermore, for multivariate Gaussian distribution where the uncertainties occur only in the covariance, each entry in the expected Fisher information matrix is given by [177]

$$\mathcal{I}_{i,j}(\theta) = \frac{1}{2}\text{tr}(\Sigma(\theta)^{-1}\Sigma_i(\theta)\Sigma(\theta)^{-1}\Sigma_j(\theta)), \tag{2.4.20}$$

where $\Sigma(\theta)$ denotes the autocovariance matrix in latent or joint process MLE and $\Sigma_i(\theta) = \frac{\partial}{\partial\theta_i}\Sigma(\theta)$. The trace term can be again approximated by the stochastic trace estimator

$$\hat{\mathcal{I}}_{i,j}(\theta) \approx \frac{1}{2N_l}\sum_{l=1}^{N_l} u_l^T\Sigma(\theta)^{-1}\Sigma_i(\theta)\Sigma(\theta)^{-1}\Sigma_j(\theta)u_l, \tag{2.4.21}$$

where u_l are N_l independent Rademacher random vectors $u_l \in \mathbb{R}^d, l = 1, 2, \ldots, N_l$. The covariance components in (2.4.21) are computed by the same low-rank approximations as in the rest of the paper. We recall from Algorithm 4 that $A_1 = L_oC_z^T$ is computed when setting up the score equations. From (2.4.21) it follows that

$$\begin{aligned}\hat{\mathcal{I}}_{i,j}(\theta) \approx & \frac{1}{2N_l}\sum_{l=1}^{N_l} u_l^T (K_{\epsilon_y} + L_o\text{Cov}(z,z;\theta)L_o^T)^{-1}(L_o\text{Cov}_i(z,z;\theta)L_o^T) \\ & (K_{\epsilon_y} + L_o\text{Cov}(z,z;\theta)L_o^T)^{-1}(L_o\text{Cov}_j(z,z;\theta)L_o^T)u_l, \\ \approx & \frac{1}{2N_l}\sum_{l=1}^{N_l} u_l^T (K_{\epsilon_y} + A_1K(\theta)A_1^T)^{-1}(A_1K_i(\theta)A_1^T) \\ & (K_{\epsilon_y} + A_1K(\theta)A_1^T)^{-1}(A_1K_j(\theta)A_1^T)u_l. \end{aligned} \tag{2.4.22}$$

Observing the structure of the term, we will evaluate the Fisher information matrix in the following sequence. First we compute $W = (K_{\epsilon_y} + A_1K(\theta)A_1^T)^{-1}A_1$ by the SMW formula. Note that the W component for SMW needs to be assembled only once. This step takes $O(N)$ K_{ϵ_y} linear system solves and $O(mN^2)$ time. Then forming the $N \times N$ matrix $U = A_1^TW$ takes $O(mN^2)$ time. After that we compute the right terms $\{A_1^Tu_l\}$, $l = 1, 2, \cdots, N_l$ with $O(mNN_l)$ cost and the left terms $\{w_l\}$, where $w_l = W^Tu_l$, $l = 1, 2, \cdots, N_l$ with $O(mNN_l)$ cost. Now we finish our precomputing process with a cost independent from r. Then we assemble the entries in the Fisher information matrix. For each i, j, we compute N_l terms to approximate the trace. Each term takes the form $w_l^TK_i(\theta)UK_j(\theta)A_1^Tu_l$ with $O(N^2)$ cost. Since we have r^2 entries in the Fisher information matrix, the total assembling cost is $O(r^2N_lN^2)$.

To summarize, the whole Fisher information matrix takes $O(N)$ K_{ϵ_y} linear system solves and an assembly cost of $O(mN(N + N_l) + r^2N_lN^2)$ time to compute. The cost of inverting the expected Fisher information matrix can be ignored. Similarly, for computing the Fischer matrix in the joint process model, we draw N_l independent Rademacher vector $\{v_l\} \in \mathbb{R}^{d+D}$, $l = 1, 2, \ldots, N_l$. Using the low-rank approach, we get the following approximation for its entries.

$$\hat{\mathcal{I}}_{i,j}(\theta) \approx \frac{1}{2N_l}\sum_{l=1}^{N_l} v_l^T \left(\begin{pmatrix} K_{\epsilon_y} & 0 \\ 0 & K_{\epsilon_z}\end{pmatrix} + \begin{pmatrix} A_1 & 0 \\ 0 & C_{z_o}^T\end{pmatrix}\begin{pmatrix} K(\theta) & K(\theta) \\ K(\theta) & K(\theta)\end{pmatrix}\begin{pmatrix} A_1^T & 0 \\ 0 & C_{z_o}\end{pmatrix}\right)^{-1}$$
$$\begin{pmatrix} A_1 & 0 \\ 0 & C_{z_o}^T\end{pmatrix}\begin{pmatrix} K_i(\theta) & K_i(\theta) \\ K_i(\theta) & K_i(\theta)\end{pmatrix}\begin{pmatrix} A_1^T & 0 \\ 0 & C_{z_o}\end{pmatrix}\left(\begin{pmatrix} K_{\epsilon_y} & 0 \\ 0 & K_{\epsilon_z}\end{pmatrix} + \begin{pmatrix} A_1 & 0 \\ 0 & C_{z_o}^T\end{pmatrix}\right.$$
$$\left.\begin{pmatrix} K(\theta) & K(\theta) \\ K(\theta) & K(\theta)\end{pmatrix}\begin{pmatrix} A_1^T & 0 \\ 0 & C_{z_o}\end{pmatrix}\right)^{-1}\begin{pmatrix} A_1 & 0 \\ 0 & C_{z_o}^T\end{pmatrix}\begin{pmatrix} K_j(\theta) & K_j(\theta) \\ K_j(\theta) & K_j(\theta)\end{pmatrix}\begin{pmatrix} A_1^T & 0 \\ 0 & C_{z_o}\end{pmatrix} v_l \tag{2.4.23}$$

The computational cost analysis is similar to the one for the Fischer matrix of the latent process model (2.4.22). The precomputing step takes $O(N)$ linear system solves with the observation noise covariance matrix and $O((n+m)N(N+N_l))$ vector and matrix assembly time. Assembling the Fisher information matrix takes $O(r^2 N_l N^2)$ time. The cost of inverting the expected Fisher information matrix can be ignored. Since the dimension of the parameters is finite, taking $N, N_l = O(\log n)$ provides us a scalable approach to estimating the Fisher information matrix for estimating the parameters of our uncertainty model.

2.5 Numerical Experiments

In this section, we present numerical tests based on both synthetic and realistic datasets. The synthetic case is based on the 1D viscous Burger's equation and the corresponding model-generated data. The realistic case concerns reanalyzed satellite-observed water vapor and wind velocity data. For the synthetic case, we investigated all model and computation features described in this paper, including numerical accuracy of parameter estimations and uncertainty quantification when carrying out low-rank-based reduction, Section 2.2.2, comparisons of the different covariance models Section 2.3.1 and Section 2.3.2, effect of higher-order terms Section 2.3.3, accuracy of the confidence interval estimates when carrying out maximum likelihood Section 2.4.3, and the measured computational scaling of the approximated algorithm. For the real atmospheric dataset, we consider only linear models, and we focus primarily on improvement in kriging accuracy compared with the case when the cross-covariance information is ignored, and we model comparisons between Section 2.3.1 and Section 2.3.2. Since the contribution of this work is in providing a scalable approach for both the kriging and the maximum likelihood tasks for implicitly defined Gaussian process models, we will be less interested in doing an extensive modeling comparison. We will focus on demonstrating that the approximating nature of our method does not significantly alter the predictive power of a physics-based approach to modeling the covariance, which was demonstrated in [57].

2.5.1 One-Dimensional Viscous Burger's Equation

Burger's equation is a fundamental partial differential equation that combines the nonlinear advection and the linear diffusion. It can be investigated in $1+1$ (one spatial dimension and one temporal dimension) dimensional settings and is a standard test case for data assimilation algorithms [213, 9, 90, 112, 189]. The equation takes the following form:

$$\frac{\partial w}{\partial t} + \frac{1}{2}\frac{\partial (w^2)}{\partial x} = \nu \frac{\partial^2 w}{\partial x^2} + f,\ \ (x,t) \in (0,1) \times (0,T). \tag{2.5.1}$$

For assessing the effectiveness of our uncertainty quantification approach we include an additional external forcing term f that is deterministic and time-independent. We select a smaller diffusion coefficient, and we choose more complicated initial conditions to engender more long-lasting chaotic behavior. To numerically solve the equation, we denote the value of its approximation at grid point $(j\Delta x, m\Delta t)$ by w_j^m and of the constant forcing as f_j. We construct a forward solver of the system using the following finite difference scheme:

$$\frac{w_j^{m+1} - w_j^m}{\Delta t} + \frac{(w_{j+1}^m)^2 - (w_{j-1}^m)^2}{4\Delta x} - \frac{\nu}{(\Delta x)^2}(w_{j+1}^{m+1} - 2w_j^{m+1} + w_{j-1}^{m+1}) - f_j = 0. \quad (2.5.2)$$

To apply our framework to this problem, we find the solution of the problem on the two-dimensional grid. Given the initial condition I and boundary condition B, we write $z^T = (B^T, I^T)$. Then we define $w = G(z)$, where w is the solution produced by the numerical scheme (2.5.2). In all our synthetic experiments we take y_o to consist of a subset of the components of w to which we add Gaussian observation noise $\epsilon_y \sim \mathcal{N}(0, \sigma^2 I)$ and y_p to consist of another such subset. We use I to denote the identity matrix with proper size. Similarly, we take z_o to consist of a subset of the latent process z and Gaussian observation noise $\epsilon_z \sim \mathcal{N}(0, \sigma^2 I)$. And we denote another subset by z_p. Note that for simplicity we assume that the Gaussian noise for both the output process y and latent process z has the same variance; this has no bearing on the scalability conclusions. Extending the scenario to unequal variances is straightforward. The mappings F_o, F_p required by our framework (2.3.1)-(2.3.2) will be obtained from the corresponding components of G. Their Jacobian matrices are denoted by L_o, L_p respectively. The actual observation y_o, z_o and prediction components y_p, z_p may change to suit the validation we try to carry out, and we will define them at multiple points in the material below.

To generate the synthetic data, we use the following settings:

$$\nu = 0.01,\ T = 0.1,\ f = \pi \sin(\pi x)(\cos(\pi x) + \nu\pi), \quad (2.5.3)$$

$$I \sim \mathcal{N}(sin(\pi x), \alpha_I \exp(-\frac{r^2}{2l^2})), \quad (2.5.4)$$

$$B \sim \mathcal{N}(0, \alpha_B \exp(-\frac{r^2}{2(lT)^2})), \quad (2.5.5)$$

$$\alpha_I = 0.1,\ \alpha_B = 0.1,\ l = 0.15,\ \sigma = 0.05. \quad (2.5.6)$$

To simplify the notation, we describe the normal processes defining the distributions of I and B in terms of the mean functions and covariance kernels. The distribution used in computations is obtained by sampling them at the chosen mesh points. In the definition of the covariance kernels, r is the distance between two points, and l is the length scale parameter. Note that the above notations are consistent with Section 2.3 and Section 2.4. Denote the covariance of the latent process z as

$$\mathrm{Cov}(z,z) = \begin{pmatrix} \alpha_I \exp(-\frac{r^2}{2l^2}) & 0 \\ 0 & \alpha_B \exp(-\frac{r^2}{2(lT)^2}) \end{pmatrix}. \tag{2.5.7}$$

The latent process joint distribution can be written as the following:

$$\mathcal{M}_1 : \begin{pmatrix} y_o \\ y_p \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} m(y_o) \\ m(y_p) \end{pmatrix}, \begin{pmatrix} L_o \mathrm{Cov}(z,z) L_o^T + \sigma^2 I & L_o \mathrm{Cov}(z,z) L_p^T \\ L_p \mathrm{Cov}(z,z) L_o^T & L_p \mathrm{Cov}(z,z) L_p^T \end{pmatrix}\right), \tag{2.5.8}$$

while the joint process distribution follows

$$\mathcal{M}_2 : \begin{pmatrix} y_o \\ z_o \\ y_p \\ z_p \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} m(y_o) \\ m(z_o) \\ m(y_p) \\ m(z_p) \end{pmatrix}, \begin{pmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{pmatrix}\right), \tag{2.5.9}$$

where

$$K_{11} = \begin{pmatrix} \sigma^2 I + L_o \mathrm{Cov}(z,z) L_o^T & L_o \mathrm{Cov}(z, z_o) \\ \mathrm{Cov}(z_o, z) L_o^T & \sigma^2 I + \mathrm{Cov}(z_o, z_o) \end{pmatrix}, \tag{2.5.10}$$

$$K_{21} = \begin{pmatrix} L_p \mathrm{Cov}(z,z) L_o^T & L_p \mathrm{Cov}(z, z_o) \\ \mathrm{Cov}(z_p, z) L_o^T & \mathrm{Cov}(z_p, z_o) \end{pmatrix}. \tag{2.5.11}$$

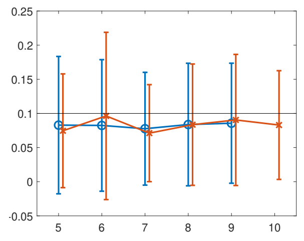
We may vary the mesh size Δx, Δt, but we always maintain $\frac{1}{\Delta x} = \frac{T}{\Delta t}$ so that we have an equal number of mesh points in the two dimensions. We denote the number of mesh points $\frac{1}{\Delta x} = \frac{T}{\Delta t} = k$ then $B \in \mathbb{R}^{2k}$, $I \in \mathbb{R}^k$. The dimension of w in (2.5.2) and thus the potential dimension of y would be $O(k^2)$. For a realistic setup the number of prediction points should be on the order at most comparable with the data size; we thus choose the dimension of y_o and y_p to be $O(k)$. The size of the GP regression problem will thus be growing at most linearly with k.

When carrying out our low-rank approximation, Section 2.2.2, we use $N = 12 \log k$ interpolation points to approximate $\mathrm{Cov}(z, z)$. Also we use $N_l = 20 \log k$ random vectors for stochastic trace estimation when approximating trace terms in Section 2.3.3, Section 2.4.

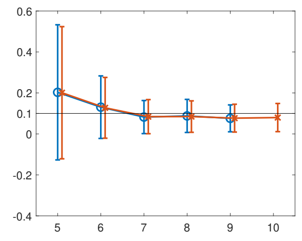
Numerical Accuracy of Parameter Estimation

First we explore the numerical accuracy of estimating the covariance model parameters by our scalable maximum likelihood approach from Section 2.4, inclusive of the computation of the confidence intervals using the expected Fisher information matrix from Section 2.4.3. The parameters here are $\theta = (\alpha_I, \alpha_B, l, \sigma)$, with their true values given by (2.5.6). The data are created by simulating the PDE on $k = 2^{11}$ mesh points in both spatial and temporal dimensions, resulting in a total number of 2^{22} data points in a regular grid. Then all subsequent calculations for obtaining the maximum likelihood estimates and their confidence intervals, including the ones for getting the functions G and F that require the discretization (2.5.2), will be carried out on $k = 2^q$ mesh points in each dimension for q ranging from 5 to 10. This mimics the real situation where the function F is a "black box," but it allows the introduction of an external mesh parameter. For each k, we randomly sample 10 snapshots in time of the output process which results in $d = 10k$ observations for latent process MLE Section 2.4.1 and additional $D = 0.6k$ equally spaced observations of the latent process for joint process MLE Section 2.4.2. We note that a random protocol was also used in [57]. In our experiments we solve the nonlinear score equations (2.4.10) and (2.4.17) by the Matlab `fsolve` function (which uses divided differences approximations of the gradient) with the `trust-region-dogleg` algorithm. The stopping condition is set to be a relative tolerance of 10^{-6}. Finding a good initial point is always difficult in maximum likelihood calculations [177]. Here we precompute the initial guess by exact maximum likelihood estimations using a small subset of available observations. Specifically, we start from the smallest, 2^5, dataset first. For a dataset of this size, we can perform both exact and approximated MLE efficiently. The initial point for optimization is obtained via randomly perturbing the ground truth parameter values from -15% to 15%. For larger datasets of size 2^q for $q \geq 6$, we first draw a subset of 2^5 observations and perform exact MLE starting from this perturbed ground truth. Then the resulting parameter estimators are used as initial guesses in the following parameter fittings with all the available 2^q data.

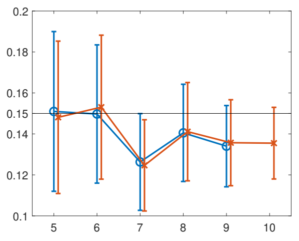
The estimated parameters and corresponding 95% confidence intervals are summarized in Figure 2.5.1 along with the corresponding estimates using the exact likelihood score equations for both the latent model (2.4.3) and the joint model (2.4.13). We note that the *exact* calculation is exact for the Gaussian process model obtained by the *linearization* of F, which is only an approximation of the true likelihood, which is most likely not Gaussian. We observe that the parameter estimates by our approximated approach and exact log-likelihood are fairly close, in terms of both the estimated parameter values and the width of confidence intervals. On the other hand, at large q and k, the exact parameters may be outside the confidence interval, although obviously so only for σ. At that point the error from the linearization exceeds the statistical error. One way to mitigate this (though, also approximate) is to use the nonlinear corrections from Section 2.3.3 for the likelihood, too. It has proven difficult, however, to extend the algebra from Section 2.3.3 for using the derivative of the covariance as well at this time, and we will pursue that direction in future work. From Figure 2.5.1, we note that the approximate likelihood calculations produce results close to the exact likelihood, even if both models are incorrect due to the nonlinearity. Moreover, as we show in the subsequent sections, the parameter estimates we produce with this approximate (and biased) likelihood calculation do improve the GP regression estimates when plugged into the covariance expression. We also note that the parameter error is within 5% for all cases except $q = 5$ in FIG. 2.5.1 (g).



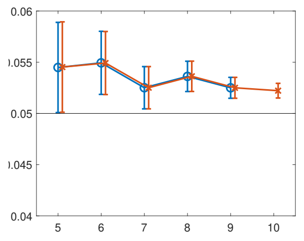
(a) α_I Latent



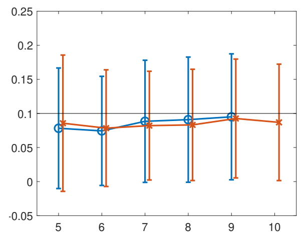
(b) α_B Latent



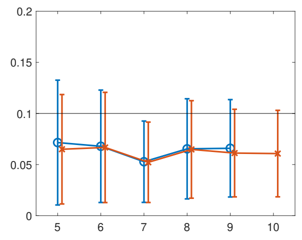
(c) l Latent



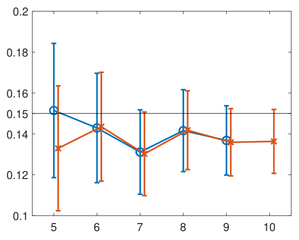
(d) σ Latent



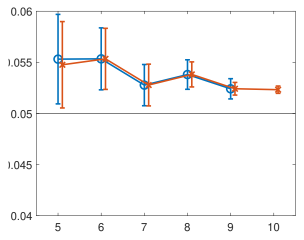
(e) α_I Joint



(f) α_B Joint



(g) l Joint



(h) σ Joint

Figure 2.5.1: Parameter estimations and corresponding 95% confidence intervals computed via the expected Fisher information matrix. We solve the nonlinear system of score equations (2.4.3),(2.4.13) using the physics-based covariance model with the number of mesh points $k = 2^q$ in each dimension for q ranging from 5 to 10 (x-axis). Red lines with crosses represent MLE using low-rank approximations. The first row describes the results for latent process MLE Section 2.4.1. The second row describes the results for joint process MLE Section 2.4.2. MLE results using exact score equations are provided for $q = 5$ to 9 (x-axis) by using blue lines with circles. For $q = 10$ the exact MLE calculation ran out of memory.

Comparison of Covariance Models

We now examine the predictive performance of the physics-based covariance model Section 2.2, inclusive of the low-rank Section 2.2.2 and implicit formulation approximation Section 2.2.3 effects. We compare the latent process kriging Section 2.3.1 and joint process kriging Section 2.3.2 with independent process kriging, which ignores the relations between the two processes by removing the cross-covariance. The latter setup aims to exhibit the features of the most prevalent approaches in modeling where a difficulty of expressing valid cross-covariances models results in the need for modeling different observables independently.

Independent kriging. In this setting we separate the output and latent processes by considering their covariances independently and setting the cross-covariance to zero. We use the following covariance structure:

$$K_{11} = \begin{pmatrix} \mathrm{Cov}(y_o, y_o) + \sigma^2 I_d & 0 \\ 0 & \mathrm{Cov}(z_o, z_o) + \sigma^2 I_D \end{pmatrix}, \tag{2.5.12}$$

$$K_{21} = \begin{pmatrix} \mathrm{Cov}(y_p, y_o) & 0 \\ 0 & \mathrm{Cov}(z_p, z_o) \end{pmatrix}, \tag{2.5.13}$$

$$\mathcal{M}_{\text{ind}} = \begin{pmatrix} m(y_p) \\ m(z_p) \end{pmatrix} + (K_{21})(K_{11})^{-1} \begin{pmatrix} y_o - m(y_o) \\ z_o - m(z_o) \end{pmatrix}. \tag{2.5.14}$$

At this stage the ideal comparison would be with a well-chosen Gaussian kernel to model $\mathrm{Cov}(y_o, y_o)$ and $\mathrm{Cov}(z_p, z_o)$ separately. Since our model originates in the approximation of a nonlinear equation, the implied covariance of the linearized process for meaningful boundary conditions will not be stationary (since the advection would depend on space).

Modeling a nonstationary process in the output space is a difficult endeavor with no simple approach (other than using a simple stationary kernel class, e.g., a squared exponential [148]). Since our focus is on understanding the effect of ignoring covariance and getting access to a model that is reasonable and easy to set up, we still use the physics-based covariance model $\mathrm{Cov}(y_o, y_o) = L_o \mathrm{Cov}(z, z) L_o^T$ and $\mathrm{Cov}(y_p, y_o) = L_p \mathrm{Cov}(z, z) L_o^T$, where L_o, L_p correspond to the Jacobians of mapping F_o, F_p, respectively. Thus, we consider the effect of z on y for the purpose of the autocovariance but not for the purpose of the cross-covariance. While it would be more principled to set up a model that completely ignores the physics, we believe that using the "best model" for the auto-covariance in this circumstance allows us to sharply assess the effects of ignoring the cross-covariance, an approach also used in the paper that inspired this work [57]. Moreover, this does not affect the main concern of our paper, namely, whether the approximation we use to make the computation scalable still presents enough accuracy overall.

We note that if we made this choice, the prediction carried out by (2.5.14) would be identical to the latent process approach Section 2.3.1, provided that the parameters θ stay the same. Since, however, in the case of the model (2.5.12) $-$ (2.5.13) we use both z_o and y_o data to estimate θ, whereas for the latent process model in Section 2.3.1 we use only y_o, the parameters θ and thus the kriging results will be different. The joint process approach from Section 2.3.2, however, will use values of both y_o and z_o both for estimating θ and for kriging.

We consider a model using the joint process covariance model structure (2.3.8)-(2.3.10) with the hyperparameters defined above (2.5.6) that were used to produce the data (the "true" parameters). We denote this true model by $\mathcal{M}_*$.

We then take a fixed mesh size with $k = 200$, which results in a total of $40,000$ data points for the numerical approximation of PDE using (2.5.2). We pick $d = 400$ data points in the output field w as observations y_o, which corresponds to 1% of the total field to observe.

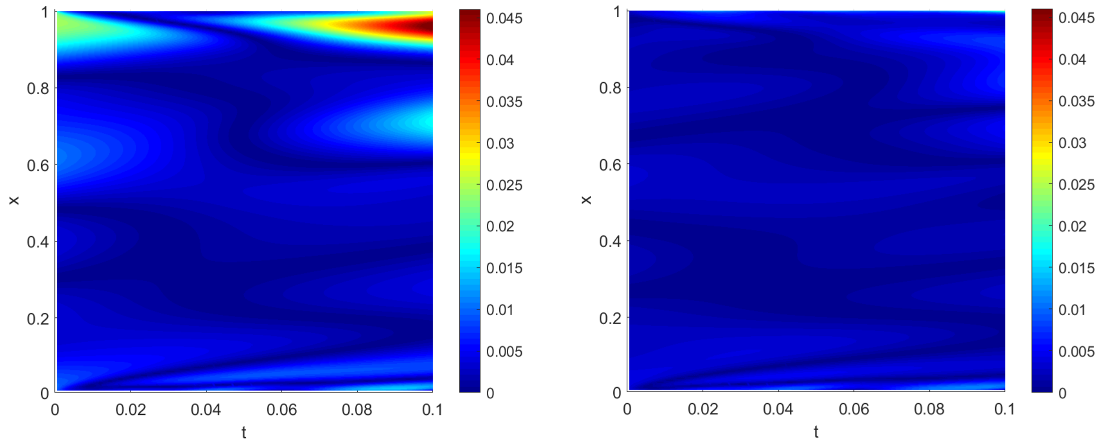
In the joint process case Section 2.3.2, we have $D = 20$ additional observations each from the initial conditions and boundary conditions as latent observations z_o, which are 10% of the latent variables. Note that the dimension of w is roughly the square of the dimension of z, so the dimension of the observations of z and w are comparable. Note also that the setting is similar to [57].

Since the computational complexity is not a major concern in this subsection, we predict the entire remaining field $y_p = w \setminus y_o$ and $z_p = z \setminus z_o$. We first generate one sample using the hyperparameters (2.5.6), termed the calibration sample, which is used to fit the covariance models $\mathcal{M}_1$ (2.3.3), $\mathcal{M}_2$ (2.3.8)-(2.3.10),$\mathcal{M}_{\text{ind}}$ (2.5.12)-(2.5.14) as described in Section 2.5.1. Then we further generate 50 independent samples used for tests in our experiment, termed validation samples. The results are averaged and summarized in the upper block of Table 2.1.

We note that the RMS error on y_p of the three physics-based covariance models $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_*$ is significantly smaller on the validation samples than the error of the independent kriging $\mathcal{M}_{\text{ind}}$, which demonstrates the contribution of the cross-covariance terms to the predictions. Furthermore, the two models using joint process kriging $\mathcal{M}_2$ and $\mathcal{M}_*$ have similar performance and are both better by about 15% on the validation samples compared with the latent process kriging $\mathcal{M}_1$ because of the additional information conveyed by the observations z_o. This improved accuracy occurs despite the fact that we have low-rank Section 2.2.2, differentiation Section 2.2.3 and linearization approximation errors in the computation of the covariance. While our starting the MLE at an initial guess obtained by fitting only a subset of observations may allow for the possibility of other local minima, the approximated likelihood has minima that are mostly indistinguishable from the true parameters for the purpose of the accuracy of the kriging.

Correction for Nonlinearity

Since the Burger's equation is nonlinear, we now consider adding higher-order terms in the covariance model to account for the nonlinear behaviors generated by the equation using the methods described in Section 2.3.3. To demonstrate the benefit gained from the higher-order terms, we consider the same experiment settings as in Section 2.5.1. Note that we consider the correction for nonlinearity only on models $\mathcal{M}_1$, $\mathcal{M}_2$, and $\mathcal{M}_*$. The models with higher-order terms included are denoted by $\mathcal{M}_1^+$, $\mathcal{M}_2^+$, and $\mathcal{M}_*^+$, respectively. Specifically, $\mathcal{M}_1^+$ follows the latent process kriging with higher-order terms in Algorithm 3. $\mathcal{M}_2^+$ and $\mathcal{M}_*^+$ follow the joint process kriging with higher-order terms in Algorithm 8. When setting the parameters $\theta = (\alpha_I, \alpha_B, l, \sigma)$ in $\mathcal{M}_1^+$ and $\mathcal{M}_2^+$, we use the parameters fitted by the approximated MLE described in Section 2.4 but for the linear approximation process only, that is, the same process described in Section 2.5.1 (in other words we consider the nonlinearity when kriging but not when carrying out MLE). In contrast, for $\mathcal{M}_*^+$ we use the true parameters (2.5.6). The performance of the nonlinearly corrected models on the same calibration sample and validation samples is summarized in the lower block of Table 2.1. We note that the performance of all three models is significantly better. The marginal improvement varies from $25\% - -50\%$ than their counterparts using covariance models without higher-order terms $\mathcal{M}_1$, $\mathcal{M}_2$, and $\mathcal{M}_*$ because the higher-order terms help account for the nonlinearities. The improvement is the largest in the joint model $\mathcal{M}_2$ to the point where it exceeds the one with the true parameters. It is unlikely that this situation will hold for all possible experimental validation setups, but the improvement in RMSE is clear across all models when considering nonlinear effects. Additionally, to provide a visual comparison of the predictions, we combine the predicted data and the observations to reconstruct the whole field w. To demonstrate the effectiveness of the higher-order terms, we compute the average error distribution over the 50 validation samples for joint process kriging models $\mathcal{M}_2$ and $\mathcal{M}_2^+$. The comparison is shown in Figure 2.5.2. As we can observe, the pattern of the error is much improved. The error is uniformly smaller for covariance models with higher-order terms (which indicates that in the max norm the error improvement is significant), which demonstrates the usefulness of the nonlinear correction. To support this observation, we report quartiles to describe the distribution of the error surface. The three quartiles (Q_1, Q_2, Q_3) of the error are 0.012, 0.025, and 0.048 for model $\mathcal{M}_2$ in Figure 2.5.2(a). In contrast, the three quartiles for model $\mathcal{M}_2^+$ in Figure 2.5.2(b) are 0.007, 0.015, and 0.024. Therefore the spatial distribution of the error has a much thinner tail for model $\mathcal{M}_2^+$ compared with model $\mathcal{M}_2$; indeed the third quartile of the error for $\mathcal{M}_2^+$ is about half of the third quartile of the error for $\mathcal{M}_2$.



(a) Without higher-order terms (b) With higher-order terms

Figure 2.5.2: Error distribution of the predicted field compared with the true data Section 2.5.1. **Left**: Results from joint process kriging $\mathcal{M}_2$. The covariance model follows (2.3.8)-(2.3.10), which includes up to second-order terms corresponding to (2.2.4). **Right**: Results from joint process kriging with higher-order terms $\mathcal{M}_2^+$. The covariance model follows (2.3.15)-(2.3.17), which includes up to fourth-order terms corresponding to (2.2.5)(2.2.6). The higher-order terms should account for part of the nonlinear behavior of Burger's equation.

Demonstrating Quasilinear Scaling

We now present numerical experiments to demonstrate the scalability of our proposed approach when computing the covariance parameters θ by the maximum likelihood approach in the Section 2.5.1 setting. Specifically, we carry out the computation of the approximated score equations Section 2.4, followed by the kriging procedures described in Section 2.3 to predict the field of interest at prescribed points. To vary the size of the problem, we take mesh size $k = 2^q$ for q ranging from 10 to 14. We randomly observe $d = 3k$ data points from the output process w and $D = 1.5k$ data points from the latent process z. Then we predict $3k$ points for the output process and $1.5k$ points for the latent process at sites that are also randomly chosen. The random part of our experiment is repeated 50 times. For both latent process kriging Section 2.3.1 and joint process kriging Section 2.3.2, we compare the average time cost over the 50 of our workflow without higher-order terms (Algorithm 1&2) and with higher-order terms (Algorithm 3&8). For the latent process MLE Section 2.4.1 and joint process MLE Section 2.4.2, we compare the time of solving the approximated score equations (2.4.3) and (2.4.13) with the time of solving the exact ones (where the covariance matrices are exactly computed and not approximated by interpolation); the timings reported are for the computation on the first sample only. The results are summarized in Figures 2.5.3 and 2.5.4. Lines corresponding to the theoretical scaling $O(k \log^2 k)$ are added to each plot to demonstrate the quasilinear scaling of our approaches.

As we can observe, the scaling of the proposed approximated approaches in latent and joint process kriging follow the $O(k \log^2 k)$ theoretical line. Comparing the algorithm with and without higher-order terms, we observe that the algorithm without higher-order terms scales slightly better because of the extra operations when considering higher-order terms. This effect will be alleviated as the datasets grow larger. For the score equations computation and Fisher information matrix estimation, the quasilinear scaling is clearly demonstrated in Figure 2.5.4. Note that our approximated approach requires stochastic estimation of the trace term, which is not as efficient as direct computation in small datasets, namely, $q = 10$ in Figure 2.5.4.

In summary, for the synthetic data case we conclude that our approach scales quasilinearly

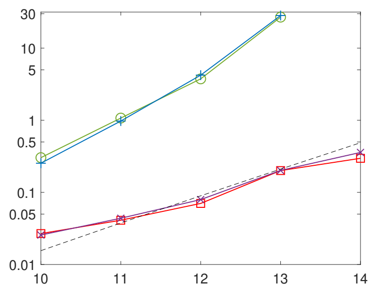
Section 2.5.1, that the approximations that we have proposed still allow an accurate recovery of the parameters defining the shape of the uncertainty Section 2.5.1, that the accurate treatment of the cross-covariance significantly improves prediction Section 2.5.1, and that including the nonlinear corrections in the covariance slightly improves the RMSE of the prediction but may significantly help with removing complex error artifacts Section 2.5.1.

2.5.2 Real Data Experiment

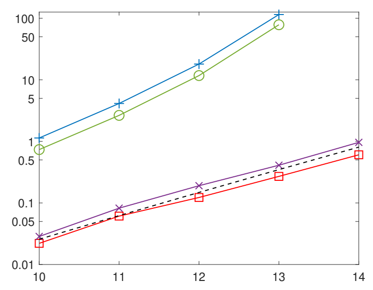
In this subsection, we consider realistic climate observation data and a simple tropical climate model connecting key quantities as the physical information we will use to inform our covariance model. The model equation is given by

$$\frac{\partial w}{\partial t} = -\tilde{Q}\nabla u - \frac{1}{\tau_w} w + b_w \nabla^2 w + D_w \dot{W}, \tag{2.5.15}$$

where u is the first baroclinic mode of the zonal wind velocity anomalies, w is the water vapor anomalies in middle troposphere, and $\dot{W}$ is Gaussian white noise. The core dynamic (2.5.15) uses a modified equation from [173]. The term $-\tilde{Q}\nabla u$ is a traditional model of convective adjustment of water vapor caused by wind. The term $-\frac{1}{\tau_w}w$ accounts for moisture sink associated with deep convection and precipitation. $b_w \nabla^2 w$ represents the diffusion of water vapor, and $D_w \dot{W}$ represents stochastic forcing. In short, this model represents the effect of precipitation, natural diffusion, and turbulent advection in a simplified, linearized form. Besides the deterministic dynamic components, the stochastic moisture forcing is, in part, representative of mesoscale convective processes that are not represented by the larger-scale dynamics in (2.5.15). Therefore one can reasonably assume that such processes are approximately spatiotemporally uncorrelated. The parameter values $\tilde{Q} = 0.45$, $\tau_w = 3.99$, $b_w = 0.7$, and $D_w = 0.003$ are all dimensionless and are suggested by [173]. To numerically solve the model (2.5.15), we use the initial condition w_0 and random field u as model inputs

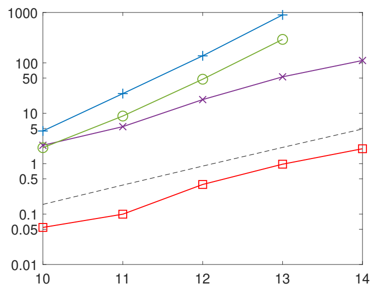


(a) Time for latent process kriging

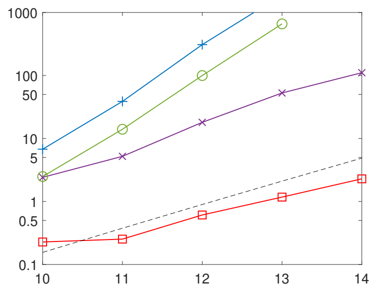


(b) Time for joint process kriging

Figure 2.5.3: Times taken in seconds to solve the GP regression problem (y-axis) vs. number of mesh points 2^q for q ranging from 10 to 14 (x-axis) by (a) latent process kriging (Algorithm 1&3) and (b) joint process kriging (Algorithm 2&8). We present the time to compute latent/joint process kriging without higher-order terms exactly ($\bigcirc$) and approximately ($\square$) and latent/joint process kriging with higher-order terms exactly ($+$) and approximately ($\times$). All results are averaged over 50 tests. Note that the y-axis is in logarithmic scale. The lines corresponding to theoretical $O(k\log^2 k)$ scaling (dash lines) are added to each plot.



(a) Time for estimating latent & Joint process score equations



(b) Time for estimating the empirical Fisher information matrix

Figure 2.5.4: Times taken in seconds (y-axis) vs. number of mesh points 2^q for q ranging from 10 to 14 (x-axis) for evaluating (a) latent and joint process score equations (Algorithm 4&5) and (b) the expected Fisher information matrix via equation (2.4.22), (2.4.23). For both plots, we present the time to evaluate the score equations or the expected Fisher information matrix for latent process exactly ($\bigcirc$) and approximately ($\square$) and for joint process exactly ($+$) and approximately ($\times$). All results are averaged over 50 tests. Note that the y-axis is in logarithmic scale. The lines corresponding to theoretical $O(k\log^2 k)$ scaling (dash lines) are added to each plot.

with periodic boundary condition. The equation can be solved by using the Euler-Maruyama method [100] and also in Fourier space [162]. We recommend solving in Fourier modes since a semi-analytical solution [162] exists for each Fourier mode and can significantly reduce the time cost of numerical simulation. To apply our framework, we numerically solve w as output process, and we treat the concatenation of the initial condition w_0, the random process u, and $\dot{W}$ as the latent process of our framework, in other words, $z^T = (w_0^T, u^T)$ in our framework §(2.3.1)-(2.3.2). Similar to Section 2.5.1, we define $w = G(z)$, where w is the solution produced by semi-analytical solver of stochastic differential equation (2.5.15). In the following test, we denote y_o as the observed subset of w, which includes additional observation noise, and y_p as another subset that is going to be predicted. The mappings F_o, F_p required by our setup (2.3.1)-(2.3.2) are obtained by corresponding components of mapping G.

Data Source and Data Processing

To associate the model (2.5.15) with a suitable real dataset, we argue that the zonal wind velocity serves as a natural surrogate of u and that the relative humidity (mixing ratio) in 500 hPa pressure level is a common surrogate of the water vapor w in middle troposphere.

The data source we used here is the National Centers for Environmental Prediction–National Center for Atmospheric Research (NCEP-NCAR) reanalysis project [91]. The daily zonal wind velocity and relative humidity data are used. Both datasets have a spatial resolution of $2.5^\circ \times 2.5^\circ$ and a daily temporal resolution from 1 January 1979 to 31 December 2011. The connections between model variables and observations are largely based on previous work [175, 174, 136].

The influence of seasonal cycle has been removed from both datasets, and a further 120-day mean of previous 120-day signals is subtracted, as is recommended by the Climate Variability and Predictability program [196]. Then both datasets are projected to the parabolic cylinder functions, and only the first meridional mode is used. This step converted the two-dimensional spatial data to one dimensional, which facilitates our computation. Then the datasets have two dimensions remaining (one spatial dimension and one temporal dimension). Furthermore, we treat the observation data as true signals and generate artificial observations by adding independent Gaussian noise $\epsilon_w \sim \mathcal{N}(0, \sigma_w^2 I)$ and $\epsilon_u \sim \mathcal{N}(0, \sigma_u^2 I)$ to water vapor data w and zonal wind velocity data u, respectively. Parameters σ_w and σ_u are taken to be 15% of the standard deviation of each true signal, similar to other previous studies (e.g., [48]).

Note that the real data are normalized to have the same variance as the model data before using. This approach differs from [173], where the data are nondimensionalized using the standard equatorial reference scale [174] because here we use anomalies in our modified model equation. For more detailed raw data preprocessing methods see [174]. In Figure 2.5.5 we show the observational relative humidity data for all longitudes from Dec. 1982 to April 1983.

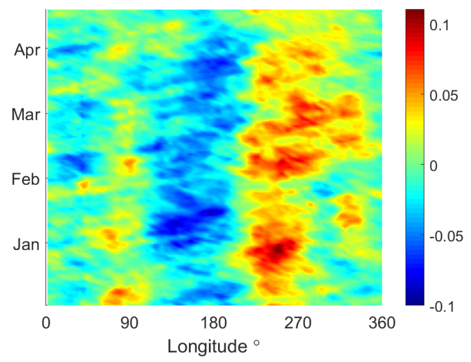


Figure 2.5.5: Observations of relative humidity at 500 hPa pressure level. The time period is from Dec. 1, 1982 to April 18, 1983. The data have been nondimensionalized.

End-to-End Numerical Tests

We combine our proposed scalable MLE parameter estimation and GP regression together and test their performance. Four covariance structures are compared here. The latent process kriging $\mathcal{M}_1$ (2.3.3) and the joint process kriging $\mathcal{M}_2$ (2.3.8)-(2.3.10) follow the same design as we discussed in Section 2.3. Additionally the joint independent kriging $\mathcal{M}_{\text{ind}}$ ignores the cross-covariance between the latent process z and the output w and is formulated as follows:

$$K_{11} = \begin{pmatrix} \mathrm{Cov}(y_o, y_o) + \sigma_w^2 I & 0 \\ 0 & \mathrm{Cov}(z_o, z_o) + K_{\epsilon_z} \end{pmatrix}, \tag{2.5.16}$$

$$K_{21} = \begin{pmatrix} \mathrm{Cov}(y_p, y_o) & 0 \\ 0 & \mathrm{Cov}(z_p, z_o) \end{pmatrix}, \tag{2.5.17}$$

$$\mathcal{M}_{\mathrm{ind}} = m(y_p) + (K_{21})(K_{11})^{-1} \begin{pmatrix} y_o - m(y_o) \\ z_o - m(z_o) \end{pmatrix}, \tag{2.5.18}$$

where z_o consists of an observed subset of the components of the latent process z and z_p consists of another subset that is to be predicted. K_{ϵ_z} is the observation noise covariance matrix. Since z is a concatenation of initial value w_0 and wind velocity field u, K_{ϵ_z} is block-diagonal with Gaussian observation noise covariance matrices in its diagonal blocks. Similar to the approach in Section 2.5.1, we use the physics-based covariance model $\mathrm{Cov}(y_o, y_o) = L_o \mathrm{Cov}(z, z) L_o^T$ and $\mathrm{Cov}(y_p, y_o) = L_p \mathrm{Cov}(z, z) L_o^T$, where L_o, L_p correspond to the Jacobians of mapping F_o, F_p, respectively. The computation involving the three models is carried out but using low-rank approximations. We include kriging using the exact joint process covariance model (2.3.8)-(2.3.10) but without low-rank approximations as contrast. We denote this exact joint process kriging model as $\mathcal{M}_e$.

For all the covariance models $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_{\mathrm{ind}}$, and $\mathcal{M}_e$ used here, each component of the latent process z is modeled by a Gaussian process with square exponential covariance function. In other words, we assume

$$w_0 \sim \mathcal{N}(m(w_0), \alpha_w \exp(-\frac{r^2}{2l_w^2})), \tag{2.5.19}$$

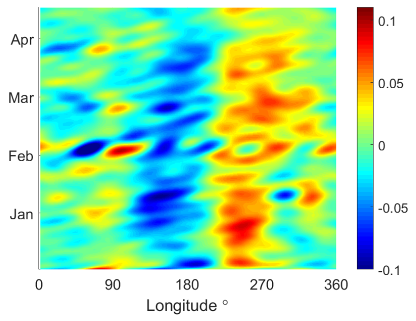
$$u \sim \mathcal{N}(m(u), \alpha_u \exp(-\frac{r^2}{2l_u^2})), \tag{2.5.20}$$

with magnitude parameters α_w and α_u and length scale parameters l_w, l_u. Let $\theta = (\alpha_w, \alpha_u, l_w, l_u)$ be the unknown parameters that we will estimate; we assume that the other parameters, including model parameters in (2.5.15) and observation error covariance, are known and specified at the beginning of Section 2.5.2 based on [173]. All four parameters will be estimated from the observation data via score equations.

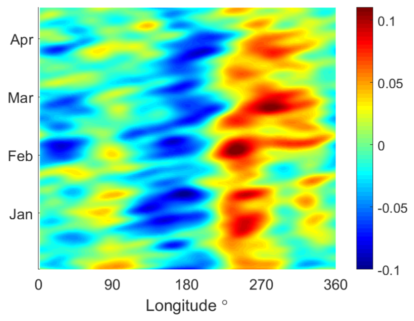
The processed relative humidity observation data are divided into different time snapshots with equal lengths. Each snapshot has 144×140 grid points (20,160 data points) that represent 144 longitudes each day and an 140-day time window. First we randomly pick one snapshot field as the calibration sample. We then sample random observations from the snapshot field for parameter fitting. In the spatial dimension we take evenly distributed grid points with indices $(4 : 4 : 144)$ in Matlab notation to observe. In the temporal dimension we take 46 random slices. In total it corresponds to $36 \times 46 = 1656$ observations which represents 8.21% of the total field. For models $\mathcal{M}_1$, $\mathcal{M}_2$, and $\mathcal{M}_{\text{ind}}$, we use scalable latent and joint process MLE described in Section 2.4. For model $\mathcal{M}_e$, we directly evaluate the score equations without approximations. The fitted parameters are used in the following kriging processes. The validation samples are chosen to be snapshots of the same size but at least 120 days away from the calibration sample, which can help remove the effect of the trend of most intraseasonal oscillations. Like the calibration sample, we take random time slices with a total 8.21% of possible observations for kriging. Numerical results are summarized in Table 2.2. We observe that the approximation we made to the likelihood does not result in a significant kriging error, by comparing the validation sample performance of $\mathcal{M}_1$, $\mathcal{M}_2$, and $\mathcal{M}_e$ models whose performance is indistinguishable (despite the fact that model $\mathcal{M}_e$ is more expensive and not scalably computable). We also observe that ignoring the covariance structure reduces the performance of the $\mathcal{M}_{ind}$ by about 20% (comparing the RMS error of $\mathcal{M}_2$ and $\mathcal{M}_{\text{ind}}$ for output process y_p in the calibration sample of TABLE 2.2). We note that the improvements of the joint process model $\mathcal{M}_2$ over other covariance models are less significant in the real data case. However, we also see that our algorithms scale very well and allow computations in circumstances where exact algorithms cannot cope even in terms of memory. Exact computations of maximum likelihood estimations and kriging can be costly or impossible even for the independent covariance model $\mathcal{M}_{\text{ind}}$ in our large data set cases, or other realistic applications. Therefore our approximation ensures scalability at a negligible cost to the accuracy, and the accurate modeling of the cross-covariance pays off in accuracy improvements for real data , even if not as strongly as in the synthetic data case.

To provide a visual representation of the results, we illustrate in Figure 2.5.6 the relative humidity predictions on the calibration sample using model $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_{\text{ind}}$. The predicted time window is Dec. 1982 to April 1983, which is the same as observations in Figure 2.5.5. In Figure 2.5.7 we show histograms of the predictive errors of the two models, respectively. Note that the vertical axis represents the probability density of a certain error range in logarithmic scale. For bars with zero probability, we truncate the logarithm and set it as the base value of the bar plot in order to prevent it from approaching negative infinity. As a result, the zero probability bars are also of height zero. We observe that the prediction error is reduced and has thinner tails when using the physics-based covariance models $\mathcal{M}_1$, $\mathcal{M}_2$, and particularly significant improvements can be observed for the latter. Specifically, the prediction error for joint process model $\mathcal{M}_2$ is much evenly distributed.

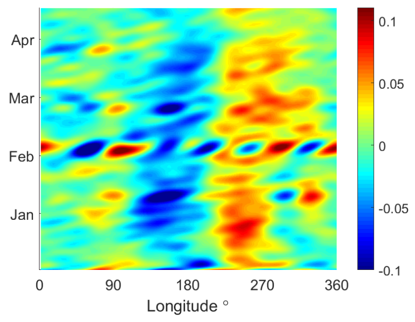
We conclude that in this case the physics-based model produces a moderate improvement in prediction with this real dataset. We have tried other configurations; and while the physics-based cross-correlation model was consistently better than the independent model,



(a) Latent process kriging $\mathcal{M}_1$



(b) Joint process kriging $\mathcal{M}_2$

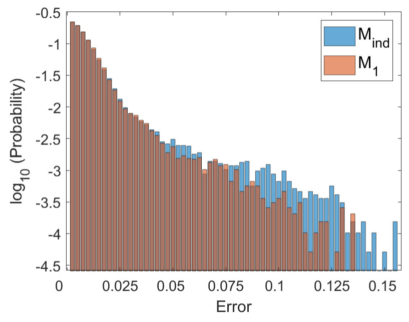


(c) Joint independent kriging $\mathcal{M}_{\text{ind}}$

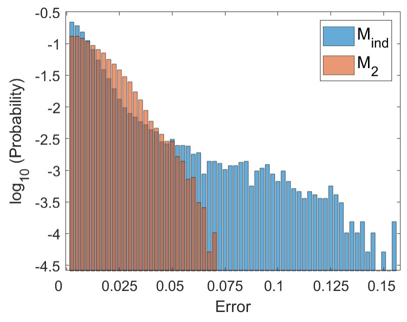
Figure 2.5.6: Predictions corresponding to (a) latent process kriging $\mathcal{M}_1$, (b) joint process kriging $\mathcal{M}_2$, and (c) joint independent kriging $\mathcal{M}_{\text{ind}}$. The time period is from Dec. 1, 1982, to April 18, 1983. Both datasets are nondimensional.

the improvement seemed not particularly significant in several cases. Several areas for improvement remain here, such as considering less smooth covariance kernels (at the price of increasing the rank of the approximation or even changing the approximation strategy) and datasets with greater large-scale variability or more pronounced nonlinear effects.

We note, however, that the computational performance was approximately linear with the number of data points and that the statistical performance was comparable to the one of exactly computed models that had far more intense computational requirements. We conclude that our approach of a physics-based implicitly defined kernel using our scalable approximation method brings a notable methodological improvement for real datasets in that it is scalable as opposed to the classical approach and that it sometimes leads to noticeable improvements in the statistical performance.



(a) Error histograms $\mathcal{M}_1$ vs. $\mathcal{M}_{\text{ind}}$



(b) Error histograms $\mathcal{M}_2$ vs. $\mathcal{M}_{\text{ind}}$

Figure 2.5.7: Error histograms corresponding to independent process kriging $\mathcal{M}_{\text{ind}}$ and joint process kriging $\mathcal{M}_2$. Note that the vertical axis represents the probability densities in logarithmic scale. The logarithm of zero density has been truncated to meet the base value of each bar plot.

2.6 Discussion

GPs are powerful tools in statistical modeling. In that space, the covariance structure takes a critical role in forecast accuracy and efficiency. Few pointers exist, however, on how to design good covariance kernels for complex processes. On the other hand, many phenomena in the natural sciences such as physics, biology, Earth science, and chemistry have been well studied, and mathematical or physical models have been established to interpret the underlying relations among the variables. Including such information from the physical models when constructing the covariance structure can be meaningful for forecasts when performing inferences on processes with partially known physical relations. One drawback of GPs, however, is that their classical computational approach, based on the Cholesky factorization of an often dense kernel, scales cubically with the number of observations. For large datasets, a reduction in the computational cost is necessary before such methods become practical.

In this chapter, we utilized physics-based, implicitly defined covariance models and present a low-rank approximation of the covariance matrix that allows the GP regression and the MLE to be conducted in quasilinear time scale. The implicitly defined nature of the kernel gives significant flexibility to the user who needs only access to a coarse physical model represented by a forward solver. Moreover, we proposed a method to approximate the expected Fisher information matrix for quantifying the uncertainties of the estimates. Furthermore, we proposed approaches to include higher-order terms in auto-covariance matrices that are essential for describing nonlinear processes while maintaining quasilinear scaling. In summary, we presented a coherent framework for efficiently interpolating the random field and produced uncertainty estimates of this task by means of partial observations and exploiting approximate physical relationships. We presented several numerical experiments that demonstrate the accuracy of the approximated MLE by showing that the approximated parameter estimates and the uncertainties are relatively close to the exact values when the latter are known. Then we used the estimated parameters in the physics-based covariance models and demonstrated that a covariance model that is complete and has correct physically consistent structure yields significant improvements in forecasts accuracy and efficiency. We applied the framework to real climate data to further illustrate the effectiveness of our algorithms.

Our approach is implicit in that it requires only a black-box forward solver of the approximate physical relationships, making the extension of the algorithm to other models immediate. This feature increases the flexibility of the approach, benefiting from plentiful well-studied numerical discretization schemes and well-established simulation toolkits for solving physical models. Also our approach consists of independent calls to the solver that can fully take advantage of the parallel computing capability of modern hardware.

Our approach does have several shortcomings. Our approximations of the gradient and Hessian can be accurate under certain regularity conditions of the physical model. For non-differentiable or even discontinuous models, however, the approximations can be inefficient, making the approach case-dependent. Perhaps more problematic is the fact that we apply the approach to noise kernels that are very smooth. While this is a common choice, in many cases it may not be appropriate, and we need to change the approximation method. We chose the Chebyshev interpolation method because it is simple to implement and the compressed covariance matrix is differentiable, but this can be improved by using the Nyström method and its variants [105, 198], hierarchical matrix approximations [6, 33], or adaptive low-rank approximations [116, 197]. In any case, simple global low-rank approximations like the ones we used here work well for covariance functions that are sufficiently smooth and dominated by long-range relationships between data points. When the covariance function changes more rapidly, global low-rank methods cannot capture the variability and other methods, such as hierarchical matrix approximations [6, 33], may be needed.

Algorithm 3: Latent Process Kriging with Higher-Order Terms $\mathcal{M}_1^+$

Use the low-rank approximation with $N = O(\log n)$, and then solve the approximated problem,

$$m(y_p)+\left(L_pC_z^TKC_zL_o^T+\frac{1}{N_l}\sum_{i=1}^{N_l}(v_i^TM_pu_i)(v_i^TM_ou_i)^T\right)$$
$$\left(K_{\epsilon_y}+L_oC_z^TKC_zL_o^T+\frac{1}{N_l}\sum_{i=1}^{N_l}(v_i^TM_ou_i)(v_i^TM_ou_i)^T\right)^{-1}(y_o-m(y_o)). \tag{2.3.23}$$

Step 1. Compute $A_1 = L_oC_z^T \in \mathbb{R}^{d\times N}$, $A_2 = L_pC_z^T \in \mathbb{R}^{(m-d)\times N}$ by $O(N)$ forward solves.

Step 2. Compute the Cholesky factorization $K = P^TP$. This takes $O(N^3)$ time.

Step 3. Draw $2N_l$ independent Rademacher vectors $\{u_i\}, \{v_i\}$. Compute

$$\phi_i = C_z^TP^Tu_i, \psi_i = C_z^TP^Tv_i. \tag{2.3.24}$$

This takes $O(2N_l(N^2 + nN)))$ time.

Step 4. Compute the vector-Hessian-vector product $w_i = \psi_i^TH_o\phi_i$, $w_i' = \psi_i^TH_p\phi_i$, $i = 1, 2, \ldots, N_l$, by the approximation (2.3.2). This approximation takes $O(N_l)$ forward solves. Define the matrices $A_3 \leftarrow \frac{1}{\sqrt{2N_l}}(w_1, w_2, \cdots, w_{N_l}) \in \mathbb{R}^{d\times N_l}$, $A_4 \leftarrow \frac{1}{\sqrt{2N_l}}(w_1', w_2', \cdots, w_{N_l}') \in \mathbb{R}^{(m-d)\times N_l}$. From (2.3.22) we will approximate the higher-order terms by

$$\frac{1}{2}\mathrm{tr}(H_o\mathrm{Cov(z,z)}H_o^T\mathrm{Cov(z,z)}) \approx A_3A_3^T, \tag{2.3.25}$$
$$\frac{1}{2}\mathrm{tr}(H_p\mathrm{Cov(z,z)}H_o^T\mathrm{Cov(z,z)}) \approx A_4A_3^T. \tag{2.3.26}$$

Step 5. Solve the modified inverse problem

$$\begin{aligned}\alpha &= (K_{\epsilon_y} + A_1KA_1^T + A_3A_3^T)^{-1}(y_o - m(y_o)).\\ &= \left(K_{\epsilon_y} + \begin{pmatrix}A_1P^T & A_3\end{pmatrix}\begin{pmatrix}PA_1^T\\ A_3^T\end{pmatrix}\right)^{-1}(y_o - m(y_o))\end{aligned} \tag{2.3.27}$$

by applying the SMW formula for the rank-$(N + N_l)$ perturbation, as in Algorithm 1. The time cost is dominated by computing the matrix products that define the blocks of $\begin{pmatrix}A_3^TK_{\epsilon_y}^{-1}A_3 & A_3^TK_{\epsilon_y}^{-1}A_1P^T\\ PA_1^TK_{\epsilon_y}^{-1}A_3 & PA_1^TK_{\epsilon_y}^{-1}A_1P^T\end{pmatrix}$. This takes $O(N + N_l)$ linear system solves with K_{ϵ_y} and $O(m(N + N_l)^2)$ time.

Step 6. **Return** $m(y_p) + (A_2KA_1^T + A_4A_3^T)\alpha$ by matrix-vector product. It takes $O(2m(N + N_l) + N^2)$ time.

Algorithm 4: Latent Process Score Equations

Step 1. Compute $A_1 = L_o C_z^T \in \mathbb{R}^{d\times N}$ using $O(N)$ forward solves.

Step 2. Compute the approximation of the first term in the score equations,

$$\alpha_1^i \approx \frac{1}{2}(y_o - \overline{y_o})^T (K_{\epsilon_y} + A_1 K(\theta) A_1^T)^{-1} (A_1 K_i(\theta) A_1^T)(K_{\epsilon_y} + A_1 K(\theta) A_1^T)^{-1}(y_o - \overline{y_o}) \tag{2.4.6}$$

$$= \frac{1}{2}(y_o - \overline{y_o})^T W K_i(\theta) W^T (y_o - \overline{y_o}), i = 1, 2, \ldots, r \tag{2.4.7}$$

where $W = (K_{\epsilon_y} + A_1 K(\theta) A_1^T)^{-1} A_1 \in \mathbb{R}^{d\times N}$ is computed by the SMW formula. The time cost is dominated by computing matrix products $(A_1^T K_{\epsilon_y} A_1)$, which takes $O(N)$ linear system solves with the noise covariance matrix K_{ϵ_y} and $O(mN^2)$ time to assemble. Note that W needs to be computed only once for all i, after which the number of operations depends only on N. Therefore the dominant cost is independent of r for this step. We compute and store only W in this step.

Step 3. Draw N_l independent Rademacher vectors $\{u_j\} \in \mathbb{R}^d$. Approximate the second trace term by

$$\alpha_2^i \approx -\frac{1}{2}\mathrm{tr}((K_{\epsilon_y} + A_1 K(\theta) A_1^T)^{-1}(A_1 K_i(\theta) A_1^T)) \tag{2.4.8}$$

$$\approx -\frac{1}{2N_l}\sum_{j=1}^{N_l} u_j^T W K_i(\theta) A_1^T u_j. \tag{2.4.9}$$

In this step, we compute only $\{A_1^T u_j\}$ and $W^T u_j$ for $j = 1, 2, \cdots, N_l$.The cost is $O(mNN_l)$.

Step 4. Set $g_i(\theta) = \alpha_1^i + \alpha_2^i$ for $i = 1, 2, \cdots, r$. Assembling (2.4.7) and (2.4.9) for each i, j takes a total $O(rN_l N^2 + mN)$ time.

Step 5. The left-hand side of the score equations is now available as

$$g_i(\theta) = 0,\ i = 1, 2, \cdots, r. \tag{2.4.10}$$

Problem (2.4.10) can be solved by, for example, Broyden's method, which requires only the left-hand side for a given θ and can thus be computed by Steps 1–4.

Algorithm 5: Joint Process Score Equations

Step 1. Compute $A_1 = L_o C_z^T \in \mathbb{R}^{d\times N}$, $A_2 = L_p C_z^T \in \mathbb{R}^{(m-d)\times N}$ by $O(N)$ forward solves.

Step 2. Compute the approximation of the first term in score equations,

$$\alpha_1^i \approx \frac{1}{2}\begin{pmatrix} y_o - \overline{y_o} \\ z_o - \overline{z_o}\end{pmatrix}^T \left(\begin{pmatrix} K_{\epsilon_y} & 0 \\ 0 & K_{\epsilon_z}\end{pmatrix} + \begin{pmatrix} A_1 & 0 \\ 0 & C_{z_o}^T\end{pmatrix}\begin{pmatrix} K(\theta) & K(\theta) \\ K(\theta) & K(\theta)\end{pmatrix}\begin{pmatrix} A_1^T & 0 \\ 0 & C_{z_o}\end{pmatrix}\right)^{-1}$$
$$\begin{pmatrix} A_1 & 0 \\ 0 & C_{z_o}^T\end{pmatrix}\begin{pmatrix} K_i(\theta) & K_i(\theta) \\ K_i(\theta) & K_i(\theta)\end{pmatrix}\begin{pmatrix} A_1^T & 0 \\ 0 & C_{z_o}\end{pmatrix}\left(\begin{pmatrix} K_{\epsilon_y} & 0 \\ 0 & K_{\epsilon_z}\end{pmatrix} + \begin{pmatrix} A_1 & 0 \\ 0 & C_{z_o}^T\end{pmatrix}\right.$$
$$\left.\begin{pmatrix} K(\theta) & K(\theta) \\ K(\theta) & K(\theta)\end{pmatrix}\begin{pmatrix} A_1^T & 0 \\ 0 & C_{z_o}\end{pmatrix}\right)^{-1}\begin{pmatrix} y_o - \overline{y_o} \\ z_o - \overline{z_o}\end{pmatrix}. \tag{2.4.14}$$

Notice that the expression is symmetric. We first compute the term

$$W = \left(\begin{pmatrix} K_{\epsilon_y} & 0 \\ 0 & K_{\epsilon_z}\end{pmatrix} + \begin{pmatrix} A_1 & 0 \\ 0 & C_{z_o}^T\end{pmatrix}\begin{pmatrix} K(\theta) & K(\theta) \\ K(\theta) & K(\theta)\end{pmatrix}\begin{pmatrix} A_1^T & 0 \\ 0 & C_{z_o}\end{pmatrix}\right)^{-1}\begin{pmatrix} A_1 & 0 \\ 0 & C_{z_o}^T\end{pmatrix}. \tag{2.4.15}$$

The time cost is dominated by computing the matrix products $(A_1^T K_{\epsilon_y}^{-1} A_1)$ and $(C_{z_o} K_{\epsilon_z}^{-1} C_{z_o}^T)$, which require $O(N)$ linear system solves with observation noise covariance matrices and $O((n+m)N^2)$ assembly time. This computation needs to be done only once for all $i = 1, 2, \ldots, r$. We will store W and assemble the entire expression (2.4.14) later on.

Step 3. Draw N_l independent Rademacher vector $\{u_j\} \in \mathbb{R}^{d+D}$. Approximate the trace term in (2.4.13) by

$$\alpha_2^i \approx -\frac{1}{2N_l}\sum_{j=1}^{N_l} u_j^T W \begin{pmatrix} K_i(\theta) & K_i(\theta) \\ K_i(\theta) & K_i(\theta)\end{pmatrix}\begin{pmatrix} A_1^T & 0 \\ 0 & C_{z_o}\end{pmatrix} u_j. \tag{2.4.16}$$

As in Algorithm 4 in this step we compute only the terms $(W^T u_j)$ and $\begin{pmatrix} A_1^T & 0 \\ 0 & C_{z_o}\end{pmatrix} u_j$ for each j. We store these vectors and assemble the whole term later on. This step requires $O((n+m)NN_l)$ time to compute.

Step 4. We now set $g_i(\theta) = \alpha_1^i + \alpha_2^i$ for $i = 1, 2, \cdots, r$. The main computational expense consists of computing (2.4.14) and (2.4.16) for each i, j, which takes $O(rN_l N^2 + (n+m)N)$ time.

Step 5. The left-hand side of the score equations is now available as

$$g_i(\theta) = 0, i = 1, 2, \cdots, r. \tag{2.4.17}$$

The problem can be solved by using, for example, Broyden's method and *Step 1-4* to compute the components of the gradient.

Table 2.1: Predictive RMS error and relative error using 2-norm. The observation density is 1% for w and 10% for z, although the dimension of the observation vectors is comparable. The observations are chosen randomly in the field. The predictions are conducted for all the remaining field in w and z except the observed positions. y_p, z_p denote the absolute RMS error of the predicted y, z field respectively. ϵ_y and ϵ_z denote the relative error of the predicted y, z fields using vector 2-norm. The first four rows contain covariance models without higher-order terms from Section 2.5.1, and the last three rows contain covariance models with higher-order terms from Section 2.5.1.

	Validation Samples				Calibration Sample			
Models	y_p	ϵ_y	z_p	ϵ_z	y_p	ϵ_y	z_p	ϵ_z
$\mathcal{M}_1$	0.0273	0.0381	-	-	0.0269	0.0296	-	-
$\mathcal{M}_2$	0.0226	0.0316	0.0323	0.0662	0.0234	0.0258	0.0296	0.0512
$\mathcal{M}_{\text{ind}}$	0.0312	0.0435	0.0411	0.0842	0.0335	0.0367	0.0394	0.0681
$\mathcal{M}_*$	0.0226	0.0316	0.0323	0.0662	0.0235	0.0258	0.0295	0.0510
$\mathcal{M}_1^+$	0.0197	0.0272	-	-	0.0200	0.0220	-	-
$\mathcal{M}_2^+$	0.0120	0.0166	0.0260	0.0533	0.0115	0.0126	0.0239	0.0414
$\mathcal{M}_*^+$	0.0120	0.0166	0.0260	0.0533	0.0116	0.0127	0.0238	0.0412

Table 2.2: Predictive RMS error and relative error using 2-norm. The observation density is 8.21% for q and 8.21% for u. The observations are randomly chosen in the field. The predictions are conducted for all the remaining field in y and z except the observed positions. y_p, z_p denote the absolute RMS error of the predicted y, z fields respectively. ϵ_y and ϵ_z denote the relative error of the predicted y, z fields using vector 2-norm.

	Validation Samples				Calibration Sample			
Models	y_p	ϵ_y	z_p	ϵ_z	y_p	ϵ_y	z_p	ϵ_z
$\mathcal{M}_1$	0.0144	0.6153	-	-	0.0168	0.5489	-	-
$\mathcal{M}_2$	0.0139	0.5958	0.0410	0.8819	0.0165	0.5393	0.0393	0.6630
$\mathcal{M}_{\text{ind}}$	0.0193	0.8253	0.0604	1.3010	0.0201	0.6600	0.0988	1.6680
$\mathcal{M}_e$	0.0138	0.5913	0.0345	0.7433	0.0168	0.5494	0.0401	0.6778

CHAPTER 3
SCALABLE PHYSICS-BASED MAXIMUM LIKELIHOOD ESTIMATION USING HIERARCHICAL MATRICES

3.1 Introduction

GPs have been widely applied throughout the statistical and machine learning communities for modeling and regression problems [54, 58]. For overviews of their usage, one can refer to several extensive monographs [148, 125, 209]. One of the most appealing features of GPs is that they provide a fully probabilistic framework for modeling variability in stochastic processes, predicting unobserved values and providing prediction uncertainties. Gaussian process regression, also referred to as kriging, is considered one of the standard methods for modeling a range of natural phenomena from geophysics to biology [209, 176, 101]. In practice, we may want to model multiple processes that are highly correlated. Extending the technique to multiple target variables is known as co-kriging [176, 35] or multi-kriging [54]. In this work, we focus on multivariate co-kriging where the variables can be extracted from spatial processes or spatio-temporal processes.

An important feature of GPs is that the model can be fully characterized by its mean and covariance functions. Correct and accurate covariance models play an essential role in setting up meaningful GP regression problems. When applying kriging to single random fields, one can construct or choose proper parametric covariance functions based on a variety of practical or theoretical guidelines [148, 176]. When modeling multiple processes, however, it is hard to construct correct covariance structure that accounts for how the processes interact. To tackle the difficulty, many efforts have been carried out to incorporate prior knowledge about the underlying physical principles of the processes into the covariance model structure. A typical strategy of including prior knowledge of a real system governed by known physical laws is hierarchical Bayesian modeling [69, 20]. Examples include atmospheric modeling [207, 23, 156, 53] and environmental sciences [23, 21, 22, 208, 55]. Recently in [57], the authors proposed a systematic framework of assembling consistent auto- and cross-covariance models based on known physical processes. The endeavor showed that significant improvements in the forecast efficiency can be achieved by using physics-based covariance models. To address the complexity issue for large scale problems, our previous work [49] propose a way to construct the low-rank approximations of the physics-based covariance models using black-box forward solvers for implicit physical models. It results in a scalable approach of performing GP analysis using physics-based covariance models. However the low-rank assumption may not hold in many applications where the covariance model is often of full-rank. In this work, we propose to relax the assumption and approximate the physics-based covariance models in more general hierarchical matrix format while still only relying on forward applications of the physical models and maintaining a quasilinear complexity. We include more details about the physics-based covariance models in Section 3.2.

Another feature of maximum likelihood computation for GPs is that the main step in the calculation requires computing the inverse and determinant of the covariance matrix, in turn typically carried out by Cholesky factorization. If the matrix has no exploitable structure, the Cholesky factorization requires $O(n^3)$ computations and $O(n^2)$ storage for an $n \times n$ matrix. With the dramatic increase in the efficiency and capability of data collection, the computational and memory cost can soon become prohibitive for large scale problems. As a result, devising approximations to reduce both the computational and memory cost became a burgeoning field, using, for example, low-rank methods [170, 135], matrix tapering [68, 93, 44], Gaussian predictive processes [16], fixed-rank kriging [59], and Gaussian Markov random field approximations [157, 158, 67]. A matrix-free approach for solving the multi-parametric Gaussian maximum likelihood problem was developed in [8].

Recently, significant efforts have been expanded to apply hierarchical matrices to induce low-rank block structure in the original dense matrices to improve both the storage complexity and the computation complexity for several common matrix operations, including matrix-vector products, factorizations and linear system solves. Various representations for hierarchical matrices have been proposed in the past [4, 31, 32, 212, 77] with active applications in different fields. The technique has also been proposed for Gaussian process computations and parameter estimations [33, 131, 122, 5, 47], uncertainty quantification and conditional generation [160], and high-dimensional Gaussian probability computations [70, 40]. In this work, we focus on a specific class of matrices referred to as as hierarchical off-diagonal low-rank (HODLR) matrices [4]. In our applications, the off-diagonal sub-blocks are well approximated by low-rank matrices and this structure significantly reduces the complexity of many linear algebra operations. In [4] it was shown that a typical HODLR matrix can be hierarchically factorized into a sequential product of block low-rank updates to the identity, yielding direct algorithms for linear system solve and evaluation of determinants in quasilinear scale. Both operations are essential components for evaluating the Gaussian likelihood.

When a square matrix can be well-approximated by a hierarchical matrices structure, many methods can be applied to construct the hierarchical approximation via compressing the off-diagonal blocks to low-rank. In [5], the authors provided several ways to obtain the low-rank factorization of the off-diagonal blocks assuming the blocks are explicitly given. In [215, 52, 128], the methods for constructing hierarchical representations are presented relying on an $O(1)$ access to individual matrix entries. When explicit matrix entry access is not available, one can use randomized algorithms to efficiently approximate the target matrix from “black box” matrix-vector multiplication subroutines [119, 34]. Compared to the aforementioned explicit approaches, randomized methods avoid direct evaluations of the local blocks, which is essential when the dense matrix is costly to evaluate [3]. The numerical accuracy of randomized algorithms has also been well-studied; see [79] for details.

In this work, we combine physics-based covariance models [57] and randomized "black-box" hierarchical matrix construction [3] and enhance them with derivative computations to design an efficient HODLR framework for Gaussian process maximum likelihood estimation. Compared to previous work, particularly [57], which proposed implicit covariance models assuming a low rank of the process covariance structure, and [71] which used HODLR for explicit Gaussian kernels, our contributions are that (a) we use an HODLR representation of implicitly defined kernels and propose an approach that leads to it being computed with only $O(\log n)$ forward model evaluations, (b) we use randomized sketching of the off-diagonal blocks and indicate a way to allow for the derivative information to be well approximated in the same structure, and (c) we propose an exact way of computing traces of products of HODLR matrices which in turn allow the computation of the score function and Fischer information matrix for HODLR structures in $O(n \log^2 n)$.

The rest of the paper is structured as follows. In Section 3.2, we review the physics-based covariance model and the maximum likelihood estimation procedure for parameter estimations. Section 3.3 contains a review of HODLR matrices techniques and linear algebra algorithms. We then introduce a matrix-free way, inspired by [3] to construct the HODLR approximation via matrix-vector products of the targeting covariance matrix and differentiating the whole process to get its derivatives. In Section 3.4 we use the obtained hierarchical approximations to derive approximated log-likelihood, score equations and information matrix for parameter estimation. An application of the proposed algorithm to Gaussian wind field model is presented in Section 3.5. We end the paper with a discussion in Section 3.6.

3.2 Physics-based Covariance Models and Maximum Likelihood Estimation

To facilitate the description of our proposed approach, we assume a general physical model to work with. Consider a general deterministic physical model:

$$y = F(z), \tag{3.2.1}$$

where y is an m-dimensional random field and z is an n-dimensional random field. $F : \mathbb{R}^m \to \mathbb{R}^n$ is a sufficiently regular mapping. Here y is the output process (i.e. the process we can partially observe and want to predict) and z is the latent process (i.e. the process that is not usually observable but is strongly correlated with the output process). F can be interpreted as the physical relation that governs the processes. One example is the horizontal wind field model $U = \nabla \times \phi + \nabla \chi$ where U is the horizontal wind component, ϕ is the stream function and χ is the velocity potential [82]. In terms of our model (3.2.1), $y \leftarrow U$ and $z \leftarrow (\phi, \chi)$. This model will be used later in this study.

We note that many physical models can be adapted or converted to this form. For example, many physical processes can be modeled using partial differential equations (PDEs). After proper discretizations the discretized PDE generally takes the form of (3.2.1). One can also consider stochastic partial differential equations (SPDEs), if treating the random term as part of the latent process. Note that we will use the general form (3.2.1) in the following text. However in real applicable cases both the output and latent processes can be a concatenation of several independent processes. Separating independent variables and exploiting their correlations may introduce extra sparsity in the covariance model though we do not explore that avenue here.

3.2.1 Physics-based Covariance Models

In [57], the authors proposed a systematic way to incorporate the physical relation (3.2.1) into the design of covariance structure. Note that although [57] only considers covariance models for spatial processes, the same approach can be adapted to spatio-temporal context with little extra effort.

Suppose we model the latent process as Gaussian processes. Specifically, let's assume $z = (z(x_1), z(x_2), \dots, z(x_m))^T$, where $z(x)$ is a Gaussian random field indexed by a spatial location $x \in \mathbb{R}^d, d \geq 1$. Then z follows a $N(\mu_z, \Sigma_z)$ distribution. Additionally the covariance matrix Σ_z is further parameterized by a covariance function $K_z(\theta)$ which depends on a parameter vector $\theta \in \mathbb{R}^p$. Denote $\mathrm{E}(z) = \bar{z}$ and the perturbation around the mean by $\delta z = z - \bar{z}$. By [57], the covariance model of y satisfies

$$K(\theta) = LK_z(\theta)L^T + O(||\delta z||^3), \tag{3.2.2}$$

where L is the Jacobian matrix of F evaluated at $\bar{z}$. Note that (3.2.2) utilizes the linearization of F which would incur a third-order error in δz. If the function F is linear, the error term vanishes and (3.2.2) becomes an exact covariance model. The covariance model can be applied when the underlying physical relation can be well approximated by its linearization. When the function is highly nonlinear, [57] proposed a higher-order covariance model which can reduce the error to $O(||\delta z||^5)$ order, but due to its complexity we do not pursue that correction here and leave it to future work.

We note that in practice only a very limited fraction of the random field can be observed directly, i.e. $m \gg n$. So the covariance model (3.2.2) is often of full-rank. Approximation methods based on the low-rank structure of the covariance matrix become very inaccurate in this case. Moreover, explicit construction of the covariance matrix becomes very costly since it requires matrix-matrix multiplications with large scale matrices. These facts prompt us to propose a low-rank approximation method for the off-diagonal blocks only in a hierarchical, fully matrix-free, manner that we discuss in §3.3.

3.2.2 Maximum Likelihood Estimation and Parameter Inference

We carry out the estimation and inference by assuming that $y \sim \mathcal{N}(F(\bar{z}), K(\theta))$. If F is a linear mapping the relationship is exact, otherwise, the distribution is only an approximation. We note, however, that the approach is quite common in nonlinear Bayesian inverse problems and extended Kalman filtering [144, 38, 111, 149]. For the rest of the paper we will assume the distribution of y to be the one stated, even if it may be only an approximation in practice. Inferring the parameter vector $\theta = (\theta_1, \ldots, \theta_p)^T \in \mathbb{R}^p$ is of great scientific interest. Under our assumption about y we get the (approximate, for F nonlinear) log-likelihood function:

$$L(\theta) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log|K(\theta)| - \frac{1}{2}(y-\bar{y})^T K(\theta)^{-1}(y-\bar{y}), \tag{3.2.3}$$

where $|A|$ denotes the determinant of square matrix A and $\bar{y} = F(\bar{z})$. The maximum likelihood estimator of θ is the value $\hat{\theta}$ which maximizes (3.2.3). Since the nonzero mean $\bar{y}$ brings only simple algebraic changes in our algorithm, we will assume $\bar{y}$ to be 0 in the rest of the text to simplify the notation. For the same reason, we drop the explicit dependence of K on parameters θ in the rest of the paper.

Evaluating the log-likelihood (3.2.3) requires the evaluation of the log-determinant and the inverse of covariance matrix K (or rather, the linear system solve of K). If the goal is to maximize the log-likelihood, alternatively one can avoid log-determinant computation by considering the score equations, which are obtained by setting the gradient of the log-likelihood function to be zero. The gradient of (3.2.3) with respect to the parameters θ is given by

$$S_j(\theta) = -\frac{1}{2}\mathrm{tr}\left(K^{-1}K_j\right) + \frac{1}{2}y^T K^{-1}K_j K^{-1}y,\ j = 1, \ldots, p, \tag{3.2.4}$$

where $S_j(\theta)$ denotes the derivative of L with respect to parameter θ_j, $K_j = \frac{\partial K}{\partial \theta_j}$ and $\mathrm{tr}(A)$ denotes the trace of square matrix A. One can find the minimizer of log-likelihood (3.2.3) or equivalently the root of the score equations (3.2.4) by quasi-Newton methods, for example, Broyden's method [37] [66], which only requires evaluations of the first-order score equations. However matrix-matrix operations with the inverse of K are still required (even if carried one column at a time) to evaluate the term containing the trace.

Additionally, being able to efficiently estimate the observed Fisher information matrix can facilitate the uncertainty quantification of the given estimators, for example, building the confidence intervals of the estimates. Specifically, denote the maximum likelihood estimator (MLE) of the parameters by $\hat{\theta}$ and the observed Fisher information matrix evaluated at the maximum likelihood estimates by $\mathcal{I}(\hat{\theta})$. As the asymptotic theory suggests [176], if the smallest eigenvalue of $\mathcal{I}$ tends to infinity as the sample size grows, one can expect that

$$(\hat{\theta} - \theta^\star) \xrightarrow{D} \mathcal{N}(0, \mathcal{I}(\hat{\theta})^{-1}), \tag{3.2.5}$$

where $\theta^\star$ represents the ground truth parameter values. We can then construct confidence intervals based on the asymptotic estimation. Furthermore, for multivariate Gaussian processes where the uncertainties only occur in the covariance, each entry in the observed Fisher information matrix is given by

$$\mathcal{I}_{i,j}(\hat{\theta}) = \frac{1}{2}\mathrm{tr}\left[\left(K^{-1}K_i K^{-1}K_j\right)|_{\theta=\hat{\theta}}\right]. \tag{3.2.6}$$

Even for circumstances where (3.2.5) may not be guaranteed to hold, carrying out the maximum likelihood, and estimating the uncertainty in the parameters is very useful since it gives information whether the parameters are estimable in the first place, or whether only some of them may be [60]. This may be the case for the widely-used Matern covariance class, where only some parameters can be guaranteed to be estimable [176]. As asymptotic normality relies on the accuracy of the Taylor expansion of the score equations at the true parameter [60], a well behaved Fischer information matrix is an indication of approximate normality of the parameters being valid.

Unfortunately evaluating (3.2.3), (3.2.4) and (3.2.6) can be very costly. The matrix K (3.2.2) is generally dense (since $F(\cdot)$ may contain an inverse differential operator in cases of interest). A standard Cholesky factorization approach requires $O(n^3)$ computations and $O(n^2)$ memory since the covariance matrix is often unstructured and (3.2.6) requires matrix-matrix operations with the inverse of the covariance and not just solving linear systems with K.

To circumvent these difficulties, in the next section we will utilize hierarchical matrix techniques, specifically HODLR matrices, to reduce both the computational and storage complexity to quasilinear scale.

3.3 HODLR Approximations for Covariance Matrices

In this section, we approximate the covariance matrix K from §3.2 by the HODLR format [4] and define the algorithms that will compute the relevant components efficiently.

3.3.1 HODLR Matrices

The HODLR matrix format uses a divide-and-conquer strategy to recursively divide the whole covariance matrix to sub-blocks and approximate all off-diagonal blocks by low-rank matrices. A typical 2-level symmetric positive definite matrix (SPD; the only kind relevant here due to properties of covariance matrices)-HODLR matrix $A \in \mathbb{R}^{n\times n}$ can be written in the following form:

$$A = \begin{bmatrix} A_1^{(1)} & W_1^{(1)} X_1^{(1)T} \\ V_1^{(1)} U_1^{(1)T} & A_2^{(1)} \end{bmatrix} \tag{3.3.1}$$

$$= \begin{bmatrix} \begin{bmatrix} A_1^{(2)} & W_1^{(2)} X_1^{(2)T} \\ X_1^{(2)} W_1^{(2)T} & A_2^{(2)} \end{bmatrix} & W_1^{(1)} X_1^{(1)T} \\ X_1^{(1)} W_1^{(1)T} & \begin{bmatrix} A_3^{(2)} & W_2^{(2)} X_2^{(2)T} \\ X_2^{(2)} W_2^{(2)T} & A_4^{(2)} \end{bmatrix} \end{bmatrix}, \tag{3.3.2}$$

where the superscripts indicate the level of the approximation. All the off-diagonal blocks are subsequently approximated using low-rank matrices W, X with the appropriate subscripts. For example in the first level, $W_1^{(1)}, X_1^{(1)} \in \mathbb{R}^{n/2 \times k}$ where $k \leq n/2$. The local rank k can be different for different levels (adaptive rank strategy) to guarantee the approximation error of each local block to be upper-bounded by some error tolerance ϵ. Here for the simplicity of the presentation and complexity analysis we assume all local blocks have the same constant local rank k. For a 2-level HODLR matrix, the diagonal blocks $A_1^{(2)}, A_2^{(2)}, A_3^{(2)}, A_4^{(2)}$ in the leaf level (level 2 in this case) are kept to remain full-rank (and identical to the respective sub blocks of the original matrix A).

Taking advantage of the hierarchical structure, an SPD HODLR matrix admits several fast factorization algorithms, including its "basic" factorization introduced in [5], symmetric factorization [6], QR factorization [104], LU factorization [130]. Here we use the "basic" factorization to factorize the HODLR matrix to the product of a sequence of block low-rank updates of the identity matrix:

$$A = \bar{A} \prod_{i=\tau}^{1} (I + U^{(i)} V^{(i)T}), \tag{3.3.3}$$

where $\bar{A}$ is a block-diagonal matrix containing all the leaf level diagonal blocks of A (denoted by $A_i^{(2)}$ in (3.3.2)), I denotes the identity matrix of proper size and each $(I + U^{(i)} V^{(i)T})$ is a block-diagonal low-rank (rank-$2k$) update to the identity, for which the size of the diagonal blocks depends on the level i. Take $(I + U^{(2)} V^{(2)T})$ as an example, we have

$$(I + U^{(2)} V^{(2)T}) = \begin{bmatrix} I + U_1^{(2)} V_1^{(2)T} & 0 \\ 0 & I + U_2^{(2)} V_2^{(2)T} \end{bmatrix}, \tag{3.3.4}$$

with two diagonal blocks. Generally, $(I + U^{(i)} V^{(i)T})$ has 2^{i-1} diagonal blocks with size $n/2^{i-1} \times n/2^{i-1}$. Each diagonal block is a rank $2k$ update to identity. For example in (3.3.4), $U_1^{(2)}, U_2^{(2)}, V_1^{(2)}, V_2^{(2)}$ are all of size $n/2^{i-1} \times 2k$. More details about the factorization can be found in Appendix B.3.

In (3.3.3), τ is the number of levels of the HODLR matrix A. The factorization can be computed in $O(n \log^2 n)$ time if the local rank k is fixed and the number of level grows with $O(\log n)$. We note that an asymmetric HODLR matrix can also be factorized in the form of (3.3.3), but symmetric factorization [6] only exists for SPD HODLR matrices.

Utilizing the structure of the factorization (3.3.3), the determinant of A can be computed efficiently via Sylvester's determinant identity and the linear system can be efficiently solved by recursively applying the Woodbury matrix identity [210]. To summarize, once the decomposition (3.3.3) is obtained, the subsequent numerical linear algebra operations including log-determinant computation, linear system solves and matrix-vector products can all be performed efficiently in at most $O(n \log^2 n)$ operations. A list of common HODLR matrix linear algebra operations with their complexities are given in Table 3.1. For a more complete analysis of supported operations, we refer readers to [77] for a discussion.

Ops.	`A*v`	`A\v`	`det(A)`	`A+B`	`A*B`	`A\B`
Compl.	$O(n \log n)$	$O(n \log^2 n)$	$O(n \log^2 n)$	$O(n \log n)$	$O(n \log^2 n)$	$O(n \log^2 n)$

Table 3.1: Complexity (Compl.) of common arithmetic operations (Ops.). A, B are both HODLR matrices with size n, level $\log n$ and constant local rank. v is a vector of length n.

We note that all operations involving two HODLR matrices in Table 3.1 will lead to increase of off-diagonal ranks. The computational efficiency of the resulting matrix will significantly decrease due to the accumulation of low-rank updates from recursive calls. To tackle the issue, one can combine the arithmetic operations with re-compression to reduce the rank after the computation of each level. Most studies conduct these operations inexactly and heavily rely on the assumptions that the off-diagonal ranks encountered during an operation remain $O(k)$. In this study, we will also encounter operations involving two HODLR matrices. Based on the specific needs of our computation, we derive two new types of HODLR operations which can be conducted exactly and in quasilinear time scale. Details can be found in Section 3.4.2.

In terms of storage complexity, both the HODLR matrix and its factors take $O(n \log n)$ memory. The remaining problem is to construct the HODLR approximation for the given covariance matrix (3.2.2).

3.3.2 Randomized Matrix-free Construction of HODLR Matrices

Assume the SPD matrix $K = LK_zL^T$ in (3.2.2) can be well-approximated by the HODLR format. Note that in our covariance model (3.2.2), $m \gg n$ thus the matrix K_z may be difficult to store. Moreover, the matrix L may be exceedingly difficult to obtain explicitly since the direct model F may be arbitrarily complex, for example a climate code. Therefore, if the task is to construct the HODLR approximation of K, the direct approach dividing the dense covariance matrix recursively and constructing the low-rank approximation for each off-diagonal block (for example, using SVD) is infeasible.

We then turn to the feasibility of computing the HODLR approximation of K by accessing it only by means of matrix-vector products $K\tilde{y}$, which in turn requires efficient access to matrix-vector products with L, K_z, and L^T. Since L is the Jacobian of the operator F, each L-vector and L^T-vector product can be evaluated via forward mode and reverse mode automatic differentiation (AD) [133] with the same order of cost as evaluating a forward solve of the physical model F. The covariance matrix K_z can be structured (e.g. HODLR itself) or sparse, and thus it can be reasonably assumed to allow for fast matrix-vector products. Therefore in many circumstances, L and K_z admits fast and efficient matrix-vector multiplication operations. We refer readers to Section 3.5 for examples.

We now turn to the issue of building the HODLR approximation of K. If one explicitly constructs the covariance matrix K first, this generally takes $O(m)$ forward model evaluations and storing the covariance matrix takes $O(n^2)$ memory, which may be infeasible. An important task, consequently, will be to enable the computation of the HODLR approximation of K in (3.2.2) using much less memory and forward model evaluations. For the rest of the paper, we assume there exists a fast solver of computing matrix-vector products for covariance matrix K with any vector and we track the number of K-vector products for complexity analysis, aiming to get it much smaller than the brute force $O(m)$.

To this end, randomized algorithms for matrix factorizations have proven to be robust and accurate [79]. Here we adapt the idea in [119] to HODLR matrices and propose a way to construct the HODLR approximation of the covariance matrix K using only K-vector multiplications in §3.3.2 and §3.3.2 in conjunction with randomized algorithms to efficiently produce the low-rank approximations to the off-diagonal blocks.

Randomized Matrix factorization

We adopt the simplest randomized matrix factorization algorithm [79] to facilitate our computation and presentation, though other variants of the algorithm including randomized SVD [116] can also be used here with little modification. For a given matrix B and a given target rank k, the algorithm is summarized in Algorithm 6.

Algorithm 6: Randomized Low-rank Approximation

```
**Input**: A matrix $B \in \mathbb{R}^{p\times q}$ and a given target rank $k$, $k < q$.
*Step 1.* Draw a Gaussian random sampling matrix $\Omega \in \mathbb{R}^{p\times k}$.
*Step 2.* Compute $Y = B\Omega$. Now $Y \in \mathbb{R}^{p\times k}$.
*Step 3.* Compute the QR factorization for $Y$ with column pivoting. Denote the Q
  factor by $Q \in \mathbb{R}^{p\times k}$.
*Step 4.* Compute $Q^T B$ by $(B^T Q)^T$.
**Output**: $\hat{B} = Q(Q^T B)$ is a rank-$k$ approximation of $B$.
```

The accuracy of the algorithm and its variants has been well-studied [79, 211, 116]. When using the matrix 2-norm to measure the approximation error, the minimal error is σ_{k+1} which is the $(k+1)$-th largest singular value and the lower bound is achieved by the singular vector decomposition (SVD) truncated to the rank k. It can be shown that the approximation error can be bounded by k, c and σ_k. Specifically, when the singular values of B decay rapidly, Algorithm 6 can provide an approximation that is very close to the optimal truncated SVD solution with high probability [211].

Construction From Matrix-vector Products

To discuss our construction of the HODLR matrix approximation (3.3.2), using Algorithm 6 and employing only $O(\log_2(n)k)$ matrix-vector multiplications, we follow the approach in [129]. The idea is to compress the off-diagonal blocks via randomized sampling. In this subsection we review the HODLR compression procedure to lay the foundation for the next several subsections. Note that our approach goes beyond [129] by further considering the derivatives of the construction procedure.

Assume we work with a two-level covariance matrix K:

$$K = \begin{bmatrix} \begin{bmatrix} A_1^{(2)} & B_1^{(2)} \\ B_1^{(2)T} & A_2^{(2)} \end{bmatrix} & B_1^{(1)} \\ B_1^{(1)T} & \begin{bmatrix} A_3^{(2)} & B_2^{(2)} \\ B_2^{(2)T} & A_4^{(2)} \end{bmatrix} \end{bmatrix}. \tag{3.3.5}$$

While the algorithm does not have requirements about the matrix size, we will assume n is divisible by 4 for simplifying the presentation.

Processing level 1. Assume the row index set and the column index set of $B_1^{(1)}$ are given by $\mathcal{I}_1^{(1)}$ $\left(\triangleq 1 : \frac{n}{2}\right)$ and $\mathcal{I}_2^{(1)}$ $\left(\triangleq \left(\frac{n}{2}+1\right) : n\right)$ respectively. We draw a $(n/2) \times k$ sampling matrix $R_1^{(1)}$ with i.i.d standard normal entries. Then we construct a patterned $n \times k$ sampling matrix by filling $R^{(1)}(\mathcal{I}_1^{(1)}, :)$ with 0 and $R^{(1)}(\mathcal{I}_2^{(1)}, :) = R_1^{(1)}$. Here we use the MATLAB notation $R^{(1)}(\mathcal{I}_1^{(1)}, :)$ to indicate the selected rows from $R^{(1)}$ according to the index set $\mathcal{I}_1^{(1)}$. Now we observe that $R^{(1)^T} = \begin{bmatrix} 0 & R_1^{(1)^T} \end{bmatrix}$ and thus

$$KR^{(1)} = K \begin{bmatrix} 0 \\ R_1^{(1)} \end{bmatrix} = \begin{bmatrix} B_1^{(1)} R_1^{(1)} \\ \begin{bmatrix} A_3^{(2)} & B_2^{(2)} \\ B_2^{(2)T} & A_4^{(2)} \end{bmatrix} R_1^{(1)} \end{bmatrix}. \tag{3.3.6}$$

By restricting the row index of the right-hand side to index set $\mathcal{I}_1^{(1)}$ and discarding the bottom half, we get $B_1^{(1)} R_1^{(1)}$ as the randomly sampled column space of $B_1^{(1)}$. The result corresponds to steps 1 and 2 in Algorithm 6. For carrying out step 3, we compute the QR factorization of $B_1^{(1)} R_1^{(1)}$. Denote the orthogonal basis matrix by $Q_1^{(1)} \in \mathbb{R}^{\frac{n}{2} \times k}$. To compute $Q_1^{(1)T} B_1^{(1)}$ we form another patterned matrix $S^{(1)} \in \mathbb{R}^{n \times k}$ by filling $S^{(1)}(\mathcal{I}_2^{(1)}, :)$ with 0 and

$S^{(1)}(\mathcal{I}_1^{(1)},:) = Q_1^{(1)}$. Note that $S^{(1)^T} = \begin{bmatrix} Q_1^{(1)^T} 0 \end{bmatrix}$ and

$$KS^{(1)} = K \begin{bmatrix} Q_1^{(1)} \\ 0 \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} A_1^{(2)} & B_1^{(2)} \\ B_1^{(2)T} & A_2^{(2)} \end{bmatrix} Q_1^{(1)} \\ B_1^{(1)T} Q_1^{(1)} \end{bmatrix}. \tag{3.3.7}$$

Likewise, we restrict the output to the row index set $\mathcal{I}_2^{(1)}$ and discard the first half to obtain $\tilde{B}_1^{(1)} \triangleq B_1^{(1)T} Q_1^{(1)}$. We then obtain *the low-rank (rank k) representation of the off-diagonal block*

$$\hat{B}_1^{(1)} = Q_1^{(1)} \tilde{B}_1^{(1)^T} = Q_1^{(1)} (Q_1^{(1)T} B_1^{(1)}). \tag{3.3.8}$$

Processing level 2. Next we move to the blocks of the second level in the hierarchy, i.e. $B_1^{(2)}$ and $B_2^{(2)}$. We draw a patterned sampling matrix of $R^{(2)} = (0, R_1^{(2)T}, 0, R_2^{(2)T})^T \in \mathbb{R}^{n\times k}$, with $R_i^{(2)} \in \mathbb{R}^{\frac{n}{4}\times k}$, $i = 1, 2$, having Gaussian entries. Here, the nonzero rows of $R^{(2)}$ correspond to column indices of $B_1^{(2)}$ $\left(\triangleq \left(\frac{n}{4}+1\right) : \frac{n}{2}\right)$ and $B_2^{(2)}$ $\left(\triangleq \left(\frac{3n}{4}+1\right) : n\right)$ in K. We observe that

$$\left(K - \begin{bmatrix} 0 & \hat{B}_1^{(1)} \\ \hat{B}_1^{(1)T} & 0 \end{bmatrix}\right) \begin{bmatrix} 0 \\ R_1^{(2)} \\ 0 \\ R_2^{(2)} \end{bmatrix} \approx \begin{bmatrix} \begin{bmatrix} A_1^{(2)} & B_1^{(2)} \\ B_1^{(2)T} & A_2^{(2)} \end{bmatrix} & 0 \\ 0 & \begin{bmatrix} A_3^{(2)} & B_2^{(2)} \\ B_2^{(2)T} & A_4^{(2)} \end{bmatrix} \end{bmatrix} \begin{bmatrix} 0 \\ R_1^{(2)} \\ 0 \\ R_2^{(2)} \end{bmatrix} = \begin{bmatrix} B_1^{(2)} R_1^{(2)} \\ A_2^{(2)} R_1^{(2)} \\ B_2^{(2)} R_2^{(2)} \\ A_4^{(2)} R_2^{(2)} \end{bmatrix}. \tag{3.3.9}$$

Note that it is impossible to directly calculate the matrix subtraction on the left-hand side of (3.3.9) since the matrix K is not explicitly available. Instead, we access K only via matrix-vector products. Furthermore, we only store the obtained low-rank factors of $\hat{B}_1^{(1)}$ explicitly. The $\hat{B}_1^{(1)}$-vector or $\hat{B}_1^{(1)}$-matrix products are done by sequentially applying both low-rank factors ($Q_1^{(1)}$ and $\tilde{B}_1^{(1)^T}$) to the target matrix or vector.

We form the new patterned matrix $S^{(2)} = (Q_1^{(2)T}, 0, Q_2^{(2)T}, 0)^T \in \mathbb{R}^{n\times k}$, where the positioning is such that the zero blocks are square. Then we can compute

$$\left(K - \begin{bmatrix} 0 & \hat{B}_1^{(1)} \\ \hat{B}_1^{(1)T} & 0 \end{bmatrix}\right) \begin{bmatrix} Q_1^{(2)} \\ 0 \\ Q_2^{(2)} \\ 0 \end{bmatrix} \approx \begin{bmatrix} \begin{bmatrix} A_1^{(2)} & B_1^{(2)} \\ B_1^{(2)T} & A_2^{(2)} \end{bmatrix} & 0 \\ 0 & \begin{bmatrix} A_3^{(2)} & B_2^{(2)} \\ B_2^{(2)T} & A_4^{(2)} \end{bmatrix} \end{bmatrix} \begin{bmatrix} Q_1^{(2)} \\ 0 \\ Q_2^{(2)} \\ 0 \end{bmatrix} = \begin{bmatrix} A_1^{(2)} Q_1^{(2)} \\ B_1^{(2)T} Q_1^{(2)} \\ A_3^{(2)} Q_2^{(2)} \\ B_2^{(2)T} Q_2^{(2)} \end{bmatrix}. \tag{3.3.10}$$

Similar to the first step, we now extract the even block entries from the right-hand side of (3.3.10), that is the ones corresponding to rows indices $\left(\frac{n}{4}+1\right) : \frac{n}{2}$, $\tilde{B}_1^{(2)} = B_1^{(2)T} Q_1^{(2)}$, and $\left(\frac{3n}{4}+1\right) : n$, $\tilde{B}_2^{(2)} = B_2^{(2)^T} Q_2^{(2)}$. We then obtain *the low rank representation of the off-diagonal blocks on the second level*

$$\hat{B}_1^{(2)} = Q_1^{(2)} \tilde{B}_1^{(2)^T} = Q_1^{(2)} (Q_1^{(2)T} B_1^{(2)}), \quad \hat{B}_2^{(2)} = Q_2^{(2)} \tilde{B}_2^{(2)^T} = Q_2^{(2)} (Q_2^{(2)T} B_2^{(2)}). \tag{3.3.11}$$

If there are finer levels, we can proceed to the next level in a similar fashion by first "removing" all off-diagonal blocks from the previous levels.

Processing the leaf level. To sketch the leaf level $A_1^{(2)}, A_2^{(2)}, A_3^{(2)}, A_4^{(2)}$, we construct sampling matrix S of size $n \times \text{size}(\mathrm{A}_1^{(2)})$ which is a vertical concatenation of identity matrices, $S = \left[I_{\frac{n}{4}}, I_{\frac{n}{4}}, I_{\frac{n}{4}}, I_{\frac{n}{4}}\right]^T$. We will exploit the fact that we have an approximation of all level 1 and 2 off-diagonal blocks,

$$\left(K - \begin{bmatrix} \begin{bmatrix} 0 & \hat{B}_1^{(2)} \\ \hat{B}_1^{(2)T} & 0 \end{bmatrix} & \hat{B}_1^{(1)} \\ \hat{B}_1^{(1)T} & \begin{bmatrix} 0 & \hat{B}_2^{(2)} \\ \hat{B}_2^{(2)T} & 0 \end{bmatrix} \end{bmatrix} \right) S$$

$$\approx \begin{bmatrix} \begin{bmatrix} A_1^{(2)} & 0 \\ 0 & A_2^{(2)} \end{bmatrix} & 0 \\ 0 & \begin{bmatrix} A_3^{(2)} & 0 \\ 0 & A_4^{(2)} \end{bmatrix} \end{bmatrix} \begin{bmatrix} I_{\frac{n}{4}} \\ I_{\frac{n}{4}} \\ I_{\frac{n}{4}} \\ I_{\frac{n}{4}} \end{bmatrix} = \begin{bmatrix} A_1^{(2)} \\ A_2^{(2)} \\ A_3^{(2)} \\ A_4^{(2)} \end{bmatrix}. \tag{3.3.12}$$

Asymptotic complexity. Assume all off-diagonal blocks have the same rank k and the number of levels is $\tau = log_2\left(\frac{n}{n_l}\right)$, where n_l is the size of the leaf block. Overall, the entire procedure requires $O(k\tau)$ K-vector products and $O(nk^2\tau^2)$ time complexity. If we assume constant off-diagonal rank $k = O(1)$ and the number of levels grows as $O(\log n)$, for example $\tau = \lfloor \log_2\left(\frac{n}{k}\right) \rfloor$, the computational complexity is $O(\log n)$ K-vector products and $O(n \log^2 n)$ complexity. The storage complexity is $O(n \log n)$. More details about the complexity analysis can be found in [129, Section 4.1]. These assumptions are consistent with the assumptions in [6, 129] for complexity analysis.

3.3.3 Differentiating the HODLR Approximation

With the HODLR approximation of the covariance matrix K at hand, we can now handle the log-determinant term and the matrix inverse term of K in the log-likelihood (3.2.3) and score (3.2.4) efficiently. However we now need to compute or produce an adequate approximation of K_j. To this end, we will also approximate the derivatives of the covariance matrix in HODLR format. We will explore the fact that our HODLR approximation consists of a sequence of covariance matrix-vector products, block subsettings, and QR factorizations. The entire process is differentiable, as long as the patterned sampling matrices (e.g. $S^{(1)}$, $S^{(2)}$, $R^{(1)}$, $R^{(2)}$ in §3.3.2) and subsetting patterns are kept fixed. We thus differentiate the approximation algorithm while simultaneously constructing the HODLR approximation for both the covariance matrix and its derivatives.

As an illustrative example, we follow the same framework as in §3.3.2 and approximate K and its derivatives to 2-level HODLR form.

Processing level 1. Starting from the first level we differentiate (3.3.6) with respect to any parameter θ_j using forward mode AD [133] with the same order of cost as evaluating the K-vector product. Recall that we keep R^1 fixed, that is independent of θ. In other words we draw $R^{(1)}$ once and keep it fixed for all optimization iterations. Differentiating through the subsetting operation (the top matrix on level 1 in §3.3.2) we obtain the identity

$$\frac{\partial B_1^{(1)} R^{(1)}}{\partial \theta_j} = \frac{\partial K R^{(1)}}{\partial \theta_j} (\mathcal{I}_1^{(1)}, :). \tag{3.3.13}$$

In the next step we differentiate the QR factorization of $B_1^{(1)} R_1^{(1)}$, specifically the Q factor $Q_1^{(1)}$ as summarized in Algorithm 7, [46], to obtain $\frac{\partial Q_1^{(1)}}{\partial \theta_j}$.

At the next step of the Algorithm from §3.3, we compute $B_1^{(1)T} Q_1^{(1)}$ via (3.3.7). By differentiating the K-vector product and keeping in mind the dependence between $Q_1^{(1)}$ and parameters θ_j, we can obtain $\frac{\partial B_1^{(1)T} Q_1^{(1)}}{\partial \theta_j}$,

$$\frac{\partial B_1^{(1)T} Q_1^{(1)}}{\partial \theta_j} = \left(\frac{\partial B_1^{(1)}}{\partial \theta_j} \right)^T Q_1^{(1)} + B_1^{(1)T} \frac{\partial Q_1^{(1)}}{\partial \theta_j}, \tag{3.3.18}$$

$$\overset{(3.3.7),(3.3.5)}{=} \frac{\partial K S^{(1)}}{\partial \theta_j} (\mathcal{I}_2^{(1)}, :) + K \begin{bmatrix} \frac{\partial Q_1^{(1)}}{\partial \theta_j} \\ 0 \end{bmatrix} (\mathcal{I}_2^{(1)}, :). \tag{3.3.19}$$

Algorithm 7: Differentiate the QR factorization [46]

Input: A full column rank matrix $B \in \mathbb{R}^{p\times q}$ where $p > q$. Assume the (compact) QR factorization of B is given by

$$B = Q_1 R, \quad Q_1 \in \mathbb{R}^{p\times q},\ Q_1^T Q_1 = I_q,\ R \in \mathbb{R}^{q\times q} \tag{3.3.14}$$

Also assume dB is known, the differentiation can be computed in the following steps.

Step 1. Form $Y = Q_1^T dBR^{-1}$.

Step 2. Let $d\Omega$ be a $q \times q$ skew-symmetric matrix. Notice that both R and its differentiation dR are upper triangular matrices. Therefore $d\Omega$ and dR can be uniquely computed from the following identity (derived from the product rule $dB = dQ_1R + Q_1dR$ inserted in Y)

$$Y = d\Omega + dRR^{-1}. \tag{3.3.15}$$

Step 3. Compute

$$\begin{aligned} dQ_1 &= Q_1 d\Omega + Q_2Q_2^T dBR^{-1} && (3.3.16)\\ &= Q_1 d\Omega + (I - Q_1Q_1^T)dBR^{-1}. && (3.3.17) \end{aligned}$$

Output: dQ_1, dR.

Note that we cannot compute (3.3.18) directly since the kernel matrix K is not explicitly available. Instead we use the same trick as in (3.3.7) and rewrite both terms with K-vector products as in (3.3.19). Specifically we can compute the first term in (3.3.19) via AD assuming the factor $Q_1^{(1)}$ is held as a constant, as is the case for $S^{(1)}$ in (3.3.7). The second term can be computed by k K-vector products with derivatives $\frac{\partial Q_1^{(1)}}{\partial \theta_j}$.

Recall we approximate $B_1^{(1)}$ by low-rank factorization $\hat{B}_1^{(1)} = Q_1^{(1)}(Q_1^{(1)T}B_1^{(1)})$, (3.3.8). We differentiate both sides by parameter θ_j to obtain *the low-rank representation of the derivative of the first-level off-diagonal block, $B_1^{(1)}$:*

$$\frac{\partial \hat{B}_1^{(1)}}{\partial \theta_j} = \frac{\partial Q_1^{(1)}}{\partial \theta_j}\left(B_1^{(1)T}Q_1^{(1)}\right)^T + Q_1^{(1)}\left(\frac{\partial B_1^{(1)T}Q_1^{(1)}}{\partial \theta_j}\right)^T. \tag{3.3.20}$$

We now summarize the workflow that we used to efficiently compute the derivative of the low approximation since we will apply this philosophy at each level:

1. We differentiate (3.3.6), we get $\frac{\partial B_1^{(1)} R_1^{(1)}}{\partial \theta_j}$, (3.3.13), using differentiation of k K-vector products.

2. We use Algorithm 7 to differentiate the QR factorization of $B_1^{(1)} R_1^{(1)}$. We obtain $\frac{\partial Q_1^{(1)}}{\partial \theta_j}$.

3. We differentiate (3.3.7), exploit the HODLR structure (3.3.5) and obtain the first term in (3.3.19).This steps requires differentiation of k K-vector products. Then we use another k K-vector products with the derivatives of the Q-factors to form the second term of (3.3.19). Combining two terms together, we obtain $\frac{\partial B_1^{(1)T} Q_1^{(1)}}{\partial \theta_j}$.

4. The derivative of the low-rank approximation, $\frac{\partial \hat{B}_1^{(1)}}{\partial \theta_j}$, is available by means of the right hand side of (3.3.20).

Note that both terms in (3.3.20) are of rank k, which means the rank of $\frac{\partial \hat{B}_1^{(1)}}{\partial \theta_j}$ is at most $2k$. In practice we never construct (3.3.20) explicitly. Instead we store the low-rank components of both terms in (3.3.20) and invoke them when necessary. Also note that our workflow requires $3k$ matrix-vector products with either the covariance matrix K or the covariance matrix derivative K_j and nk storage for the components (two $\frac{n}{2} \times k$ blocks).

Processing level 2. The second-level procedure is the same as for the first level except the approximations from level 1 need to be removed. We start by differentiating (3.3.9),

$$\left(K_j - \begin{bmatrix} 0 & \frac{\partial \hat{B}_1^{(1)}}{\partial \theta_j} \\ \frac{\partial \hat{B}_1^{(1)T}}{\partial \theta_j} & 0 \end{bmatrix}\right) \begin{bmatrix} 0 \\ R_1^{(2)} \\ 0 \\ R_2^{(2)} \end{bmatrix} = \frac{\partial}{\partial \theta_j}\left(K \begin{bmatrix} 0 \\ R_1^{(2)} \\ 0 \\ R_2^{(2)} \end{bmatrix}\right) - \begin{bmatrix} \frac{\partial \hat{B}_1^{(1)}}{\partial \theta_j} \begin{bmatrix} 0 \\ R_2^{(2)} \end{bmatrix} \\ \frac{\partial \hat{B}_1^{(1)T}}{\partial \theta_j} \begin{bmatrix} 0 \\ R_1^{(2)} \end{bmatrix} \end{bmatrix} \approx \begin{bmatrix} \frac{\partial B_1^{(2)} R_1^{(2)}}{\partial \theta_j} \\ \frac{\partial A_2^{(2)} R_1^{(2)}}{\partial \theta_j} \\ \frac{\partial B_2^{(2)} R_2^{(2)}}{\partial \theta_j} \\ \frac{\partial A_4^{(2)} R_2^{(2)}}{\partial \theta_j} \end{bmatrix}. \tag{3.3.21}$$

Note that we do not explicitly differentiate the matrices on the left-hand side of (3.3.21) since we do not have direct access to matrix K. Instead we differentiate the matrix-vector products in (3.3.9) (see the expression in the middle). The first term in the middle expression is computed via AD by differentiating the K-vector products. The second term is computed by matrix-vector products with the low-rank representation of $\frac{\partial \hat{B}_1^{(1)}}{\partial \theta_j}$ (3.3.20). Now we obtain $\frac{\partial B_1^{(2)} R_1^{(2)}}{\partial \theta_j}$ and $\frac{\partial B_2^{(2)} R_1^{(2)}}{\partial \theta_j}$ by truncating the right-hand side of (3.3.21). We use these low-rank matrices in Algorithm 7 and obtain the derivatives of the Q-factors with respect to the parameters: $\frac{\partial Q_1^{(2)}}{\partial \theta_j}$, $\frac{\partial Q_2^{(2)}}{\partial \theta_j}$.

Next, we differentiate (3.3.10) (reversing the order of terms in that equation to make our argument) by treating $Q_1^{(2)}$, $Q_2^{(2)}$, components of $S^{(2)}$, as constant matrices,

$$\begin{bmatrix} \frac{\partial A_1^{(2)}}{\partial \theta_j} Q_1^{(2)} \\ \frac{\partial B_1^{(2)T}}{\partial \theta_j} Q_1^{(2)} \\ \frac{\partial A_3^{(2)}}{\partial \theta_j} Q_2^{(2)} \\ \frac{\partial B_2^{(2)T}}{\partial \theta_j} Q_2^{(2)} \end{bmatrix} \overset{(3.3.10),(3.3.5)}{\approx} \left(K_j - \begin{bmatrix} 0 & \frac{\partial \hat{B}_1^{(1)}}{\partial \theta_j} \\ \frac{\partial \hat{B}_1^{(1)T}}{\partial \theta_j} & 0 \end{bmatrix} \right) \begin{bmatrix} Q_1^{(2)} \\ 0 \\ Q_2^{(2)} \\ 0 \end{bmatrix}$$

$$= \frac{\partial}{\partial \theta_j} \left(K \begin{bmatrix} Q_1^{(2)} \\ 0 \\ Q_2^{(2)} \\ 0 \end{bmatrix} \right) - \begin{bmatrix} \frac{\partial \hat{B}_1^{(1)T}}{\partial \theta_j} \begin{bmatrix} Q_2^{(2)} \\ 0 \end{bmatrix} \\ \frac{\partial \hat{B}_1^{(1)}}{\partial \theta_j} \begin{bmatrix} Q_1^{(2)} \\ 0 \end{bmatrix} \end{bmatrix}. \qquad (3.3.22)$$

In the last term, the first component only is the one requiring access to K and the real computation weight. It can be computed via AD by differentiating K-vector products and treating $Q_1^{(2)}$, $Q_2^{(2)}$ as constant matrices. For the second component of the last term we do matrix-vector products with the low-rank representation of $\frac{\partial \hat{B}_1^{(1)}}{\partial \theta_j}$ (3.3.20) . Furthermore, by replacing $Q_1^{(2)}$, $Q_2^{(2)}$ in (3.3.10) with $\frac{\partial Q_1^{(2)}}{\partial \theta_j}$, $\frac{\partial Q_2^{(2)}}{\partial \theta_j}$, reverting its order, and using (3.3.5) we have

$$\begin{bmatrix} A_1^{(2)} \frac{\partial Q_1^{(2)}}{\partial \theta_j} \\ B_1^{(2)T} \frac{\partial Q_1^{(2)}}{\partial \theta_j} \\ A_3^{(2)} \frac{\partial Q_2^{(2)}}{\partial \theta_j} \\ B_2^{(2)T} \frac{\partial Q_2^{(2)}}{\partial \theta_j} \end{bmatrix} \approx \left(K - \begin{bmatrix} 0 & \hat{B}_1^{(1)} \\ \hat{B}_1^{(1)T} & 0 \end{bmatrix} \right) \begin{bmatrix} \frac{\partial Q_1^{(2)}}{\partial \theta_j} \\ 0 \\ \frac{\partial Q_2^{(2)}}{\partial \theta_j} \\ 0 \end{bmatrix}. \tag{3.3.23}$$

Now we subset the left-hand side of (3.3.22) and (3.3.23) to the terms involving $B_1^{(2)}, B_2^{(2)}$ and use the product rule to compute

$$\frac{\partial (B_1^{(2)T} Q_1^{(2)})}{\partial \theta_j} = \frac{\partial B_1^{(2)T}}{\partial \theta_j} Q_1^{(2)} + B_1^{(2)T} \frac{\partial Q_1^{(2)}}{\partial \theta_j}, \tag{3.3.24}$$

$$\frac{\partial (B_2^{(2)T} Q_2^{(2)})}{\partial \theta_j} = \frac{\partial B_2^{(2)T}}{\partial \theta_j} Q_2^{(2)} + B_2^{(2)T} \frac{\partial Q_2^{(2)}}{\partial \theta_j}. \tag{3.3.25}$$

We apply the product rule to (3.3.11) to obtain *the low-rank representations of the derivatives of the second level off-diagonal blocks:*

$$\frac{\partial \hat{B}_1^{(2)}}{\partial \theta_j} = \frac{\partial Q_1^{(2)}}{\partial \theta_j} \left(B_1^{(2)T} Q_1^{(2)} \right)^T + Q_1^{(2)} \left(\frac{\partial (B_1^{(2)T} Q_1^{(2)})}{\partial \theta_j} \right)^T, \tag{3.3.26}$$

$$\frac{\partial \hat{B}_2^{(2)}}{\partial \theta_j} = \frac{\partial Q_2^{(2)}}{\partial \theta_j} \left(B_2^{(2)T} Q_2^{(2)} \right)^T + Q_2^{(2)} \left(\frac{\partial (B_2^{(2)T} Q_2^{(2)})}{\partial \theta_j} \right)^T. \tag{3.3.27}$$

To access this representation, we store the low-rank components in the right-hand side. For example to store (3.3.26), we store $\frac{\partial Q_1^{(2)}}{\partial \theta_j}$, $(B_1^{(2)T} Q_1^{(2)})$ as the two low-rank factors for the first term and $Q_1^{(2)}$, $\left(\frac{\partial B_1^{(2)T} Q_1^{(2)}}{\partial \theta_j} \right)$ as the two low-rank factors for the second term. All these matrices are of rank k resulting in a total rank of $2k$. This computation requires k K-vector products with the derivatives of the Q components of the QR factorization and $2k$ K-vector products with the partial derivative K_j, and a storage of $4\ \frac{n}{4} \times k$ blocks, or equivalently one $n \times k$ block.

Processing the leaf level. For the leaf level, the derivative of all the diagonal blocks can be estimated simultaneously by differentiating (3.3.12),

$$\left(K_j - \begin{bmatrix} \begin{bmatrix} 0 & \frac{\partial \hat{B}_1^{(2)}}{\partial \theta_j} \\ \frac{\partial \hat{B}_1^{(2)T}}{\partial \theta_j} & 0 \end{bmatrix} & \frac{\partial \hat{B}_1^{(1)}}{\partial \theta_j} \\ \frac{\partial \hat{B}_1^{(1)T}}{\partial \theta_j} & \begin{bmatrix} 0 & \frac{\partial \hat{B}_2^{(2)}}{\partial \theta_j} \\ \frac{\partial \hat{B}_2^{(2)T}}{\partial \theta_j} & 0 \end{bmatrix} \end{bmatrix} \right) S \approx \begin{bmatrix} \frac{\partial A_1^{(2)}}{\partial \theta_j} \\ \frac{\partial A_2^{(2)}}{\partial \theta_j} \\ \frac{\partial A_3^{(2)}}{\partial \theta_j} \\ \frac{\partial A_4^{(2)}}{\partial \theta_j} \end{bmatrix}. \tag{3.3.28}$$

We access matrix K_j via matrix-vector products only. We can write

$$K_j S = \frac{\partial (KS)}{\partial \theta_j}. \tag{3.3.29}$$

Recall that $S = \left[I_{\frac{n}{4}}, I_{\frac{n}{4}}, I_{\frac{n}{4}}, I_{\frac{n}{4}} \right]^T$. We can explicitly write the second term as

$$\begin{bmatrix} \begin{bmatrix} 0 & \frac{\partial \hat{B}_1^{(2)}}{\partial \theta_j} \\ \frac{\partial \hat{B}_1^{(2)T}}{\partial \theta_j} & 0 \end{bmatrix} & \frac{\partial \hat{B}_1^{(1)}}{\partial \theta_j} \\ \frac{\partial \hat{B}_1^{(1)T}}{\partial \theta_j} & \begin{bmatrix} 0 & \frac{\partial \hat{B}_2^{(2)}}{\partial \theta_j} \\ \frac{\partial \hat{B}_2^{(2)T}}{\partial \theta_j} & 0 \end{bmatrix} \end{bmatrix} S = \begin{bmatrix} \begin{bmatrix} \frac{\partial \hat{B}_1^{(2)}}{\partial \theta_j} I_{\frac{n}{4}} \\ \frac{\partial \hat{B}_1^{(2)T}}{\partial \theta_j} I_{\frac{n}{4}} \end{bmatrix} + \frac{\partial \hat{B}_1^{(1)}}{\partial \theta_j} \begin{bmatrix} I_{\frac{n}{4}} \\ I_{\frac{n}{4}} \end{bmatrix} \\ \frac{\partial \hat{B}_1^{(1)T}}{\partial \theta_j} \begin{bmatrix} I_{\frac{n}{4}} \\ I_{\frac{n}{4}} \end{bmatrix} + \begin{bmatrix} \frac{\partial \hat{B}_2^{(2)}}{\partial \theta_j} I_{\frac{n}{4}} \\ \frac{\partial \hat{B}_2^{(2)T}}{\partial \theta_j} I_{\frac{n}{4}} \end{bmatrix} \end{bmatrix}. \tag{3.3.30}$$

The $n \times k$ right hand side is computed using the low-rank representations of the off-diagonal blocks from (3.3.20), (3.3.26), and (3.3.27), respectively. Taking the second term of the upper block for an example, we get

$$\frac{\partial \hat{B}_1^{(1)}}{\partial \theta_j} \begin{bmatrix} I_{\frac{n}{4}} \\ I_{\frac{n}{4}} \end{bmatrix} = \frac{\partial Q_1^{(1)}}{\partial \theta_j} \left(B_1^{(1)T} Q_1^{(1)} \right)^T \begin{bmatrix} I_{\frac{n}{4}} \\ I_{\frac{n}{4}} \end{bmatrix} + Q_1^{(1)} \left(\frac{\partial B_1^{(1)T} Q_1^{(1)}}{\partial \theta_j} \right)^T \begin{bmatrix} I_{\frac{n}{4}} \\ I_{\frac{n}{4}} \end{bmatrix}. \tag{3.3.31}$$

The computation is done using the representation on the right by computing the rightmost factor in each term using k^2 inner products, then multiplying with the resulting $k \times k$ matrix the first factor in each of the two terms above. The effort is linear with the number of rows, though it is increasing linearily with level order, i.e. $O(nk^2 l)$.

Therefore for each θ_j, we can construct the HODLR approximation of the derivative matrix K_j with off-diagonal local rank of at most $2k$ at each level. Since the complexity of differentiating the K-vector product using AD is linear in the complexity of evaluating the K-vector product itself and we need $2k$ differentiations of K-vector products plus k extra K-vector products with the derivatives of the orthogonal columns at each level, the additional access to the matrix K is $O(k\tau)$ K-vector products. In terms of extra computations, the complexity is dominated by removing all lower level off-diagonal approximations in K-vector products. Following the similar complexity analysis in §3.3.2, the computational complexity is given by $O(nk^2\tau^2)$. The extra storage complexity is $O(nk\tau)$ due to the off-diagonal low-rank components.

Note that we need to repeat the process for all parameters $\theta_1, \ldots, \theta_p$. In summary, to obtain the HODLR approximations of the derivative matrices of K with respect to all parameters, the extra complexity is $O(pk\tau)$ K-vector products, $O(pnk^2\tau^2)$ time and $O(pnk\tau)$ memory.

Similarly if assuming constant off-diagonal rank $k = O(1)$ and the number of levels grows as $O(\log n)$, for example $\tau = \lfloor \log_2 (\frac{n}{k}) \rfloor$, the computational complexity is $O(p \log n)$ K-vector products and $O(pn \log^2 n)$ complexity. The storage complexity is $O(pn \log n)$.

3.4 Hierarchical Approximations of Gaussian Likelihood, Score Equations, Information Matrices

With the HODLR approximations of the covariance matrix and its derivatives at hand, we can then efficiently approximate the Gaussian log-likelihood function, the score equations and the observed Fisher information matrix.

3.4.1 Approximated Gaussian Likelihood

Assuming a constant rank k, the HODLR approximation $\tilde{K}$ requires $O(\log n)$ levels. Such a structure admits an exact factorization with $O(n \log^2 n)$ computational complexity [6]. The resulting approximation of the exact log-likelihood function defined in (3.2.3) is denoted by $\tilde{L}(\theta)$:

$$\tilde{L}(\theta) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log|\tilde{K}| - \frac{1}{2}(y-\bar{y})^T\tilde{K}^{-1}(y-\bar{y}). \tag{3.4.1}$$

As we discussed in Section 3.3.1, the determinant and linear system can both be solved in $O(n \log^2 n)$ complexity. Therefore given the factorization of $\tilde{K}$, the approximate log-likelihood (3.4.1) can be evaluated in $O(n \log^2 n)$ time.

3.4.2 Trace Computation

When evaluating the score equations, since both K and K_j have been approximated in HODLR format, the second term in (3.2.4) poses no difficulty. The bottleneck is obviously the trace of matrix products. One option for it is the use of stochastic trace estimators which converts the trace operation to a sequence of matrix-vector products with random vectors. Several choices are available including Gaussian trace estimator, Hutchinson's trace estimator, unit vector trace estimator; we refer readers to [10] for a discussion and convergence analysis. Given a relative error tolerance ϵ, generally it requires $O(\epsilon^{-2})$ random samples to achieve that accuracy.

Another option is to solve the matrix product explicitly. As we discussed in Table 3.3.1, computing $A^{-1}B$ for two HODLR matrices A, B have been studied and can be conducted in $O(n \log^2 n)$ time. However as arithmetic operations often increase the HODLR ranks, it is a common practice to combine the operations with recompression and perform low-rank truncation for each off-diagonal block to control the local rank growth. However in this case the operations incur additional error (which may be acceptable given the HODLR approximation error itself but we would like to explore avoiding it).

The third option is to take the advantage of the properties of the trace operation to simplify the computations. Based on the expression of (3.2.4) and (3.2.6) we introduce two operations $\mathrm{tr}(A^{-1}B)$ and $\mathrm{tr}(A^{-1}BC^{-1}D)$ given A, B, C, D HODLR matrices. Both operations can be conducted exactly and in quasilinear scale. The detailed algorithms are included in Appendix B.2. In this work, we will use the third option when computing the score equations and Fisher information matrix associated with the approximate log-likelihood (3.4.1).

3.4.3 Approximated Score Equations And Fisher Information Matrix

Now consider the score equations (3.2.4). The score equations of the approximate likelihood are given by

$$\tilde{S}_j(\theta) = -\frac{1}{2}\mathrm{tr}\left(\tilde{K}^{-1}\tilde{K}_j\right) + \frac{1}{2}y^T\tilde{K}^{-1}\tilde{K}_j\tilde{K}^{-1}y. \tag{3.4.2}$$

Recall $\tilde{K}_j$ is also an HODLR matrix with rank $3k$ (3.3.26). Utilizing the proposed trace operation in B.2, we can now evaluate the trace term exactly in $O(n \log^2 n)$ scale. For the second term we compute $\tilde{K}^{-1}y$ using the factorization of $\tilde{K}$ and the rest are HODLR matrix-vector products. Repeat the computation for all p parameters and we get that the approximated score equations (3.4.2) can be evaluated in an extra $O(pn\log^2 n)$ time.

With the ability to efficiently approximate the log-likelihood and the score equations, we can then apply optimization algorithms to obtain the maximum likelihood estimates of the parameters. Note that the parameter values are updated at each optimization iteration. Therefore new HODLR approximations need to be constructed for every iteration. Once the MLE estimator $\hat{\theta}$ for the parameters is obtained, the next step is to approximate the observed Fisher information matrix for uncertainty quantification. Denote the approximated Fisher information matrix by $\tilde{\mathcal{I}}$. Its entries are given by

$$\tilde{\mathcal{I}}_{i,j}(\hat{\theta}) = \frac{1}{2}\mathrm{tr}\left[\left(\tilde{K}^{-1}\tilde{K}_i\tilde{K}^{-1}\tilde{K}_j\right)|_{\theta=\hat{\theta}}\right]. \tag{3.4.3}$$

Note here all the matrices are evaluated at the MLE estimator $\hat{\theta}$. Based on the proposed operations in B.2, given the factorization of $\tilde{K}$ and HODLR matrices $\{\tilde{K}_j\}_{j=1,\cdots,p}$, evaluating each entry of the Fisher information matrix takes $O(n\log^2 n)$ time. Repeating for all entries, a total $O(p^2 n\log^2 n)$ complexity is required.

The number of parameters to estimate is often smaller comparing to the size of the observations. In summary, each optimization iteration takes $O(p\log n)$ K-vector products to construct the HODLR approximations and $O(pn\log^2 n)$ operations to carry out. After the optimization process, estimating the Fisher information matrix takes $O(p^2 n\log^2 n)$ time. If the K-vector product number of operations is quasi-linear in n as well (which, for implicitly defined covariance models with sparse state space operators e.g (3.5.6), is the case in many circumstances), then our approach has *quasillinear effort in building the covariance matrices and its derivatives, to factorize it, and per iteration of max-likelihood computation.*

3.5 Numerical Experiments

We perform several numerical experiments with synthetic datasets to demonstrate the scaling of our approach and the ability to recover the true values of the parameters from the data. We provide numerical evidence for both the computational scaling and accuracy of the parameter estimations. We note that our approach in §3.2 allows deriving covariance functions from physical principles followed by statistical analysis that employs the entire data set, the latter underpinning our objective of quasilinear performance. It is worth asking whether such a computational effort is worth the significant development cost and whether one can get away with fitting with only a portion of the data. To this end, we present in Appendix §B.1 an example of fitting a nonstationary process with a spatially linearily changing lengthscale. In that example we observe that fitting only on a subdomain results in good confidence intervals for the intercept of the lengthscale model but not the slope, whereas subsampling the data reverses that behavior, and, finally, using the entire data set produces good confidence intervals for both parameters, Figure B.1.1. Our approach in §3.2 allows for much more flexible modeling, including complex nonstationary models by, for example, using PDEs with spatially dependent parameters. It is thus likely to result in models where the dependency of those parameters on the lengthscales is complex and deciding *which* portion of the data to use for the purpose of obtaining good estimates may be a difficult endeavor. To this end, we conclude that deriving algorithms which allow the use of the entire data set may be worthwhile, and it is in this vein that we present the following numerical experiments.

3.5.1 SPDE Representation of Matérn Models

The SPDE approach to Gaussian fields can significantly reduce the computational cost of inference and sampling by invoking a GMRF approximation. One of the most popular examples is the stationary Matérn model. Recall a Gaussian field belongs to the Matérn family if its covariance function can be written in the form

$$M_{\nu,l}\left(\mathbf{x},\mathbf{y}\right) = (2^{\nu-1}\Gamma(\nu))^{-1}\left(\frac{||\mathbf{x}-\mathbf{y}||_2}{l}\right)^{\nu} K_{\nu}\left(\frac{||\mathbf{x}-\mathbf{y}||_2}{l}\right). \tag{3.5.1}$$

where Γ denotes the Gamma function and K_ν denotes the modified Bessel function of the second kind. The smoothness parameter ν controls the regularity of the random field, i.e. the degree of differentiability. An important characterization by Whittle [205, 206] is that the Matérn fields can be be defined as the solution to certain fractional order stochastic partial differential equation. Specifically, a Gaussian field with covariance of form $\sigma_m^2 M_{\nu,l}\left(\mathbf{x},\mathbf{y}\right)$ is the unique stationary solution to the SPDE

$$\left(\frac{1}{l^2} - \Delta\right)^{\frac{\nu+d/2}{2}} (\gamma w(\mathbf{x})) = \mathcal{W}(\mathbf{x}),\ \mathbf{x} \in \mathbb{R}^d, \tag{3.5.2}$$

where $\mathcal{W}$ denotes the spatial Gaussian white noise with unit variance and the marginal variance of w is given by

$$\sigma_m^2 = \frac{\Gamma(\nu)l^{2\nu}}{\Gamma(\nu+d/2)(4\pi)^{d/2}\gamma^2}. \tag{3.5.3}$$

Therefore we can control γ to get desired marginal variance for the Matérn field. It is worth pointing out that such models have sparse inverse covariance matrices, and, for example, the log-likelihood can be easily computed in quasilinear time directly without needing our approach [120] . The score equations, however, and in particular, the trace term of (3.2.4) do not have an obvious way to compute in quasilinear time if one wants to pursue maximum likelihood calculations. It is worth then investigating the potential benefit of using our approach to this end.

As discussed in [120], the SPDE formulation of Matérn field allows to simulate the random field by computing the solution of (3.5.2). And (3.5.2) can further be efficiently approximated by finite element analysis. In all the numerical simulations and studies described below, we fix the smoothness parameter $\nu = 1$ and restrict our attention to 2D space $d = 2$. In this case the order of the SPDE differential operator is an integer $\frac{\nu+d/2}{2} = 1$. Now the SPDE becomes

$$\left(\frac{1}{l^2} - \Delta\right)(\gamma w(\mathbf{x})) = \mathcal{W}(\mathbf{x}),\ \mathbf{x} \in \mathbb{R}^d. \tag{3.5.4}$$

Given a set of finite element basis functions $\{\Phi_i(\mathbf{x})\}_{i=1,\cdots,N_b}$, we solve (3.5.4) via standard finite element analysis. Assume that we observe the random process w at n observation points $(\mathbf{x}_1, \cdots, \mathbf{x}_n)$. We construct the observation matrix by evaluating the basis functions at given observation locations,

$$\mathbf{\Phi} = \begin{bmatrix} \Phi_1(\mathbf{x}_1) & \Phi_2(\mathbf{x}_1) & \cdots & \Phi_{N_b}(\mathbf{x}_1) \\ \vdots & \vdots & \ddots & \vdots \\ \Phi_1(\mathbf{x}_n) & \Phi_2(\mathbf{x}_n) & \cdots & \Phi_{N_b}(\mathbf{x}_n) \end{bmatrix}. \tag{3.5.5}$$

Note that Φ needs not be square and this flexibility can be used to define a latent grid on which the operators can be easily inverted (e.g a rectangle), but use the projection matrix to map onto the data space. Nevertheless we will assume that matrix vector products with Φ and Φ^T are easy to carry out, e.g the observations are well spread out and Φ is sparse.

Denote the finite element mass matrix $C_{ij} = \int_{\mathbb{R}^2} \Phi_i(\mathbf{x})\Phi_j(\mathbf{x})d\mathbf{x}$ and stiffness matrix $S_{ij} = \int_{\mathbb{R}^2} \nabla\Phi_i(\mathbf{x}) \cdot \nabla\Phi_j(\mathbf{x})d\mathbf{x}$. Further let $\tilde{C}$ denote the extracted diagonal matrix from C. The approximated finite-dimensional random field follows multivariate Gaussian distribution,

$$(w(\mathbf{x}_1), \cdots, w(\mathbf{x}_n))^T \sim \mathcal{N}\left(0, \mathbf{\Phi} K_w \mathbf{\Phi}^T\right),\ \text{where } K_w = \frac{1}{\gamma^2}\left(\frac{1}{l^2}C + S\right)^{-1} \tilde{C} \left(\frac{1}{l^2}C + S\right)^{-T}. \tag{3.5.6}$$

More details about the finite element analysis can be found in [30]. Note that the covariance structure of (3.5.6) consists of a sequence of products and matrix inverses of sparse finite element matrices. This model form enables fast covariance matrix-vector products through sequential sparse matrix-vector products.

3.5.2 A Matérn Based Wind Velocity Model

Here we use a Matérn based Gaussian process model for the horizontal wind components proposed in [82] as our numerical model. Specifically the horizontal wind velocity has two components $U = (u, v)^T$. The two components are connected via the Helmholtz decomposition, which states that for any given wind field U there exists a streamfunction ϕ and velocity potential χ, such that $U = \nabla \times \phi + \nabla \chi$. Assume the streamfunction and the velocity potential have the following bivariate Matérn structure:

$$K_{\phi,\chi}(\mathbf{x}, \mathbf{y}) = \begin{bmatrix} \sigma_\phi^2 & \rho\sigma_\phi\sigma_\chi \\ \rho\sigma_\phi\sigma_\chi & \sigma_\chi^2 \end{bmatrix} M_{\nu,l}(\mathbf{x}, \mathbf{y}), \tag{3.5.7}$$

where $\mathbf{x}, \mathbf{y}$ are 2D spatial locations of the random field and $M_{\nu,l}$ denotes the Whittle covariance function given by (3.5.1).

To approximate the bivariate Matérn covariance matrix (3.5.7) using the SPDE approach §3.5.1, we assume that both the streamfunction and the velocity potential are observed at the same set of locations. Then we have

$$K_{\phi,\chi} \approx \begin{bmatrix} \sigma_\phi^2 & \rho\sigma_\phi\sigma_\chi \\ \rho\sigma_\phi\sigma_\chi & \sigma_\chi^2 \end{bmatrix} \otimes K_w, \tag{3.5.8}$$

where $\otimes$ denotes the Kronecker product and K_w comes from (3.5.6).

The physics-based covariance model discussed back in Section 3.2.1 provides a flexible way to construct the covariance model of the wind velocity components using underlying physics relations. Here we restate the physics model involving the three random fields,

$$U = (u, v)^T = \nabla \times \phi + \nabla \chi = \left(-\frac{\partial}{\partial e_2}\phi + \frac{\partial}{\partial e_1}\chi, \frac{\partial}{\partial e_1}\phi + \frac{\partial}{\partial e_2}\chi\right)^T . \tag{3.5.9}$$

We can use finite difference method to discretize the operator or we can directly take the derivative of (ϕ, χ) since we have numerically approximated the solutions using finite element basis functions. To enable fast matrix-vector products, we take the finite difference discretization of the differential operators. Let $L_{1,2}$ denote the discretized one-dimensional differential operator with respect to the two directions e_1 and e_2. Then we have the discretized model

$$U = \begin{bmatrix} -L_2 & L_1 \\ L_1 & L_2 \end{bmatrix} \begin{bmatrix} \phi \\ \chi \end{bmatrix} = L \begin{bmatrix} \phi \\ \chi \end{bmatrix} . \tag{3.5.10}$$

Now we use the physics-based covariance model in (3.2.2), the covariance of the wind velocity components can be written as

$$K_U = \sigma_n I_{2n} + \hat{\Phi} L K_{\phi,\chi} L^T \hat{\Phi^T}. \tag{3.5.11}$$

Here $\hat{\Phi}$ is a latent-to-data projection operator, and we assume that the measurements are noisy with known variance σ_n (selected so the variance is a fraction of the sample variance). If u, v are available at the same points, then we can have $\hat{\Phi} = [\Phi^T, \Phi^T]^T$, where Φ is the interpolation operator from (3.5.5) (which is what we will use in our calculations). Since L is obtained by divided differences, it is sparse, moreover, we have access to efficient solvers for applying the inverse operators in K_w (3.5.6) needed to access matrix-vector products with $K_{\phi,\chi}$ (3.5.8).

In building K_U we have shown one way to construct a meaningful, moderately complex model which has the feature that matrix-vector products with K_U can be evaluated fast $O(n\log(n))$ in this case (assuming the latent space size is of the order of the data space size n), *despite the fact that K_U itself is dense. Therefore, storing it may be inaccessible for large data sets and thus even more so having access to its Cholesky factors needed for the log-determinant term in the likelihood.* A similar construction pattern can be applied beyond the classical Matérn model to circumstances where the SPDE representation can also be extended to non-stationary, non-isotropic fields [120], temporal processes [81] and spatial-temporal processes [163], and the resulting model will have the same property.

For our experiments, we use (3.5.11) and fix the smoothness parameter to $\nu = 1$ in the Matérn covariance function which corresponds to a process that is just barely not mean-square differentiable. As discussed in [82], realistic mesoscale wind fields have a smoothness parameter close to 1.25 to which our model is close. We use as true parameter values $(\rho, \sigma_\phi, \sigma_\chi, l) = (0.7, 1, 0.3, 0.5) = \theta_{\text{true}}$ and simulate five datasets on the 2D domain $[-5, 5]^2$ from the Matérn field using *R* package *RandomFields* of [165]. Each dataset contains 2^{20} observations for both ϕ and χ on an even grid across the domain; the wind field is then generated by the relation (3.5.9). We also note that to obtain the approximation of the Matérn covariance via (3.5.6), we perform finite element analysis with Neumann boundary condition on a slightly larger domain $[-5.5, 5.5]^2$ to avoid the boundary effect.

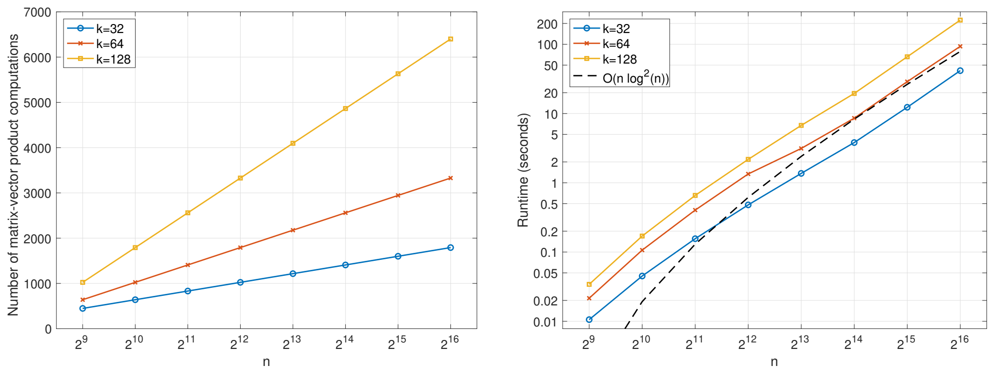
To approximate the covariance matrix by an HODLR format we use leaf level blocks with sizes 256 or 512. Therefore, the maximum number of levels of the HODLR approximation is $\lfloor \log_2(n/256) \rfloor$ where n is both the size of K_U and number of observations of the wind field U. For all off-diagonal blocks, a fixed rank is used which we will experiment with and specify later. The ordering of the observations is done by a KD-tree binary spatial partitioning. The reordering step increases the degree to which off-diagonal blocks correspond to the covariance between groups of well-separated points, a necessary characteristic for an efficient hierarchically low rank representation. All the following computations shown in this section were performed on a standard workstation with a 2.4 GHz CPU and 32 GB of memory.

Quasilinear Scaling of the Log-likelihood, Score Equations and the Fisher Information

Our datasets are generated over a $2^{10} \times 2^{10} = 2^{20}$ grid at the finest level. To demonstrate the numerical scaling, we coarsen the grid while also coarsening the finite element grid in (3.5.6) and the PDE operator L by the same factor. We then subsample observations according to the coarsened grids from the original dataset. For example, by coarsening both directions using a factor of 2, we can get a $2^9 \times 2^9 = 2^{18}$ grid. Then we subsample the generated observations of ϕ, χ, U according to the coarse grid. Now we obtain a dataset containing 2^{18} data points. Performing the same downsampling procedure again provides us a dataset with 2^{16} data points. Additionally we can randomly sample half of the grid points to observe on the 2^{18} grid. This gives us 2^{17} observations. Using the method described above, we generate datasets of size 2^r on irregular grids for any $r = 1, \cdots, 20$ by properly coarsening the grids and taking partial observations.

We generate subsampled datasets of sizes 2^r for r ranging from 9 to 16. The construction of the HODLR approximation of K_U is based on its products with sampling vectors as described in 3.3.2. Here we test three different ranks for HODLR off-diagonal blocks $k = 32, 64, 128$.

Following our proposed workflow, we first construct the HODLR approximation for K_U and all its derivatives with respect to parameters $\theta = (\rho, \sigma_\phi, \sigma_\chi, l)$ by matrix-vector products. The time complexity mainly comes from two parts: the evaluations of matrix-vector products with sampling vectors and the linear algebra computations including the QR factorization. Thus we count the number of covariance matrix-vector evaluation calls and record the time spent on the remaining linear algebra computations. The results are summarized in Fig. 3.5.1.



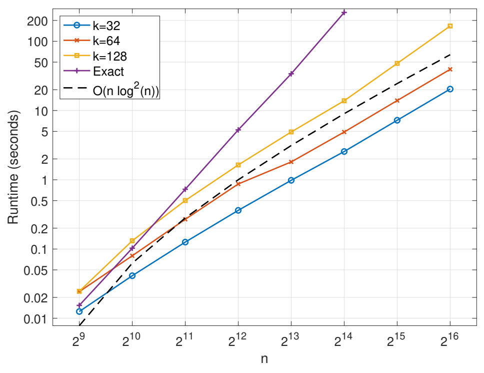
(a) Number of matrix-vector product evaluations (b) Runtime of rest of the operations

Figure 3.5.1: Computational complexity of constructing the HODLR approximation of K_U and its derivatives with respect to the parameters θ: (a) the total number of required K_U-vector products and (b) shows the runtime (in seconds) of the rest linear algebra operations for fixed off-diagonal rank 32 (blue curve with circles), 64 (red curve with crosses), 128 (yellow curve with squares) over different sizes of observations. We use number of observations of size $n = 2^r$ with r ranging from 9 to 16. In (b), to demonstrate the scaling, the theoretical line (black dashed line) corresponding to $O(n \log^2 n)$ is added to the plot.

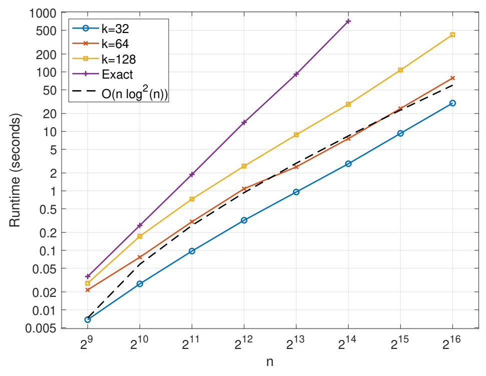
As can be seen in Fig. 3.5.1, the number of matrix-vector products is linear with $\log n$ and the slope depends on the off-diagonal rank we use. The scaling of the rest operations exactly follows the expected $O(n \log n)$ scale. This validates our complexity estimation.

Next, as the fundamental building blocks of approximating the score equations and Fisher information matrix, we want to show the two new operations $\text{tr}(A^{-1}B)$ and $\text{tr}(A^{-1}BC^{-1}D)$ proposed in Appendix B.2 are indeed quasilinear scale. In Fig. 3.5.2, we demonstrate the complexity of both operations (a) $\text{tr}(A^{-1}B)$ and (b) $\text{tr}(A^{-1}BC^{-1}D)$ given HODLR matrices A, B, C, D. As can be seen above, the computational scaling of both operations follows closely the expected $O(n \log^2 n)$ line. We also compare the results with their exact counterpart when the matrices are dense and have no hierarchical structures for $r \leq 14$, as can be seen our approach is $10 - 20$ time faster at $r = 14$ at which point the exact approach runs out of memory.

We now evaluate the complexity of computing the approximated log-likelihood, score equations and observed Fisher information matrix. Given the constructed HODLR approx-



(a) $\text{tr}(A^{-1}B)$



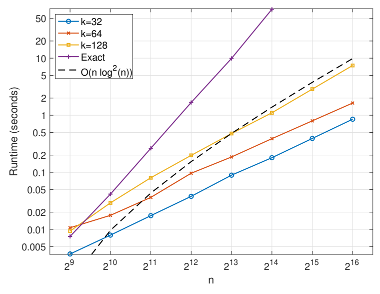
(b) $\text{tr}(A^{-1}BC^{-1}D)$

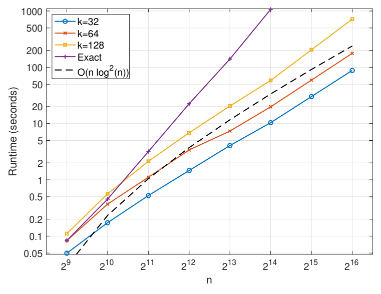
Figure 3.5.2: Runtime (in seconds) of (a) $\text{tr}(A^{-1}B)$ given two HODLR matrices A, B and $\text{tr}(A^{-1}BC^{-1}D)$ given the two products $A^{-1}B$ and $C^{-1}D$ for fixed off-diagonal rank 32 (blue curve with circles), 64 (red curve with crosses), 128 (yellow curve with squares) over $n = 2^r$ observations with r from 9 to 16. Theoretic lines corresponding to $O(n \log^2 n)$ scaling (purple curve with pluses) are added to each plot. Additionally, we include the complexity of their corresponding exact operations (black dashed line) when A, B, C, D are dense and have no structure in each plot for $r \leq 14$.

imation of the covariance matrix K_U and its derivatives with respect to all the parameters, we show the time taken to evaluate the log-likelihood, the score equations and the Fisher information matrix in Fig. 3.5.3. All the three evaluations follow the theoretical $O(n \log^2 n)$ scale, which is also indicative of the complexity of the time per iteration when we use the log-likelihood and score equations in a maximum likelihood algorithm. We conclude that our HODLR approximation algorithm for the log-likelihood, the score equations and the Fisher information matrix has quasilinear complexity and, moreover, the number of true covariance matrix-vector products required to build the approximation is $O(\log n)$, as we claimed in our analysis.

Numerical Accuracy of the Log-likelihood, Score Equations and the Fisher Information

Next we demonstrate the numerical accuracy of the approximated log-likelihood, score equations and Fisher information matrix comparing to their exact values. When carrying out


(a) Log-likelihood


(b) Score equations

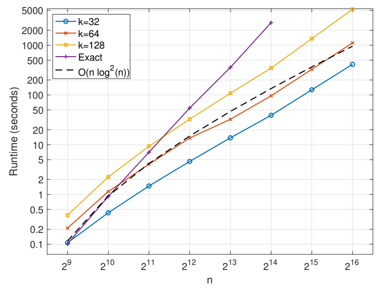

(c) Observed Fisher information matrix

Figure 3.5.3: Time taken (in seconds) to evaluate (a) the log-likelihood, (b) the score equations and (c) the observed Fisher information matrix exactly (purple curve with pluses) and using HODLR approximations for fixed off-diagonal rank 32 (blue curve with circles), 64 (red curve with crosses), 128 (yellow curve with squares) over $n = 2^r$ observations with r from 9 to 16. Theoretic lines corresponding to $O(n \log^2 n)$ scaling (black dashed line) are added to each plot.

the computations we will compare our model with another popular low-rank-type model, *the sparse spiked covariance model* [39], which can be thought of as diagonal plus low rank model. This will allow us to assess whether the workflow complexity and thus the development complexity of our model, which, as can be seen above, is considerable, is justified compared to the benefits it brings.

Sparse spiked covariance model for this model approximation class, we approximate the true covariance matrix by diagonal plus low-rank components [39]:

$$K_U \approx \sigma_n I_{2n} + VV^T. \tag{3.5.12}$$

Here σ_n is the same as in (3.5.11), and the approach is equivalent to considering the "physics-based" part of the model being low rank, not unlike our previous work [49]. We assume σ_n is known and estimate V using the randomized sketching techniques where we are only required to evaluate K_U-vector products. To ensure a fair comparison we consider a sparse spiked model with $O(\log n)$ spikes, which is equivalent to sketching $O(\log n)$ low-rank components. In this case it requires $O(\log n)$ K_U-vector products to estimate the low-rank components

V and the log-likelihood computation takes $O(n \log^2 n)$ time, which are both comparable to our HODLR model. We then compute the derivatives of the covariance matrix with respect to the parameters using the forward finite difference method since implementing the exact calculations would be a complex development effort. The score equations and the Fisher information matrix are thus computed using the numerical derivatives. Specifically we take the number of spikes to grow as $57 \log_2(n)$, which was chosen so that at the lowest value of the data size we choose the computation would be exact. That is for the lowest n considered we have $57 \log_2(n) \approx n$ (it is actually larger by 1 and we truncate it to n) so K_U would be in that case the exact covariance matrix. For the HODLR method, we used $k = 128$ for the off-diagonal rank in this experiment.

We subsample our datasets to generate observations of size 2^r for r ranging from 9 to 12. Recall that our dataset is simulated using the true parameter values $\theta_{\text{true}} = (0.7, 1, 0.3, 0.5)$. We explore the approximation accuracy of our approach both at the MLE point $(\hat{\rho}, \hat{\sigma_\phi}, \hat{\sigma_\chi}, \hat{l})$ and at a potential starting point of the optimization, which was chosen to be $\theta_{\text{init}} = (0.5, 0.5, 0.5, 0.5)$. For both points, we use the standard relative precision to measure the approximation accuracy. Note that when the log-likelihood is maximized exactly, the score equations should be exactly 0. In this case the relative precision is undefined. We also consider two scaling-free measures at the estimated MLE point from [71],

$$\eta_g := ||S(\theta) - \tilde{S}(\theta)||_{\mathcal{I}(\theta)^{-1}}, \tag{3.5.13}$$

for the score equations where $S(\theta), \tilde{S}(\theta)$ are the exact and approximated score equations respectively. We also measure

$$\eta_{\mathcal{I}} := \operatorname{tr}\left((\mathcal{I}(\theta) - \tilde{\mathcal{I}}(\theta))(\mathcal{I}(\theta)^{-1} - \tilde{\mathcal{I}}(\theta)^{-1})\right)^{1/2}, \tag{3.5.14}$$

for the Fisher information matrix, which is a natural metric for positive definite matrices.

We use $\epsilon_{\tilde{L}}, \epsilon_{\tilde{S}}, \epsilon_{\tilde{\mathcal{I}}}$ to denote the relative precision of log-likelihood, score equations and Fisher information matrix respectively. The averaged relative precision of the log-likelihood, score equations, and Fisher information matrix over all the five independently simulated datasets is summarized in part (a) of Tables 3.2 and 3.3 for our HODLR model and in part (b) of Tables 3.2 and 3.3 for the sparse spiked covariance model.

	$n = 2^9$	$n = 2^{10}$	$n = 2^{11}$	$n = 2^{12}$	$n = 2^{13}$
$\epsilon_{\tilde{L}}$	-6.72	-6.13	-5.82	-3.76	-3.79
$\epsilon_{\tilde{S}}$	-5.71	-4.73	-4.71	-2.07	-2.60
$\epsilon_{\mathcal{I}}$	-7.80	-6.07	-5.66	-1.96	-1.85

(a) HODLR covariance estimation

	$n = 2^9$	$n = 2^{10}$	$n = 2^{11}$	$n = 2^{12}$	$n = 2^{13}$
$\epsilon_{\tilde{L}}$	-15.58	-2.15	-1.07	-1.01	-0.71
$\epsilon_{\tilde{S}}$	-7.42	-1.22	-0.36	-0.24	-0.09
$\epsilon_{\mathcal{I}}$	-0.70	-0.81	-0.42	-0.36	-0.16

(b) Sparse spiked covariance estimation

Table 3.2: Averaged relative precision (on $\log_{10}$ scale) of the log-likelihood, score equations and observed Fisher information matrix. All the results are averaged for five datasets and are evaluated at the initial point of optimization $\theta_{\text{init}} = (0.5, 0.5, 0.5, 0.5)$.

	$n = 2^9$	$n = 2^{10}$	$n = 2^{11}$	$n = 2^{12}$
$\epsilon_{\tilde{L}}$	-6.68	-5.40	-6.15	-3.00
$\epsilon_{\mathcal{I}}$	-6.47	-5.16	-5.90	-1.83
η_g	-8.29	-5.15	-5.57	-0.58
$\eta_{\mathcal{I}}$	-5.57	-4.53	-4.86	-0.68

(a) HODLR covariance estimation

	$n = 2^9$	$n = 2^{10}$	$n = 2^{11}$	$n = 2^{12}$
$\epsilon_{\tilde{L}}$	-15.57	-2.16	-1.04	-1.00
$\epsilon_{\mathcal{I}}$	-0.77	-0.59	-0.41	-0.33
η_g	-11.02	0.56	1.99	2.28
$\eta_{\mathcal{I}}$	0.47	0.22	0.53	0.64

(b) Sparse spiked covariance estimation

Table 3.3: Averaged relative precision (on $\log_{10}$ scale) of the log-likelihood, score equations and observed Fisher information matrix. All the results are averaged for five datasets and are evaluated at the MLE point. Here $\epsilon_{\tilde{S}}$ is removed since the exact score equations tend to zero at the MLE point.

For the HODLR method, we see that the log-likelihood can be approximated very accurately, better than 0.1% relative accuracy both at the starting point and the MLE point. The score equations have relative errors less than 1% at the starting point and significantly better for the smaller cases. The observed Fisher information has relative error less than 1.5% (except on the scaling-free metrics on the largest case), and most times, significantly better than that, for a quantity that most times goes to infinity for increased problems size. The accuracy is worse compared to the HODLR approach in [71], though broadly comparable; we note, moreover, that that approach used explicit kernels.

Notice that when $n = 2^9$, $57 \log_2(n) = 513$ which we truncate to $512 = 2^9$. In that case the sparse spiked model is full-rank for the size of the problem. In that circumstance the log-likelihood can be estimated up to machine precision and the finite difference estimated score equations achieve square root of the machine precision. However the precision decays very quickly as the problem size grows, especially for the score equations and the Fisher information matrix. The error becomes even larger as the parameters getting closer to the exact MLE point estimates. Gradient-based optimization methods can fail with the unreliable gradient estimates. Other than $n = 2^9$, the error of this approach compared to our HODLR method is worse by at least one order of magnitude for all metrics and on average much worse than that. While some of this may conceivably be due to the numerical approximation of the derivatives, that concern is inapplicable for the likelihood approximation where errors are worse by two orders of magnitude. We conclude that HODLR is significantly superior in accuracy to the sparse spiked model, and that its additional development complexity and effort is justified.

Parameter Estimation

The previous subsections demonstrate both the scaling and the numerical accuracy of our proposed approximations. Both the two factors indicate their suitability for parameter estimations and uncertainty quantifications. Now we fit successively larger observations to obtain both the point estimates of the unknown parameters and their estimated confidence intervals using the observed Fisher information matrix. We take $n = 2^r$ observations with r from 9 to 14. In our experiments the HODLR off-diagonal rank is fixed at $k = 128$. To solve the maximum likelihood estimation we use MATLAB's `fminunc` function with the default `quasi_newton` algorithm to solve the unconstrainted optimization problem with respect to the parameters. We compare the exact model, the sparse spiked model and our proposed HODLR model. The initial guess is set to be $\theta_{\text{init}} = (0.5, 0.5, 0.5, 0.5)$. The stopping condition is chosen to be a relative tolerance of 10^{-6}. For small sets of observations $r \leq 12$, we also provide parameter values estimated using the exact score equations. Fig. 3.5.4 summarizes both the estimated parameters and their 95% confidence intervals for three independently simulated datasets.

As we can observe, the parameter estimates by our proposed approximations and the exact estimates are fairly close, both in terms of the parameter point estimates and the width of confidence intervals. Moreover, importantly, the true parameters are in the confidence intervals at our approach or very close to them except for large cases for the scale parameter, an issue also observed in [71]. We note that, for the Matérn models, despite their ubiquity in spatial statistics, it is known that some parameters may not always be estimable [218, 219, 176]. Therefore, such an outcome, where the parameters are outside the confidence intervals, is possible. What needs to be observed, however, is that whenever the exact confidence intervals could be computed they were essentially the same as the HODLR approach, except that HODLR produced them much faster. In contrast, the sparse spiked model provides decent parameter point estimates and confidence interval estimations for problems of small scales ($n \leq 2^{10}$ for example). But the results quickly diverge as we move to large-scale dataset. The confidence interval estimations become even more problematic as we have seen in Table 3.3. In practice we can encounter indefinite Fisher information approximations,

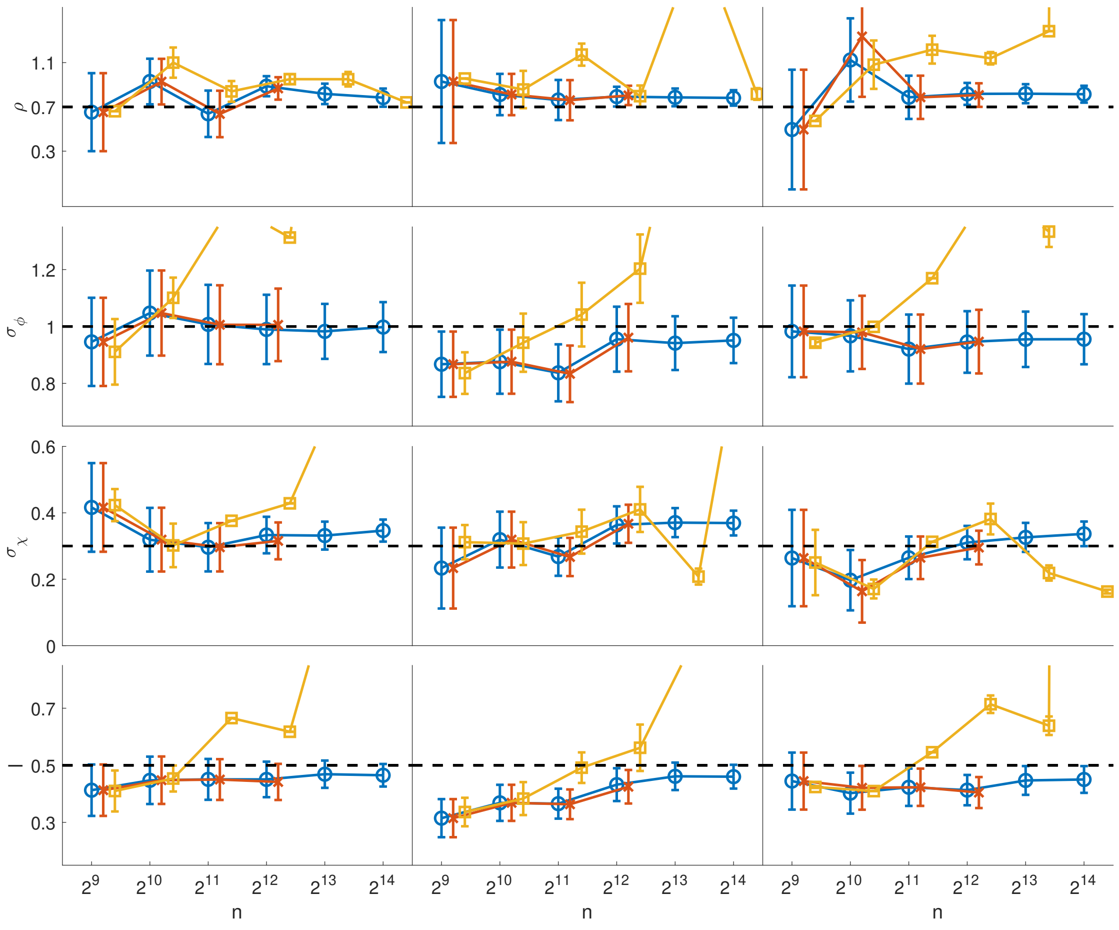


Figure 3.5.4: Estimated MLEs and their 95% confidence intervals using $n = 2^r$ observations with r from 8 to 13. Three columns in the figure represent results for three independently simulated datasets. The true parameter values $\theta_{\text{true}} = (\rho, \sigma_\phi, \sigma_\chi, l) = (0.7, 1, 0.3, 0.5)$ are added into each plot as black horizontal lines. We compare three models: (1) proposed HODLR model (blue curve with circles), (2) sparse spiked model (yellow curve with squares), (3) exact estimates (red line with x's) provided only for $r = 8, \cdots, 10$.

generating imaginary confidence interval width. In Fig. 3.5.4 we only plot the real part of the confidence intervals. We conclude that the HODLR approach systematically produces confidence intervals comparable to the exact method, whereas the sparse spiked models has a variable and, sometimes, much worse accuracy than either.

We note that finding a good initial point for parameter optimizations can be difficult in practical applications. On the other hand, smaller off-diagonal ranks $k = 36, 72$ can also provide satisfying estimates with significantly faster runtime per iteration (since the complexity is at least quadratic with local rank k). A useful strategy is to try out different starting points with smaller off-diagonal rank k and switch to larger rank when getting close to the MLE point.

3.5.3 Stationary Advection-diffusion-reaction Equation

In this example, we consider a 2D advection-diffusion-reaction equation given by:

$$-\mathrm{div}(\kappa \nabla u) + \mathbf{v} \cdot \nabla u + cu = \phi \quad \text{in } \Omega = [-5, 5]^2, \tag{3.5.15}$$

$$\frac{\partial u}{\partial n} = 0 \quad \text{on } \partial\Omega. \tag{3.5.16}$$

The coefficients $\kappa, \mathbf{v}, c$ represent the diffusion, the advective velocity and the reaction constant, respectively. Here ϕ is the latent source term we are interested in and u is the physical quantity that we can take measurements. Assume the source term ϕ is a Gaussian random field with Matérn covariance function:

$$\phi \sim \mathcal{N}\left(20 \cdot \exp\left(-\frac{||\mathbf{x}||^2}{2 \cdot 2^2}\right), \sigma_\phi^2 M_{\nu,l}\right). \tag{3.5.17}$$

The mean function is known. Similarly by the SPDE representation, we can use (3.5.6) to approximate the covariance. We use L to denote the discretized elliptic operator induced by (3.5.15) and (3.5.16). Then $Lu = \phi$ represents the elliptic PDE and u has covariance model $K_u = \sigma_n I_n + \Phi L^{-1} M_{\nu,l} L^{-T} \hat{\Phi}$, where σ_n is again observation noise. We form the sparse matrix L and use the backslash operator, effectively the standard LU factorization in Matlab. The covariance matrix-vector product is done sequentially, and involves two PDE solves and one matrix-vector product with the Matérn covariance SPDE approximation (3.5.6).

For the numerical experiments below, the magnitude parameter σ_ϕ and length scale parameter l are treated as the unknown parameters. We fix the smoothness parameter $\nu = 1$, and choose the physical parameters $\kappa = 0.001, c = 0.5, \mathbf{v} = (x_1+5, x_2+5)$. To solve the PDE (3.5.15) we discretize the domain $\Omega = [-5, 5]^2$ using a finite element mesh. The approximation of the Matérn covariance (3.5.6) is also obtained via finite element computations. However it is performed on a slightly larger domain $[-7.5, 7.5]^2$ to avoid the boundary effect.

The ground truth parameter values are taken to be $(\sigma_\phi, l) = (1, 1)$. We use the true parameters to simulate five Matérn random field using the *R* package *RandomFields* [165]. Each dataset contains 2^{20} samples of ϕ on a regular 2D grid over $[-7.5, 7.5]^2$. The observations of u are then generated by solving the PDE (3.5.15) and adding simulated measurement noise with σ_n chosen so that the sample standard deviation is 20% of the sample one.

To approximate the covariance matrix to HODLR we use leaf level blocks with size between 256 and 512. The maximum level of the HODLR approximation is thus $\lfloor \log_2 (n/256) \rfloor$ where n is both the size of K_u and number of observations of u (in other words, we choose $m = n$ here). For all off-diagonal blocks, a fixed rank $k = 128$ is used.

Numerical Accuracy of the Log-likelihood, Score Equations and the Fisher Information

We demonstrate the numerical accuracy of the approximations for different sizes of observations. Assume the starting point of parameter estimation is $(\sigma_\phi, l)_{\text{init}} = (1.5, 1.5)$.

Similarly we evaluate the accuracy using relative precision at the initial point and additionally measure (3.5.13) and (3.5.14) around the MLE point of the parameter estimations. All results in Table 3.4 and Table 3.5 are averaged from five independent datasets. We estimate the unknown parameters (σ_ϕ, l) by solving the approximated score equations (3.4.2). Fig. 3.5.5 illustrates both the estimated parameters and their 95% confidence intervals for three independently simulated datasets. Similarly the optimization is conducted using the MATLAB's `fminunc` function with the default `quasi_newton` algorithm with a relative tolerance of 10^{-6}.

	$n = 2^9$	$n = 2^{10}$	$n = 2^{11}$	$n = 2^{12}$	$n = 2^{13}$
$\epsilon_{\tilde{L}}$	-6.52	-5.62	-5.86	-5.26	-5.35
$\epsilon_{\tilde{S}}$	-4.25	-2.76	-2.68	-2.06	-2.21
$\epsilon_{\mathcal{I}}$	-6.10	-5.44	-4.96	-3.88	-2.47

Table 3.4: Averaged relative precision (on $\log_{10}$ scale) of the log-likelihood, score equations and observed Fisher information matrix. All the results are averaged for five datasets and are evaluated at the initial point of optimization $(\sigma_\phi, l)_{\text{init}} = (1.5, 1.5)$.

	$n = 2^9$	$n = 2^{10}$	$n = 2^{11}$	$n = 2^{12}$
$\epsilon_{\tilde{L}}$	-6.64	-5.87	-6.04	-5.54
$\epsilon_{\mathcal{I}}$	-6.41	-5.60	-5.38	-4.64
η_g	-7.21	-5.78	-5.34	-3.89
$\eta_{\mathcal{I}}$	-6.00	-5.33	-5.02	-4.31

Table 3.5: Averaged relative precision (on $\log_{10}$ scale) of the log-likelihood, score equations and observed Fisher information matrix. All the results are averaged for five datasets and are evaluated at the MLE point. Here $\epsilon_{\tilde{S}}$ is removed since the exact score equations tend to zero at the MLE point.

Parameter Estimation

The more relevant outcome, whether the true parameters are in the confidence intervals of the estimates obtained after the approximation, are however, better than those of §3.5.2 particularly for large examples. The parameter point estimations and the confidence intervals are summarized in Fig. 3.5.5. The parameter l is comparable in significance, though not in value, and, being smaller here it may be in itself the reason for the better fit. The approach certainly shows reasonable accuracy and consistency in our view, particularly given the fact that the covariance model is implicit. We point out that, in this section in particular, the model for u requires solving a partial differential equation to get, even if one would do an explicit treatment of the Matérn kernel itself (which would have been an option for the model in §3.5.2). Therefore in this case, for this model, our approach appears to be the only scalable alternative.

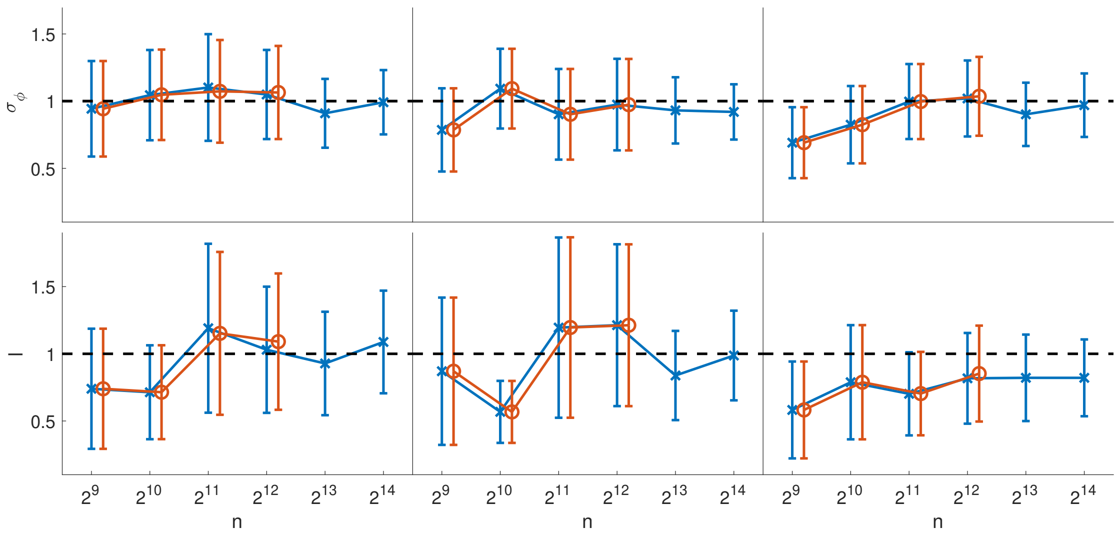


Figure 3.5.5: Estimated MLEs and their 95% confidence intervals using $n = 2^r$ observations with r from 8 to 13. Three columns in the figure represent results for three independently simulated datasets. The true parameter values $\theta_{\text{true}} = (\sigma_\phi, l) = (1, 1)$ are added into each plot as black horizontal lines. Exact estimates (circle) are provided for $r = 8, \cdots, 10$.

3.6 Discussion

In this paper, we propose a novel scheme for applying the well-known hierarchical matrices to Gaussian process maximum likelihood parameter estimation problem. For many spatial statistical problems, nearby observations have smoothly varying correlations with the rest of the system. By carefully reordering the observations and exploiting the structure, the off-diagonal blocks of the corresponding covariance matrix usually have fast decaying spectrum, which enables the application of hierarchical matrices to approximate the covariance matrix. This technique makes it possible to work with large datasets of measurements observed on unstructured grids. Estimations of the likelihood function, the score equations, and the expected Fisher information matrix all scale quasilinearly. Then the parameter estimations and uncertainty quantification can be obtained based on solving the score equation system and inverting the expected Fisher information matrix. Moreover, our approach uses ideas from [119] to construct the covariance matrix using only $O(\log(n))$ matrix-vector products. For statistical models defined implicitly, by means of stochastic differential equations that allow fast solvers for their deterministic counterparts, this allows for a very efficient way of building the HODLR approximation while allowing for a very flexible way of statistical modeling, such as, for example when dealing with nonstationarity for which explicit models are difficult to generate. We also note that in the process we proposed a way to compute exactly (without re-compression) traces of products of HODLR matrices which are important for both efficient evaluation of the score equations and computing the features of the Fisher Information matrix.

We note that the quality of the approximation depends crucially on the ordering of the observations and the local ranks of off-diagonal blocks. The proposed approach is a fixed rank strategy. An adaptive rank extension of the algorithm is certainly an important future direction to investigate. This can be done, potentially, by applying the adaptive randomized range sketching algorithm [79] with an online posterior estimate of the approximation error via samples. The choice of local ranks k itself provides a balance between complexity and accuracy. Generally the local ranks grow moderately as the problem size grows. It is also meaningful to increase the rank when the estimations get close to the optimality. Moreover, the approximation decreases in quality with increasing level order, it would be interesting to explore way of slightly increase the access to the covariance matrix and stabilize it – perhaps by combining it with stochastic estimators.

CHAPTER 4
COMMITTOR FUNCTIONS VIA TENSOR NETWORKS

4.1 Introduction

Understanding rare transitions between metastable states of a high-dimensional stochastic processes is a problem of great importance in the applied sciences. Examples of interesting transition events include chemical reactions, nucleation events during phase transitions, and conformational changes of molecules [109, 137, 217, 221, 24, 74]. In such complex systems, the dynamics linger near metastable states for long waiting periods, punctuated by sudden jumps from one metastable state to another. One important tool for describing transition events is transition path theory [201, 123, 190, 199], where the committor function plays a central role. The committor function measures the probability that the process hits a certain metastable state of the system before another and can be viewed as the solution of a backward Kolmogorov equation.

Computing the committor function in high-dimensional settings is a formidable task. Traditional numerical methods such as finite difference and finite element methods become prohibitively expensive in even moderate dimensions. To overcome the curse of dimensionality, significant efforts have been expended to apply deep learning framework to solve for high dimensional partial differential equations [202, 171, 80, 94, 147]. Most recently [95, 115, 114] have suggested representing the committor function using neural networks. Some of these approaches rely on sampling from the equilibrium distribution and so work well when the transitions are easily observed. For, e.g., chemical systems at low temperature the committor function can change sharply between the two metastable states, and transitions are rare and difficult to sample. To address this problem, [154] has proposed an adaptive importance sampling scheme. Meanwhile, the dynamical Galerkin framework for computing committor functions [184], which represents functions in a basis rather than as neural networks, approaches the sampling problem by initializing short trajectories uniformly according to known reaction coordinates.

Tensor network methods [61, 139, 193, 138, 64, 97] have emerged as an alternative to neural networks as a tool for high-dimensional problems in modern quantum physics and beyond. Typical tensor decomposition methods include tensor trains [141] (also known as linear tensor networks or matrix product states [1, 143, 204]), the CP decomposition [83], and the Tucker/Hierarchical Tucker decomposition [83, 188, 78]. These methods approximate tensors in compressed, structured formats that enable efficient linear algebra operations. More details can be found in [102, 76, 75, 96]. Moreover, tensor network methods have also been applied to solve for high-dimensional partial differential equations [98, 11].

In this paper we propose a novel approach to computing committor functions based on matrix product states/tensor trains. Specifically, we approximate both the equilibrium probability distribution and the committor function using tensor trains, achieving good performance even for high-dimensional problems in the low-temperature regime. This new approach fully bypasses the aforementioned difficulties due to sampling and establishes an alternative method for studying rare transition events between metastable states in complex, high-dimensional systems.

The rest of the paper is organized as follows. The committor function and its properties are reviewed in Section 4.2. Therein we also explain how the boundary condition can be accommodated within our tensor format and provide a summary of relevant tensor network methods. We introduce the key ingredients of our proposed method in Section 4.3. Numerical experiments for two representative classes of examples are presented in Section 4.4, demonstrating the accuracy and efficiency of the proposed algorithm. Finally, in Section 4.5 we summarize our findings.

4.2 Background and preliminaries

In this section we first review the motivation for computing committor functions, and summarize challenges and recent advances relevant to this task. We then briefly discuss tensor train decompositions, introduce the basic tensor operations, and define relevant associated notations used in this work. Throughout the paper, we use MATLAB notation for multidimensional array indexing.

4.2.1 Committor functions

The underyling stochastic process of interest is the overdamped Langevin process, defined by

$$d\boldsymbol{X}_t = -\nabla V(\boldsymbol{X}_t)\,dt + \sqrt{2\beta^{-1}}\,d\boldsymbol{W}_t, \tag{4.2.1}$$

where $\boldsymbol{X}_t \in \Omega \subset \mathbb{R}^d$ is the state of the system, $V : \Omega \subset \mathbb{R}^d \to \mathbb{R}$ is a smooth potential energy function, $\beta = 1/T$ is the inverse of the temperature T, and $\boldsymbol{W}_t$ is a d-dimensional Wiener processs. If the potential energy function V is confining for Ω (see, e.g., [25, Definition 4.2]), then one can show that the equilibrium probability distribution of the Langevin dynamics (4.2.1) is the Boltzmann-Gibbs distribution

$$p(\boldsymbol{x}) = \frac{1}{Z_\beta}\exp(-\beta V(\boldsymbol{x})). \tag{4.2.2}$$

where $Z_\beta = \int_\Omega \exp(-\beta V(\boldsymbol{x}))\,d\boldsymbol{x}$ is the partition function. We are interested in the transition between two simply connected domains $A, B \subset \Omega$ with smooth boundaries. The associated committor function $q : \Omega \to [0,1]$ is defined by

$$q(\boldsymbol{x}) = \mathbb{P}(\tau_B < \tau_A \mid \boldsymbol{X}_0 = \boldsymbol{x}), \tag{4.2.3}$$

where τ_A and τ_B are the hitting times for the sets A and B, respectively. The committor function q provides a useful statistical description of properties such as the density and probability of reaction trajectories [190, 199, 150]. However, computing the committor function can be a formidable task since it involves solving the following (possibly high-dimensional) backward Kolmogorov equation with Dirichlet boundary conditions:

$$-\beta^{-1}\Delta q(\boldsymbol{x}) + \nabla V(\boldsymbol{x}) \cdot \nabla q(\boldsymbol{x}) = 0 \text{ in } \Omega \backslash (A \cup B),\ q(\boldsymbol{x})|_{\partial A} = 0,\ q(\boldsymbol{x})|_{\partial B} = 1. \tag{4.2.4}$$

For high-dimensional problems, traditional methods such as finite difference and finite element discretization are intractable. Numerous alternative methods can effectively approximate the committor function under the assumption that the transition paths from A to B are localized in a quasi-one-dimensional reaction tube or low-dimensional manifold. For example, the finite temperature string method [200, 191] approximates the isosurfaces of the committor function with hyperplanes normal to the most probable transition paths. The diffusion map approach [56] aims to obtain the committor function on a set of points by applying point cloud discretization to the generator $L = -\beta^{-1}\Delta + \nabla V(\boldsymbol{x}) \cdot \nabla$. The method presented in [107] improves the diffusion map approach by discretizing L using a finite element method on local tangent planes of the point cloud. Also an alternative approach was introduced by solving a potential function instead, see [124] for details.

To compute the committor function, a classic approach is to solve the variational problem

$$\arg\min_{q} \int_{\Omega} |\nabla q(\boldsymbol{x})|^2 p(\boldsymbol{x})\, d\boldsymbol{x}, \quad q(\boldsymbol{x})|_{\partial A} = 0,\ q(\boldsymbol{x})|_{\partial B} = 1, \tag{4.2.5}$$

for which (4.2.4) is the Euler-Lagrange equation, see for example [127, 95, 115, 114, 154]. Specifically [95, 115, 114] proposed to parametrize the committor function q using neural networks. In order to obtain an unconstrained optimization problem, the boundary conditions are enforced in [95, 114, 154] by adding two extra penalty terms, as in

$$\arg\min_q \int_\Omega |\nabla q(\boldsymbol{x})|^2 p(\boldsymbol{x})\, d\boldsymbol{x} + \rho \int_{\partial A} q(\boldsymbol{x})^2 p_{\partial A}(\boldsymbol{x})\, d\boldsymbol{x} + \rho \int_{\partial B} (q(\boldsymbol{x}) - 1)^2 p_{\partial B}(\boldsymbol{x})\, d\boldsymbol{x}, \quad (4.2.6)$$

where $p_{\partial A}$ and $p_{\partial B}$ define probability measures supported on the boundaries ∂A and ∂B respectively. In all of these works, the objective is evaluated and optimized via stochastic sampling. By contrast, in this work, we propose to represent the committor function q in a tensor train format, which will allow for optimization via stable and efficient deterministic linear algebra operations.

Since the potential function V is confining, we can effectively restrict our domain of interest to a bounded subset of Ω. Outside of this subset, the density is small and contributes only negligibly contributes to the variational cost (4.2.5). For simplicity we shall identify Ω with this subset and assume $\Omega = \Omega_1 \times \Omega_2 \times \dots \Omega_d$ where each $\Omega_i \in \mathbb{R}$ is a bounded subset.

4.2.2 Soft boundary condition

Unfortunately, the formulation (4.2.6) is not immediately amenable to optimization within a tensor format for q. The reason is that the surface measures on ∂A and ∂B cannot themselves be identified with functions on Ω, much less as functions that can be compressed in tensor format, so the penalty terms cannot simply be viewed as inner products of tensor trains.

Therefore we instead consider an objective of the form

$$\arg\min_q \int_\Omega |\nabla q(\boldsymbol{x})|^2 p(\boldsymbol{x})\, d\boldsymbol{x} + \rho \int_\Omega q(\boldsymbol{x})^2 p_A(\boldsymbol{x})\, d\boldsymbol{x} + \rho \int_\Omega (q(\boldsymbol{x}) - 1)^2 p_B(\boldsymbol{x})\, d\boldsymbol{x}. \quad (4.2.7)$$

Here p_A and p_B are probability densities that are supported on boundaries A and B, respectively, and absolutely continuous with respect to the Lebesgue measure on Ω.

In fact, we show in C.1 that the exact optimizer of (4.2.7) admits a probabilistic interpretation similar to that of the usual committor function. As such we call the optimizer a 'soft committor function.' Specifically, the interpretation is based on a modification of the Langevin dynamics (4.2.1) in which one augments the state space Ω with two 'cemetery states' c_A and c_B. The process jumps randomly to these two states with instantaneous jump rates $\frac{\rho \cdot p_A}{\beta \cdot p}$ and $\frac{\rho \cdot p_B}{\beta \cdot p}$, respectively. The soft committor function evaluates the probability that the modified process hits c_B before c_A. This formulation is rather similar to the Poisson type equation introduced in [124] with some differences in the right hand side of the equation.

From a different perspective, when ρ is large and p_A and p_B concentrate near ∂A and ∂B, respectively, the soft committor function can be viewed as an approximation of the ordinary committor function. In fact, if p_A and p_B are Gaussian densities, then in high dimensions [29], p_A and p_B each weakly approximates a uniform measure on a suitable hypersphere. This is convenient because A and B are often chosen to be balls and the Gaussian densities have exact tensor train representations. Section 4.3.3 provides two examples on the construction of p_A and p_B using Gaussian densities. In practice we choose p_A and p_B such that they are well-representable in tensor train format (introduced in subsection Section 4.2.3.) More details on the construction of p_A and p_B will be provided below in Section 4.3.

For simplicity, in what follows we will simply refer to soft committor functions as committor functions.

4.2.3 Tensors and tensor networks

In this subsection we summarize the basic tensor operations used in this work. In particular, for ease of exposition we introduce tensor network diagram notation, which provides a convenient way of visually describing tensor operations. We also introduce the matrix product state/tensor train format for parametrizing high-dimensional functions.

In tensor diagrams, a tensor is represented by a node, where the number of incoming legs indicates the dimensionality of the tensor, i.e., the number of indices/arguments. There are two types of leg: legs indicating continuous arguments are denoted by dashed lines, and legs indicating discrete indices are denoted by solid lines. For example, Fig. 4.2.1 (a) shows the tensor diagram for a 3-tensor $\mathcal{A}$ and a 2-tensor $\mathcal{B}$, which can be viewed as two functions

$$\mathcal{A}(x_1, i_2, i_3), \quad \mathcal{B}(j_1, x_2), \tag{4.2.8}$$

respectively, where x_1, x_2 are continuous variables and i_2, i_3, j_1 are discrete variables.

We also define the multi-dimensional Kronecker delta tensor (depicted by an inverted triangular node as in Fig. 4.2.1 (b)):

$$\delta(x_1, x_2, \ldots, x_d) = \begin{cases} 1 & \text{if } x_1 = x_2 = \cdots = x_d. \\ 0 & \text{otherwise.} \end{cases} \tag{4.2.9}$$

By a slight abuse of notation, we will use the same symbol to represent the appropriate Dirac delta function when the legs represent continuous variables.

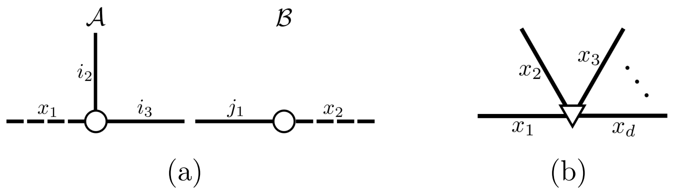


Figure 4.2.1: (a) Tensor diagrams for a 3-tensor $\mathcal{A}$ and a 2-tensor $\mathcal{B}$. (b) Tensor diagram for a d-dimensional Kronecker Delta node. Solid lines correspond to discrete variables, and dashed lines correspond to continuous variables.

Next we describe a key operation called tensor contraction. This operation is indicated visually by joining legs from different tensors. For example, in Fig. 4.2.2 (a), the third leg of $\mathcal{A}$ is joined with the first leg of $\mathcal{B}$. This corresponds to the computation

$$\mathcal{C}(x_1, i_2, x_2) = \sum_k \mathcal{A}(x_1, i_2, k)\mathcal{B}(k, x_2), \tag{4.2.10}$$

where it is implicitly assumed that the indices of the joined legs have the same range. Here x_1, x_2 are continuous variables, and i_2 is a discrete variable.

Tensor contraction can be defined for continuous legs as well. For example, in Fig. 4.2.2 (b), continuous legs of $\mathcal{A}$ and $\mathcal{B}$ are contracted, corresponding to the operation

$$\mathcal{D}(j_1, i_2, i_3) = \int_{\Omega_0} \mathcal{B}(j_1, x)\mathcal{A}(x, i_2, i_3)\, dx, \tag{4.2.11}$$

for some suitable domain Ω_0, which is implicitly assumed to be the domain of both joined legs. The resulting tensor $\mathcal{D}$ is a 3-tensor with only discrete legs.

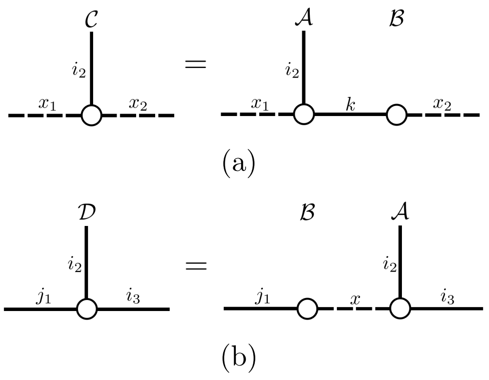


Figure 4.2.2: (a) Tensor contraction of discrete legs. (b) Tensor contraction of continuous legs.

A tensor network diagram consists of a collection of individual tensor diagrams with some pairs of legs joined, i.e., contracted. The contracted legs correspond to the so-called 'internal indices' for the tensor network, while the uncontracted legs correspond to 'external indices,' which are the indices remaining after all of the indicated contractions have been performed.

Next we introduce several low-complexity tensor networks and their corresponding diagrams. A matrix product state (MPS) or tensor train (TT) is a factorization of a d-tensor into a chain-like product of 3-tensors. Such a factorization allows one to approximate high-dimensional tensors and manipulate them efficiently, typically with $O(d)$ time and memory complexity.

Definition 1. *Let $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_d}$ be a d-tensor, with entries indexed by $(i_1, i_2, \ldots, i_d)$. Then we say that $\mathcal{A}$ is a MPS/TT with ranks $\boldsymbol{r} = (r_0, \ldots, r_d)$, where we fix $r_0 = r_d = 1$ by convention, if one can write*

$$\begin{aligned}
\mathcal{A}(i_1, i_2, \ldots, i_d) &= \sum_{\alpha_0=1}^{r_0} \cdots \sum_{\alpha_d=1}^{r_d} \mathcal{G}_1(\alpha_0, i_1, \alpha_1)\mathcal{G}_2(\alpha_1, i_2, \alpha_2) \ldots \mathcal{G}_d(\alpha_{d-1}, i_d, \alpha_d) \\
&= \mathcal{G}_1(:, i_1, :)\mathcal{G}_2(:, i_2, :) \cdots \mathcal{G}_d(:, i_d, :) \qquad (4.2.12)
\end{aligned}$$

for all $(i_1, i_2, \ldots, i_d)$. Here $\mathcal{G}_k(:, i_k, :) \in \mathbb{R}^{r_{k-1} \times r_k}$ is viewed as a matrix for each $k = 1, \ldots, d$, and the matrix product in (4.2.12) *is a 1×1 matrix, i.e., a scalar value. The 3-tensor $\mathcal{G}_k \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}$ is called the k-th tensor core of $\mathcal{A}$.*

In tensor diagrams, an MPS/TT is represented by a chain of 3-tensors, as in Fig. 4.2.3 (a). Note that the 0-th and last tensor cores can be viewed as 2-tensors since $r_0 = r_d = 1$, and as such the corresponding legs can be omitted from the diagram.

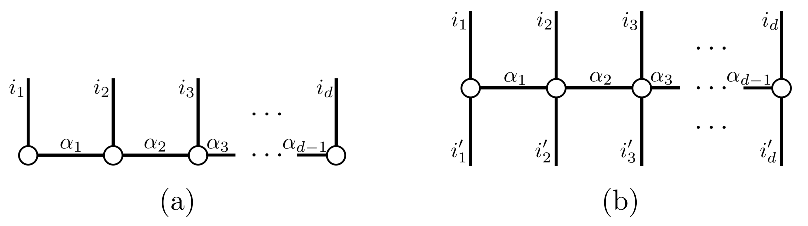


Figure 4.2.3: (a) d-dimensional TT/MPS. (b) d-dimensional MPO. The indices depicted match the expressions in (4.2.12) and (4.2.14). Note that we omit the legs for the trivial indices α_0 and α_d.

A matrix product operator (MPO) is a type of tensor network analogous to matrix for which each constituent tensor has two external, uncontracted legs as well as two internal indices contracted with neighboring tensors in a chain-like fashion. Concretely, an MPO is a tensor $O \in \mathbb{R}^{(m_1 \times m_2 \times \ldots \times m_d) \times (n_1 \times n_2 \times \ldots \times n_d)}$ that can be written in the form

$$
\begin{aligned}
O(i_1, \ldots, i_d; i'_1, \ldots, i'_d) &= \sum_{\alpha_0, \ldots, \alpha_d} \mathcal{G}_1(\alpha_0, i_1, i'_1, \alpha_1) \ldots \mathcal{G}_d(\alpha_{d-1}, i_d, i'_d, \alpha_d) && (4.2.13) \\
&= \mathcal{G}_1(:, i_1, i'_1, :) \ldots \mathcal{G}_d(:, i_d, i'_d, :), && (4.2.14)
\end{aligned}
$$

where the sum over α_k has a range defined by a corresponding rank r_k, as in Definition 1, and $r_0 = r_d = 1$ by convention. In this case we similarly say that the MPO has ranks $\boldsymbol{r} = (r_0, \ldots, r_d)$. The 4-tensor $\mathcal{G}_k \in \mathbb{R}^{r_{k-1} \times m_k \times n_k \times r_k}$ is called the k-th tensor core. A corresponding tensor network diagram is shown in Fig. 4.2.3 (b).

For further background on tensor networks and diagrams, see [139].

4.3 Proposed method

In this work we obtain the committor function by solving the variational problem (4.2.7) within a MPS/TT parametrization for the committor function q. We demonstrate that by approximating the equilibrium probability density p in MPS/TT format, this optimization problem can be solved using basic tensor operations. In particular the minimization is accomplished using a standard alternating least squares approach.

4.3.1 Discretizing the variational problem

We will represent the unknown committor function in a tensor product basis according to the product structure of the domain $\Omega = \Omega_1 \times \Omega_2 \times \ldots \times \Omega_d$. Within this basis, we will approach the variational problem (4.2.7) by Galerkin approximation.

To begin, suppose that we have an orthogonal basis for each $L^2(\Omega_k)$, denoted by $\{\phi_j^{(k)}\}_{j=1}^{\infty}$. In order to obtain a finite-dimensional problem we consider the subspace of $L^2(\Omega_k)$ spanned by only the first $L^{(k)}$ basis functions. Here the $L^{(k)}$, $k = 1, \ldots, d$, are a set of positive integers which are either fixed or determined adaptively. Then given the finite basis $\{\phi_j^{(k)}\}_{j=1}^{L^{(k)}}$, which spans a subspace of $L^2(\Omega_k)$ for each $k = 1, \ldots, d$, we can consider an expansion of q in the corresponding tensor product basis:

$$\begin{aligned} q(\boldsymbol{x}) &= \sum_{i_1,\ldots,i_d} \mathcal{Q}(i_1,\ldots,i_d)\phi_{i_1}^{(1)}(x_1)\ldots\phi_{i_d}^{(d)}(x_d), \\ &:= \sum_{\boldsymbol{i}} \mathcal{Q}(\boldsymbol{i})\phi_{i_1}^{(1)}(x_1)\ldots\phi_{i_d}^{(d)}(x_d), \end{aligned} \tag{4.3.1}$$

where, for notational convenience, we have defined $\boldsymbol{i} := (i_1, i_2, \ldots, i_d)$ and $\boldsymbol{x} := (x_1, x_2, \ldots, x_d)$. Additionally, we set $\phi^{(k)} := (\phi_j^{(k)})_{j=1}^{L^{(k)}}$. Each $\phi^{(k)}$ can be viewed as a 2-tensor via $\phi^{(k)}(j, x) = \phi_j^{(k)}(x)$, where the first index is the basis function index and the second (continuous) index is a spatial coordinate. Then the decomposition (4.3.1) for q can be depicted graphically as in Fig. 4.3.1.

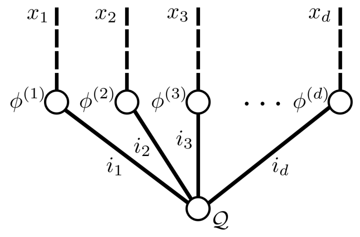


Figure 4.3.1: Tensor diagram for the decomposition (4.3.1) of the committor function.

To determine the committor function q within the truncated tensor product basis, we want to determine the coefficient tensor $\mathcal{Q}$ that is optimal in the sense of (4.2.7). By inserting the parameterization (4.3.1) into the variational problem (4.2.7), we can rewrite the optimization problem as follows,

$$
\begin{aligned}
\underset{\mathcal{Q}}{\arg\min} \quad & \underbrace{\sum_{k=1}^{d}\sum_{\boldsymbol{i},\boldsymbol{j}} H^k(\boldsymbol{i};\boldsymbol{j})\mathcal{Q}(\boldsymbol{i})\mathcal{Q}(\boldsymbol{j})}_{\int_\Omega |\nabla q(\boldsymbol{x})|^2 p(\boldsymbol{x})\,d\boldsymbol{x}} + \underbrace{\rho\sum_{\boldsymbol{i},\boldsymbol{j}} H^A(\boldsymbol{i};\boldsymbol{j})\mathcal{Q}(\boldsymbol{i})\mathcal{Q}(\boldsymbol{j})}_{\rho\int_\Omega q(\boldsymbol{x})^2 p_A(\boldsymbol{x})\,d\boldsymbol{x}} \\
& + \underbrace{\rho\sum_{\boldsymbol{i},\boldsymbol{j}} H^B(\boldsymbol{i};\boldsymbol{j})\mathcal{Q}(\boldsymbol{i})\mathcal{Q}(\boldsymbol{j}) - 2\rho\sum_{\boldsymbol{i}}\mathcal{Q}(\boldsymbol{i})h^B(\boldsymbol{i}) + \rho}_{\rho\int_\Omega (q(\boldsymbol{x})-1)^2 p_B(\boldsymbol{x})\,d\boldsymbol{x}}, \qquad (4.3.2)
\end{aligned}
$$

where

$$H^k(\boldsymbol{i};\boldsymbol{j}) = \int_\Omega \frac{\partial}{\partial x_k}\left[\phi^{(1)}_{i_1}(x_1)\dots\phi^{(d)}_{i_d}(x_d)\right]\frac{\partial}{\partial x_k}\left[\phi^{(1)}_{j_1}(x_1)\dots\phi^{(d)}_{j_d}(x_d)\right]p(\boldsymbol{x})\,d\boldsymbol{x} \qquad (4.3.3)$$

$$H^A(\boldsymbol{i};\boldsymbol{j}) = \int_\Omega \phi^{(1)}_{i_1}(x_1)\dots\phi^{(d)}_{i_d}(x_d)\phi^{(1)}_{j_1}(x_1)\dots\phi^{(d)}_{j_d}(x_d)p_A(\boldsymbol{x})\,d\boldsymbol{x} \qquad (4.3.4)$$

$$H^B(\boldsymbol{i};\boldsymbol{j}) = \int_\Omega \phi^{(1)}_{i_1}(x_1)\dots\phi^{(d)}_{i_d}(x_d)\phi^{(1)}_{j_1}(x_1)\dots\phi^{(d)}_{j_d}(x_d)p_B(\boldsymbol{x})\,d\boldsymbol{x} \qquad (4.3.5)$$

$$h^B(\boldsymbol{i}) = \int_\Omega \phi^{(1)}_{i_1}(x_1)\phi^{(2)}_{i_2}(x_2)\dots\phi^{(d)}_{i_d}(x_d)p_B(\boldsymbol{x})\,d\boldsymbol{x}. \qquad (4.3.6)$$

Here $(\boldsymbol{i};\boldsymbol{j}) = (i_1,\dots,i_d;j_1,\dots,j_d)$ is a concatenation of multi-indices. We can simply ignore the last constant ρ since it does not affect the minimizer. Computing the tensors $\{H^k\}_{k=1}^d$, H^A, H^B, and h^B is *prima facie* intractable as it requires us to perform integration over the d-dimensional domain Ω, in addition to storing tensors of exponential size in d. Moreover, the number of unknown tensor entries of $\mathcal{Q}$ is also exponential in d. Traditional approaches are therefore prohibitively expensive for d of even moderate size.

In the next two sections we show how to use MPS/TT approximations to obtain $\{H^k\}_{k=1}^d$, H^A,H^B,h^B, allowing us to solve the optimization problem (4.3.2) with computational and storage complexities of $O(d)$.

4.3.2 Constructing H^k

In this subsection we detail the construction of H^k, which corresponds to the variational energy term $\int_\Omega |\nabla q(\boldsymbol{x})|^2 p(\boldsymbol{x}) d\boldsymbol{x}$ of (4.2.7). As mentioned above, in order to obtain each H^k in (4.3.3), one needs to evaluate a d-dimensional integral and store the resulting high-dimensional tensor. To circumvent the exponential complexity in d, we assume that the equilibrium density p can be approximated as an MPS/TT as follows:

$$p(\boldsymbol{x}) = \sum_{\substack{m_1,\ldots,m_d \\ \alpha_0,\ldots,\alpha_d}} \mathcal{P}_1(\alpha_0, m_1, \alpha_1) \ldots \mathcal{P}_d(\alpha_{d-1}, m_d, \alpha_d) \psi^{(1)}_{m_1}(x_1) \cdots \psi^{(d)}_{m_d}(x_d), \tag{4.3.7}$$

where $\psi^{(k)} := (\psi^{(k)}_j)_{j=1}^{K^{(k)}}$ is a vector of univariate basis functions $\Omega_i \to \mathbb{R}$. Fig. 4.3.2 illustrates the structure of the equilibrium density p that we assume in this paper.

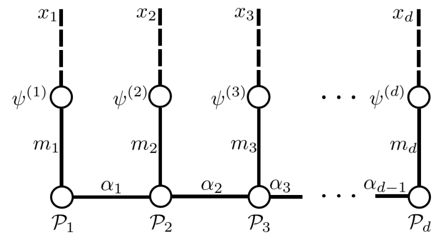


Figure 4.3.2: Tensor diagram for the approximate equilibrium density p (4.3.7).

The construction of the MPS/TT format for a given equilibrium density p will be described in Section 4.4 in the contexts of specific example problems.

Such an approximation of p amounts to changing the tensor representation of H^k depicted graphically in Fig. 4.3.3 (a) to the representation in Fig. 4.3.3 (b). Note that these calculations involve the derivatives of our univariate basis functions $\phi^{(k)}$. In our figures, we use a hollow node to represent the vector of basis functions $\phi^{(k)}$ and a filled node to represent its vector of derivatives, i.e., $d\phi^{(k)}/dx$. We observe that the $\{H^k\}_{k=1}^d$ are naturally viewed as MPOs, and moreover the construction of these MPOs can be performed using basic tensor algebra in $O(d)$ complexity. More precisely:

(1) To construct an MPO for H^k following Fig. 4.3.3 (b), note that we need to perform two types of tensor contraction: one involving the original basis functions $\phi^{(l)}$, $l \neq k$, and the other involving the derivatives $d\phi^{(k)}/dx$. Therefore we precompute these two contractions, which can be recycled to form MPOs for the H^k, $k = 1, \ldots, d$. First, we form I_l, $l = 1, \ldots, d$ by contracting three tensors $\psi^{(l)}$, $\phi^{(l)}$, $\phi^{(l)}$ and form $\tilde{I}_l$, $l = 1, \ldots, d$ by contracting three tensors $\psi^{(l)}$, $d\phi^{(l)}/dx$, $d\phi^{(l)}/dx$. The tensors I_l and $\tilde{I}_l$ are defined graphically in Fig. 4.3.4 (a) and (b). These contractions can be performed by univariate numerical integration. Next we contract each I_l with the corresponding tensor core $\mathcal{P}_l$ to obtain H_l for $l = 1, \ldots, d$. Similarly we contract $\tilde{I}_l$ with $\mathcal{P}_l$ to obtain $\tilde{H}_l$. These constructions are illustrated in Fig. 4.3.4 (a) and (b), respectively.

(2) Next we assemble H^k by substituting $H_l, \tilde{H}_l$, $l = 1, \ldots, d$ into the red boxes in Fig. 4.3.4 (c) as needed. This yields an MPO as shown in Fig. 4.3.4 (c) on the right. We denote l-th tensor core of H^k by H^k_l.

(3) Finally we repeat step (2) for all $k = 1, \ldots, d$.

The algorithm outputs core tensors $H^k_1, \ldots, H^k_d$ for the MPO H^k, $k = 1, \ldots, d$. The computational complexity of forming each pair $I_l, \tilde{I}_l$ is $O(K^{(l)}L^{(l)2})$ since a univariate numerical integration is performed for each entry. Hence the cost of computing all of the $I_l, \tilde{I}_l$ is $O(dKL^2)$, where we define $K := \max_l K^{(l)}$, $L := \max_l L^{(l)}$.

If we assume that the MPS/TT format for p has ranks $\boldsymbol{r} = (r_0, r_1, r_2, \ldots, r_d)$, and set $r = \max_l r_l$, then the contraction steps with the $\mathcal{P}_l$ altogether cost $O(dr^2KL^2)$. Therefore to construct the tensor cores $H^k_1, \ldots, H^k_d$, the total computational cost is $O(dr^2KL^2)$. The memory complexity, including that of storing the intermediate tensors I_l, $\tilde{I}_l$, H_l, $\tilde{H}_l$, is

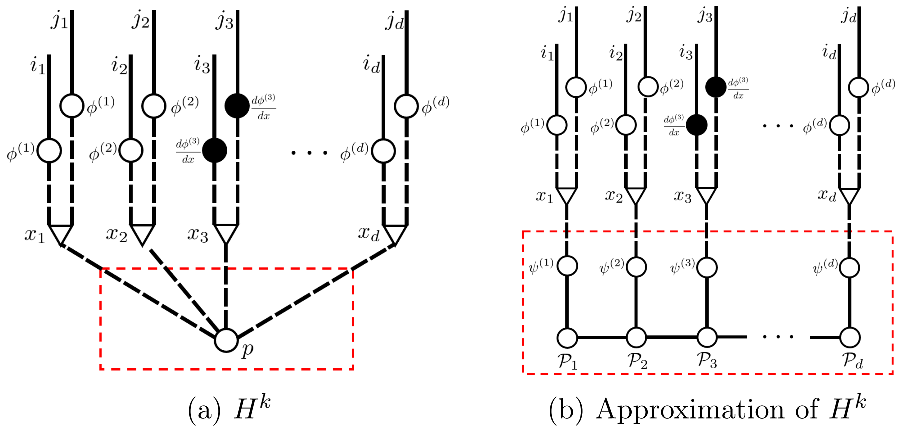


Figure 4.3.3: (a) Tensor diagram for H^k, $k = 3$, as defined in (4.3.3). (b) Approximation of H^k obtained by replacing p with its MPS/TT approximation (4.3.7). We use red dashed boxes to indicate the region of replacement. The original basis functions $\phi^{(l)}$, $l \neq k$, are represented using hollow nodes, and the derivative $d\phi^{(k)}/dx$ is distinguished using a filled node.

$O(dKL^2 + dr^2L^2)$.

Remark 1. *Our approach relies on the assumption that the equilibrium density of the system can be efficiently represented or approximated in MPS/TT format. The MPS/TT structure makes it possible to perform numerical integration for each individual tensor core, thus resulting in a computational complexity that is linear in d. When the assumptions are violated, the TT rank of the equilibrium density can grow with d, indicating the true computational complexity can be higher than our analysis.*

4.3.3 Constructing H^A and H^B

The tensors H^A and H^B derive from the two soft boundary penalty terms $\rho \int_\Omega q(\boldsymbol{x})^2 p_A(\boldsymbol{x})\, d\boldsymbol{x}$ and $\rho \int_\Omega (q(\boldsymbol{x}) - 1)^2 p_B(\boldsymbol{x})\, d\boldsymbol{x}$ in (4.2.7). The construction of the MPO format for H^A and H^B is similar to that of H^k detailed above in Section 4.3.2. However, we now further need to represent the soft boundary measures p_A and p_B as MPS/TT. Recall our motivation that p_A and p_B weakly approximate surface measures on the boundaries ∂A and ∂B, though for *any* choice of p_A and p_B we may still interpret the soft committor function probabilistically

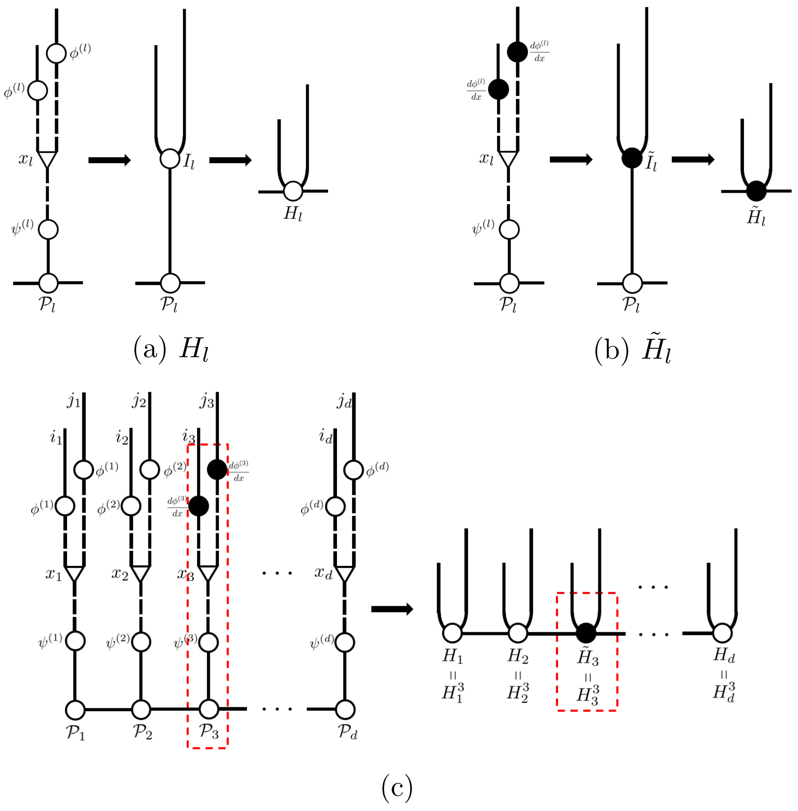


Figure 4.3.4: Diagrammatic illustration of the construction of an MPO format for H^k. (a) Precompute tensors H_l. (b) Precompute tensors $\tilde{H}_l$. (c) Assemble H^k, $k = 3$, by substituting tensors H_l and $\tilde{H}_l$. We use red dashed boxes to indicate the region of replacement. A similar construction is repeated for other k.

following C.1. The construction of approximate surface measures varies depending on the specific geometry of A and B. In many applications, A and B are balls or half-spaces, so ∂A and ∂B are spheres or hyperplanes. We discuss these two cases in detail presently.

For the case of a sphere, the Gaussian annulus theorem [29, Theorem 2.8] indicates that most of the mass of a high-dimensional Gaussian distribution concentrates on a shell. If we assume that region A is a d-dimensional ball with center $\boldsymbol{x}_A$ and radius R_A, we can approximate the uniform measure on ∂A by a Gaussian density,

$$p_A(\boldsymbol{x}) = \frac{1}{(2\pi)^{d/2}\sigma^d} \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}_A\|^2}{2\sigma^2}\right), \text{ where } \sigma = \frac{R_A}{\sqrt{d}}. \tag{4.3.8}$$

More precisely, under this probability measure, we have that

$$\text{Prob}\left(\Big| \|\boldsymbol{x}\|_2 - \sqrt{d}\sigma \Big| \geq t\sigma\right) \leq \exp\left(-\frac{t^2}{\kappa}\right), \text{ for all } t > 0, \tag{4.3.9}$$

where $\kappa > 0$ is a constant [192]. This bound indicates that the mass of p_A concentrates on a shell with radius R_A and thickness $O(1/\sqrt{d})$. It is straightforward to convert (4.3.8) into MPS/TT format since it is in fact a pure tensor product of univariate functions of each scalar variable x_k. As such our resulting MPS/TT should have ranks all equal to 1.

For the case of a hyperplane, suppose in particular that $\partial A = \{\boldsymbol{x} \in \mathbb{R}^d : x_i = c\}$. In this case consider

$$p_A(\boldsymbol{x}) = \frac{1}{(2\pi)^{1/2}\sigma} \exp\left(-\frac{(x_i - c)^2}{2\sigma^2}\right), \tag{4.3.10}$$

where σ controls the sharpness of the approximation. This choice of p_A is again a pure tensor product of univariate functions.

Now suppose that we have p_A and p_B in MPS/TT form. Specifically, assume that we have tensor cores $\{\mathcal{A}_i\}_{i=1}^d$ with associated ranks $\boldsymbol{s} = (s_0, s_1, \dots, s_d)$, together with a vector of basis functions $a^{(k)} := (a_j^{(k)})_{j=1}^{K_A^{(k)}}$ for each $k = 1, \dots, d$. Also assume that we have tensor cores $\{\mathcal{B}_i\}_{i=1}^d$ with ranks $\boldsymbol{t} = (t_0, t_1, \dots, t_d)$ and a vector of basis functions $b^{(k)} := (b_j^{(k)})_{j=1}^{K_B^{(k)}}$. Then we assume that we can write p_A and p_B as

$$p_A(\boldsymbol{x}) = \sum_{\substack{m_1,\dots,m_d \\ \alpha_0,\dots,\alpha_d}} \mathcal{A}_1(\alpha_0, m_1, \alpha_1) \dots \mathcal{A}_d(\alpha_{d-1}, m_d, \alpha_d) a_{m_1}^{(1)}(x_1) \dots a_{m_d}^{(d)}(x_d). \tag{4.3.11}$$

$$p_B(\boldsymbol{x}) = \sum_{\substack{m_1,\dots,m_d \\ \alpha_0,\dots,\alpha_d}} \mathcal{B}_1(\alpha_0, m_1, \alpha_1) \dots \mathcal{B}_d(\alpha_{d-1}, m_d, \alpha_d) b_{m_1}^{(1)}(x_1) \dots b_{m_d}^{(d)}(x_d). \tag{4.3.12}$$

In Fig. 4.3.5 we illustrate these formats graphically.

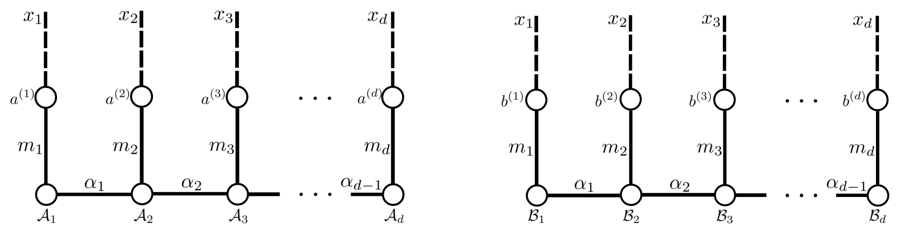


Figure 4.3.5: Tensor diagrams for (a) p_A as in (4.3.11) and (b) p_B as in (4.3.12).

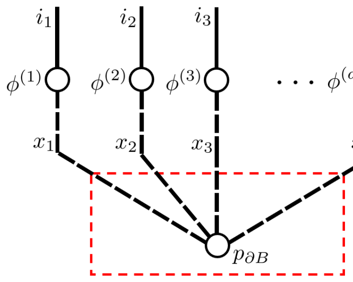
Now in light of the resemblance among (4.3.3), (4.3.4),and (4.3.5), we can use the same procedure described in Section 4.3.2 to approximate H^A and H^B as MPOs. To wit, we simply replace p in Fig. 4.3.3 with the MPS/TT approximations of $p_{\partial A}$ or $p_{\partial B}$ and replace all derivatives $d\phi^{(l)}/dx$ by $\phi^{(l)}$ since there are no derivatives in (4.3.4) and (4.3.5). Ultimately we obtain MPO formats for H_A and H_B with ranks $\boldsymbol{s}$ and $\boldsymbol{t}$ and cores H_k^A annd H_k^B, $k = 1, \ldots, d$ respectively. The computational complexities of constructing H^A and H^B are $O(ds^2K_AL^2)$ and $O(dt^2K_BL^2)$, respectively, where we define $s := \max_l s_l$, $t := \max_l t_l$, $K_A := \max_l K_A^{(l)}$, and $K_B := \max_l K_B^{(l)}$. The memory complexities are $O(dK_AL^2 + ds^2L^2)$ and $O(dK_BL^2 + dt^2L^2)$, respectively.

4.3.4 Constructing h^B

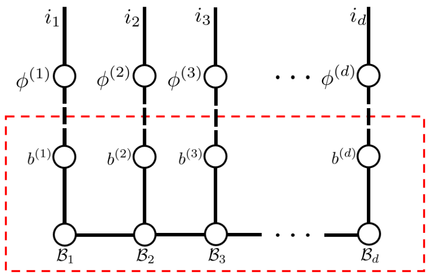
In this subsection we focus on constructing h^B, which comes from the cross term in the second penalty term $\rho \int_\Omega (q(\boldsymbol{x}) - 1)^2 p_B(\boldsymbol{x})\, d\boldsymbol{x}$ within (4.2.7). The ideas are again very similar to Section 4.3.2. The tensor diagram for h^B is shown in Fig. 4.3.6 (a). By plugging in the MPS/TT approximation of the soft boundary measure p_B (4.3.12), we obtain the approximation of h^B illustrated in Fig. 4.3.6 (b). One can further bring h^B to a standard MPS/TT form, using the contractions shown in Fig. 4.3.7. In detail, the procedure is as follows:

(1) In Fig. 4.3.7 (a), we contract the two connected basis function nodes in the red box. This results in tensors J_k, $k = 1, \ldots, d$, seen in Fig. 4.3.7 (a) on the right. This contraction requires univariate numerical integrations.

(2) Next we merge the computed tensor J_k and the tensor core $\mathcal{B}_k$ for $k = 1, \ldots, d$. The resulting 3-tensors, which are the tensor cores for h^B, are denoted h_k^B. This step is shown in Fig. 4.3.7 (b).

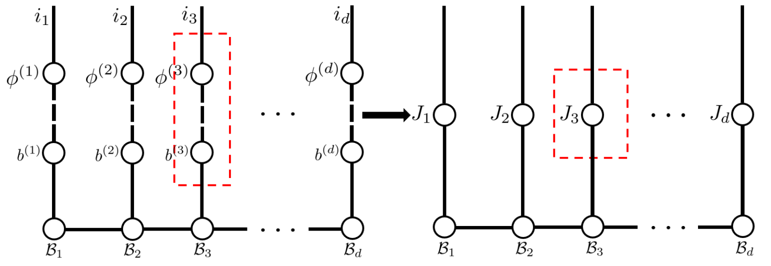


(a) h^B

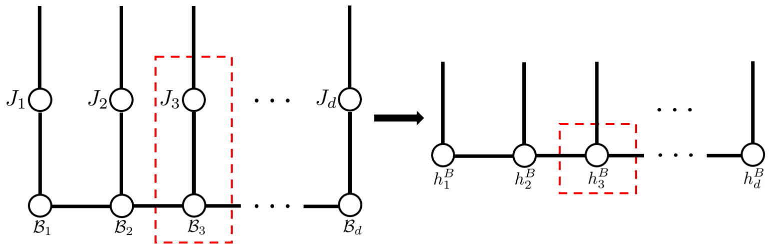


(b) Approximation of h^B

Figure 4.3.6: (a) Tensor diagram for h^B as in (4.3.6). (b) Approximation of h^B obtained by replacing the soft boundary measure p_B with its MPS/TT approximation (4.3.12). We use red dashed boxes to indicate the region of replacement.



(a)



(b)

Figure 4.3.7: Illustration of the construction of h^B in MPS/TT format. (a) Contract tensors $\phi^{(l)}$ and $b^{(l)}$ to get J_l. (b) Contract tensors J_l and $\mathcal{B}_l$ to get h_l^B. We use red dashed boxes emphasize the contractions.

This procedure yields h^B in MPS/TT format with tensor cores $h_1^B, \ldots, h_d^B$. The computational and memory complexities of constructing h^B are $O(dt^2 K_B L)$ and $O(dK_B L + dt^2 L)$, respectively.

4.3.5 Optimization

We have discussed how the MPS/TT format can be used to compress the tensors $\{H^k\}_{k=1}^d$, H^A, H^B, and h^B. In order to obtain a tractable algorithm for computing the committor function, it is natural to represent the unknown tensor $\mathcal{Q}$ in a compatible format. Indeed, without imposing some additional structure on the parameterization (4.3.1), the unknown tensor core $\mathcal{Q}$ is still of size exponential in d. Thus we approximate $\mathcal{Q}$ as in MPS/TT format as

$$\mathcal{Q}(\boldsymbol{i}) := \sum_{\alpha_0,\ldots,\alpha_d} \mathcal{Q}_1(\alpha_0, i_1, \alpha_1)\mathcal{Q}_2(\alpha_1, i_2, \alpha_2) \ldots \mathcal{Q}_d(\alpha_{d-1}, i_d, \alpha_d). \tag{4.3.13}$$

The tensor diagram for q is shown in Fig. 4.3.8. Empirically we observe that this format is able to capture the structure of q accurately, i.e., without growth of the ranks of the tensor cores. The MPS/TT format (4.3.13) for q greatly simplifies the solution of the variational problem (4.3.2). In Fig. 4.3.9 we compare the tensor diagram depictions of the original variational problem and the new simplified problem by replacing $\mathcal{Q}$ with its MPS/TT approximation. We note that all terms in the simplified form (Fig. 4.3.9 (b)) can be computed with standard MPO-MPS or MPS-MPS contractions in $O(d)$ time.

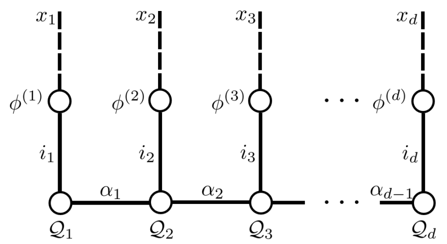
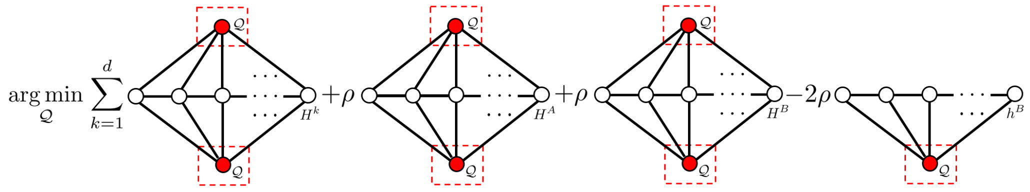


Figure 4.3.8: Tensor diagram for the parametrization of q following (4.3.1) and (4.3.13).


(a) Original variational problem

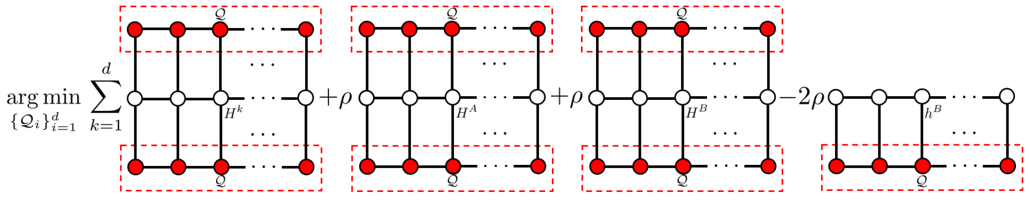

(b) Approximated variational problem

Figure 4.3.9: (a) Tensor diagram representation of the variational problem corresponding to (4.3.2). (b) Approximate variational problem obtained by replacing $\mathcal{Q}$ with its MPS/TT approximation (4.3.13). We use red dashed boxes to indicate the region of replacement. Specifically the red tensor cores are unknown variables in the optimization.

In Fig. 4.3.9, the unknown tensor cores of MPS $\mathcal{Q}$ are marked in red. A standard approach for optimization problems of the form Fig. 4.3.9 (b) is alternating least squares (ALS). In each ALS iteration, we loop over the dimensions $k = 1, \ldots, d$. For each k, we treat all coefficient tensor cores but $\mathcal{Q}_k$ as constant. This yields an unconstrained least squares problem for $\mathcal{Q}_k$. Naively the computational complexity is $O(d^2)$ since the bottleneck is the summation of d terms in Fig. 4.3.9 (b) and each term requires at least $O(d)$ tensor contractions. However by using the same trick as in the construction of H^k and carefully reusing the computed nodes, one can bring the computational complexity down to $O(d)$.

Remark 2. *Similar to the discussions for the equilibrium density, our complexity analysis relies crucially on the assumption that the TT ranks of the committor function are bounded by a constant as the dimension grows. Here we parametrize the committor function by an MPS/TT with fixed TT rank. In practice we can tune the rank parameter and monitor the numerical rank between the cores. If the rank grows rapidly with the dimensions, our complexity analysis can underestimated the true computational complexity and MPS/TT may* *not be the most efficient format of parametrizing the true committor function.*

4.4 Numerical experiments

In this section, we present numerical results that demonstrate the accuracy and efficiency of the proposed method.

4.4.1 Double-well potential

In the first numerical experiment, we consider the following potential

$$V(\boldsymbol{x}) = (x_1^2 - 1)^2 + 0.3 \sum_{i=2}^{d} x_i^2, \tag{4.4.1}$$

and we let A, B be the half-spaces

$$A = \{\boldsymbol{x} \in \mathbb{R}^d | x_1 \leq -1\}, \quad B = \{\boldsymbol{x} \in \mathbb{R}^d | x_1 \geq 1\}. \tag{4.4.2}$$

Now (4.4.1) is a double-well potential along dimension x_1, and the two boundaries ∂A and ∂B are located in the potential wells. When the temperature $T = 1/\beta$ is low, the equilibrium density $p \propto e^{-\beta V}$ is concentrated within the two wells. Meanwhile, in this case q is mostly flat with a sharp transition from 0 to 1 at $x_1 = 0$.

For this example, we can compute a ground truth solution. By symmetry, we can obtain the committor function by solving the backward Kolmogorov equation in the first dimension, i.e., setting $q_{\text{true}}(\boldsymbol{x}) = f(x_1)$, where

$$\frac{d^2 f(x_1)}{dx_1^2} - 4x_1(x_1^2 - 1)\frac{df(x_1)}{dx_1} = 0, \quad f(-1) = 0, \quad f(1) = 1. \tag{4.4.3}$$

We can solve this ODE numerically using a finite difference method on a very fine grid to produce q_{true}. The performance of our proposed method is evaluated by the following relative error metric

$$E_1 = \frac{\|q - q_{\text{true}}\|_{L^2(\Omega\backslash(A\cup B))}}{\|q_{\text{true}}\|_{L^2(\Omega\backslash(A\cup B))}}, \tag{4.4.4}$$

where $\|\cdot\|_{L^2(\Omega\backslash(A\cup B))}$ denotes the L^2-norm with respect to uniform measure over the domain $\Omega\backslash(A \cup B)$. The other error metric we use is

$$E_2 = \frac{\|q - q_{\text{true}}\|_{L^2(\Omega\backslash(A\cup B),p)}}{\|q_{\text{true}}\|_{L^2(\Omega\backslash(A\cup B),p)}}, \tag{4.4.5}$$

where $\|\cdot\|_{L^2(\Omega\backslash(A\cup B),p)}$ denotes the L^2-norm with respect to the equilibrium density p defined in (4.2.2) over the domain $\Omega\backslash(A \cup B)$.

We enforce the boundary conditions by constructing soft boundary measures p_A and p_B in MPS/TT format following (4.3.10). Meanwhile, we can exactly treat the equilibrium density p in MPS/TT format since it factorizes as a pure tensor product

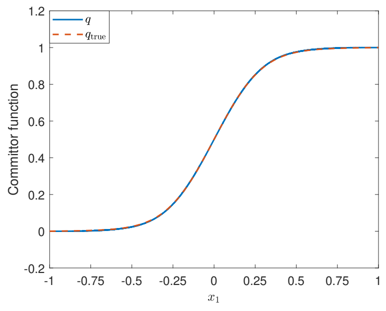
$$p(\boldsymbol{x}) = \prod_{k=1}^{d} p_k(x_k)$$

of univariate functions, given the choice of potential (4.4.1).

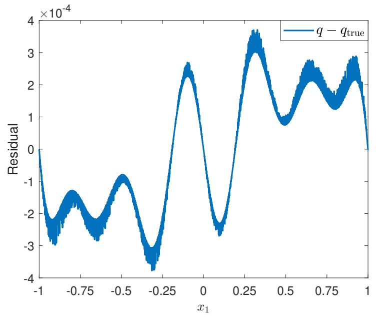
It remains to fix a univariate basis for each dimension of the committor function q. One could of course choose a generic basis such as Chebyshev polynomials, Legendre polynomials, or Fourier series. For this example, however, a better choice is to construct an appropriate truncated orthogonal polynomial basis for each dimension k according to the univariate density p_k.

We solve for the committor function at two representative temperatures $T = 0.2$ and $T = 0.05$ in $d = 20$ dimensions. For $T = 0.2$, we use the first 30 orthongonal polynomials for all dimensions. For the lower temperature $T = 0.05$, we use 60 orthogonal polynomial basis functions since the true committor function changes more sharply near $x_1 = 0$. We show q and q_{true} for $T = 0.2$ in Fig. 4.4.1 (a) and the corresponding residual $q - q_{\text{true}}$ in Fig. 4.4.1 (b). Numerical results for $T = 0.05$ are illustrated similarly in Fig. 4.4.2.

We compute the relative error E_1 in (4.4.4) using 10^5 uniformly distributed samples between $[-1, 1]$ for x_1. The relative error is $E_1 = 2.36 \times 10^{-4}$ for $T = 0.2$ and $E_1 = 1.67 \times 10^{-3}$ for $T = 0.05$. With 10^7 samples, we obtain relative error $E_2 = 1.60 \times 10^{-4}$ defined in (4.4.5) for $T = 0.2$ and $E_2 = 6.77 \times 10^{-4}$ for $T = 0.05$. Additional tests were performed with other bases such as Chebyshev polynomials and Fourier series and the behavior was similar.

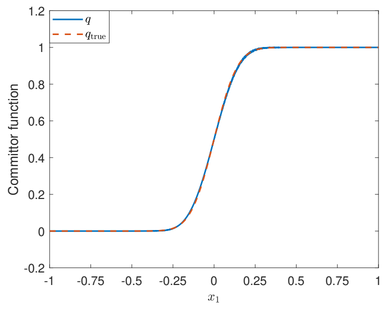


(a)

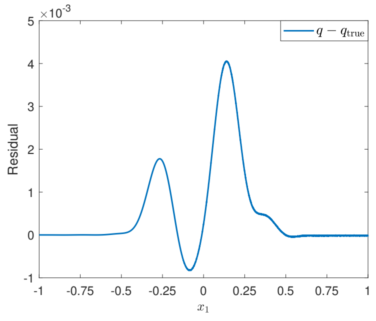


(b)

Figure 4.4.1: Numerical results for the double-well potential, $T = 0.2$. (a) The numerical solution of the committor function q, compared with the ground truth q_{true}, plotted along the x_1 dimension. (b) The residual plot $q - q_{\text{true}}$ along the x_1 dimension.



(a)



(b)

Figure 4.4.2: Numerical results for the double-well potential, $T = 0.05$. (a) The numerical solution of the committor function q, compared with the ground truth q_{true}, plotted along the x_1 dimension. (b) The residual plot $q - q_{\text{true}}$ along the x_1 dimension.

We show the numerical convergence of the solution with respect to the number of basis functions by examining the coefficient tensor of the committor function. Take $T = 0.05$ as an example where we use 60 orthogonal polynomial basis in each dimension, all basis coefficients form a 60^d coefficient tensor $\mathcal{Q} = \mathcal{Q}(i_1, \cdots, i_d)$ for $i_1, \cdots, i_d = 1, \cdots, 60$. Specifically, the first dimension is most meaningful and the rest of the dimensions are equivalent due to permutation symmetry.

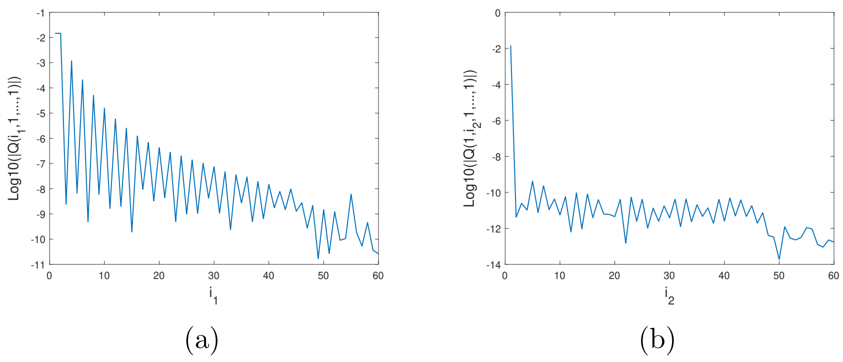


Figure 4.4.3: Coefficients of basis function representation for the double-well potential model, $T = 0.05$. (a) Log magnitude of the coefficients for the x_1 dimension $\mathcal{Q}(i_1, 1, \cdots, 1)$. (b) Log magnitude of the coefficients for the x_2 dimension $\mathcal{Q}(1, i_2, 1, \cdots, 1)$.

We visualize the numerical convergence by a slice of the committor's coefficient tensor in the first dimension (Fig. 4.4.3 (a)) and the second dimension (Fig. 4.4.3 (b)). We can observe that the basis coefficients for the first dimension decays roughly exponentially as i_1 increases. Since the committor function value should be a constant in $x_2, \ldots, x_d$, we observe the coefficient associated with $i_2 = 1$ is large while the rest of the coefficients are nearly zero.

4.4.2 Ginzburg-Landau potential

The Ginzburg-Landau theory was developed to provide a mathematical description of superconductivity [84]. In this numerical example, we consider a simplified Ginzburg-Landau model, in which the Ginzburg-Landau energy is defined for a one-dimensional scalar field

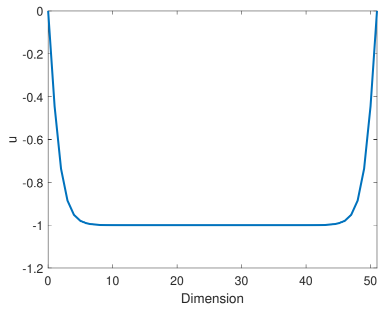
$u : [0,1] \to \mathbb{R}$ as follows:

$$\tilde{V}[u] = \int_0^1 \left[\frac{\lambda}{2}(u')^2 + \frac{1}{4\lambda}(1-u^2)^2 \right] dx, \tag{4.4.6}$$

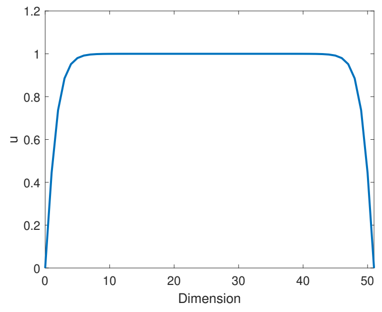
where λ is a small positive parameter and u satisfies the boundary conditions $u(0) = u(1) = 0$. We discretize u uniformly on $[0,1]$ as $U = (U_1, U_2, \ldots, U_d)$ with boundary conditions $U_0 = U_{d+1} = 0$. Then we approximate the continuous Ginzburg-Landau energy (4.4.6) with the discretization

$$V(U) := \sum_{i=1}^{d+1} \frac{\lambda}{2} \left(\frac{U_i - U_{i-1}}{h} \right)^2 + \frac{1}{4\lambda}(1 - U_i^2)^2, \tag{4.4.7}$$

where the grid spacing $h = 1/(d+1)$. We fix $d = 50$ and $\lambda = 0.03$. Note that $V(U)$ has two global minima $U_\pm$ illustrated in Fig. 4.4.4. We let A and B be the balls $\{U : \|U - U_\pm\| \leq R\}$ centered at the global minima. The radius R is set to be 2.5, chosen such that the balls A and B roughly contain the regions of high equilibrium probability density around the two centers.



(a) Local minimizer U_-



(b) Local minimizer U_+

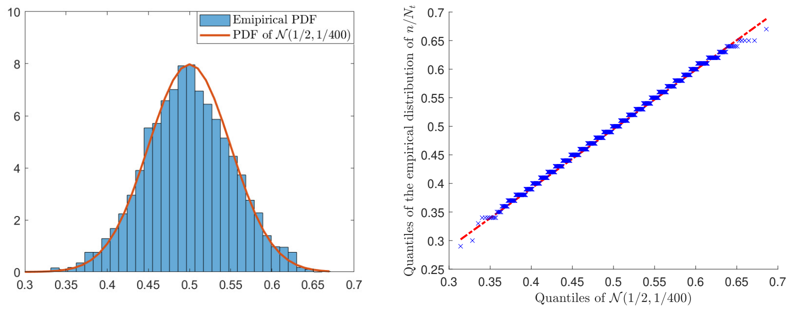
Figure 4.4.4: The two global minima of the Ginzburg-Landau energy (4.4.7) with $d = 50$ and $\lambda = 0.03$.

We present numerical results for two representative temperatures $T = 8$ and $T = 16$. We enforce the boundary conditions by constructing soft boundary measures p_A and p_B in MPS/TT format following (4.3.8). We detail the approximation of the equilibrium probability density p in MPS/TT format in Appendix C.2.

We define the domain to be the hypercube $\Omega = [-\gamma, \gamma]^{50}$. Based on our choices of λ and β, we take $\gamma = 2.6$ since this choice guarantees that the equilibrium density has negligible mass outside of Ω. To represent the committor function q, we use the first 5 Fourier basis functions $\{1, \cos(\pi x/\gamma), \sin(\pi x/\gamma), \cos(2\pi x/\gamma), \sin(2\pi x/\gamma)\}$ for each dimension. The ranks of the coefficient MPS/TT $\mathcal{Q}$ are all taken to be 6. We initialize all the entries of the unknown tensor cores of $\mathcal{Q}$ with normal $\mathcal{N}(0, 1)$ random numbers and then perform ALS, gradually increasing the penalty parameter ρ to better enforce the boundary conditions. In practice we observe that different initializations have little effect on the output of the algorithm.

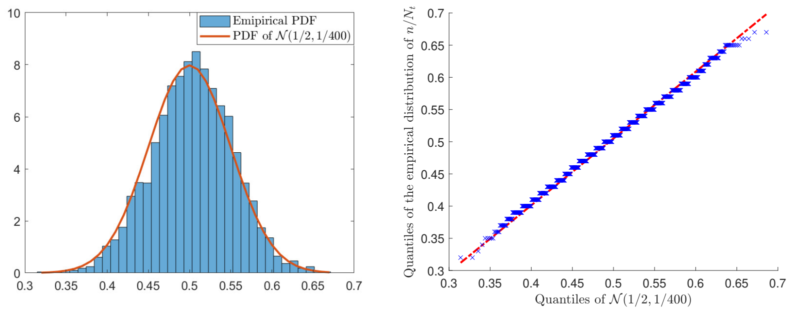
For problems of this size, traditional methods are intractable, making it difficult to obtain an exact reference q_{true} for comparison. Instead, as a proxy we study a 'thickened isosurface' around $q = 0.5$, defined as $\Gamma_\epsilon = \{U : \|q(U) - 0.5\| \leq \epsilon\}$, where $\epsilon > 0$ is a small threshold parameter. If the solution q is indeed a satisfactory approximation of the true committor function, then for any trajectories given by (4.2.1) starting from points in Γ_ϵ, the probability of entering region B before A should be close to 0.5.

To verify this, we generate samples from the equilibrium distribution by simulating the process (4.2.1). Then we filter to keep samples on the isosurface Γ_ϵ using the computed committor function q and the threshold ϵ of our choice. Let us pick N_s points in Γ_ϵ, denoted $\{\tilde{U}_j\}_{j=1}^{N_s}$. For each point $\tilde{U}_j$, we generate N_t trajectories by simulating the Langevin process (4.2.1) and use n_j to denote the number of trajectories ending up in region B before A. By the central limit theorem, when N_t is large, the distribution of n_j/N_t should be well-approximated by the normal distribution $\mathcal{N}(\frac{1}{2}, (4N_t)^{-1})$. In our numerical tests, we set $\epsilon = 5 \times 10^{-3}$, $N_s = 5000$, and $N_t = 100$. The results for $T = 8$ and $T = 16$ are illustrated in Fig. 4.4.5 and Fig. 4.4.6, respectively. We compare the histogram of $\{n_j/N_t\}_{j=1}^{5000}$ with the normal distribution $\mathcal{N}(\frac{1}{2}, 1/400)$ on the left and show the Q–Q (quantile-quantile) plot of the distribution of $\{n_j/N_t\}_{j=1}^{5000}$ versus $\mathcal{N}(\frac{1}{2}, 1/400)$ on the right. These figures demonstrate that the distribution of $\{n_j/N_t\}_{j=1}^{5000}$ is indeed in good agreement with the normal distribution $\mathcal{N}(\frac{1}{2}, 1/400)$, which indicates that our solution q provides a good approximation of the true isosurface.



(a) Histogram (b) Q-Q plot

Figure 4.4.5: Comparison of distributions for $T = 8$. (a) Empirical histogram for $\{n_j/N_t\}_{j=1}^{5000}$ compared with the density of $\mathcal{N}(\frac{1}{2}, 1/400)$. (b) Q-Q (quantile-quantile) plot of $\{n_j/N_t\}_{j=1}^{5000}$ compared with $\mathcal{N}(\frac{1}{2}, 1/400)$.



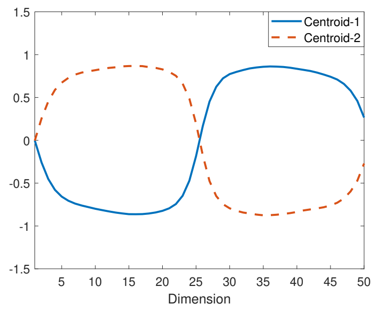
(a) Probability density (b) Q-Q plot

Figure 4.4.6: Comparison of distributions for $T = 16$. (a) Empirical histogram for $\{n_j/N_t\}_{j=1}^{5000}$ compared with the density of $\mathcal{N}(\frac{1}{2}, 1/400)$. (b) Q-Q (quantile-quantile) plot of $\{n_j/N_t\}_{j=1}^{5000}$ compared with $\mathcal{N}(\frac{1}{2}, 1/400)$.

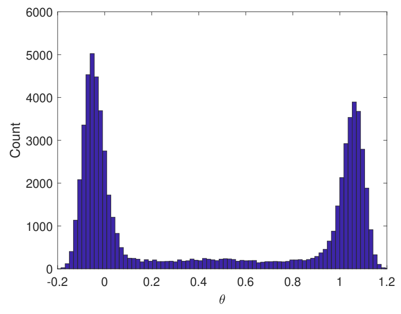
Next, consider the restriction of the equilibrium density to the $q = 0.5$ isosurface. Intuitively, the first term in the Ginzburg-Landau potential (4.4.7) encourages the configuration U to be as flat as possible. Therefore, to transition between the two boundary states U_- and U_+, it is favorable in terms of energy to have only a single sign change in the discretized function U. We compute 10^7 samples from the equilibrium density by running the overdamped

Langevin process (4.2.1), initialized at random states in Ω. We retain only the samples that fall in the thickened isosurface Γ_ϵ, $\epsilon = 5 \times 10^{-3}$. Then we perform 2-means clustering on these samples. In Fig. 4.4.7 (a) and Fig. 4.4.8 (a), we show the centroids $U^{(1)}$ and $U^{(2)}$ of the two clusters for $T = 8$ and $T = 16$, respectively. These configurations are symmetric with a single sign change.

Next we project all samples in the $q = 0.5$ isosurface to the line containing the two centroids, i.e., to points of the form $\theta U^{(1)} + (1 - \theta)U^{(2)}$. In Fig. 4.4.7 (b) and Fig. 4.4.8 (b) we plot the histograms of θ for all samples to demonstrate that these distributions are indeed bimodal. Observe that at higher temperature, the bimodality is less pronounced.



(a) Two centroids



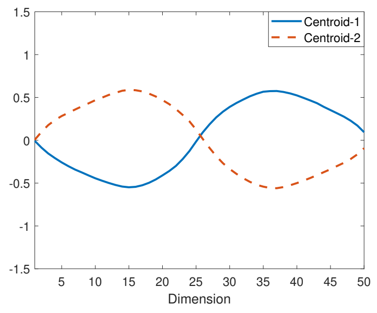
(b) Histogram of θ

Figure 4.4.7: Analysis of the $q = 0.5$ isosurface for $T = 8$. (a) Centroids of the two clusters in the $q = 0.5$ isosurface. (b) Histogram of the 1-dimensional coordinate θ of the isosurface samples along the line between the two clusters.

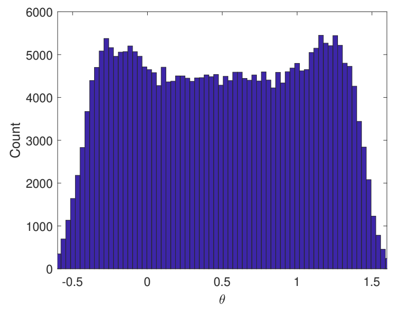
Finally, we study transition paths via the deterministic reactive flow [107]:

$$\frac{dU(t)}{dt} = \frac{1}{\beta} p(U(t)) \nabla q(U(t)). \tag{4.4.8}$$

Based on Fig. 4.4.7, we expect that at low temperatures the transition paths between A and B are localized within in two reaction tubes. We visualize one of the transition paths at temperature $T = 8$ in Fig. 4.4.9. The leftmost curve corresponds to the initial state of (4.4.8), for which $q = 0.1$. Meanwhile $q = 0.9$ for the rightmost curve. The red arrow indicates the direction of time evolution.



(a) Two centroids



(b) Histogram of θ

Figure 4.4.8: Analysis of the $q = 0.5$ isosurface for $T = 16$. (a) Centroids of the two clusters in the $q = 0.5$ isosurface. (b) Histogram of the 1-dimensional coordinate θ of the isosurface samples along the line between the two clusters.

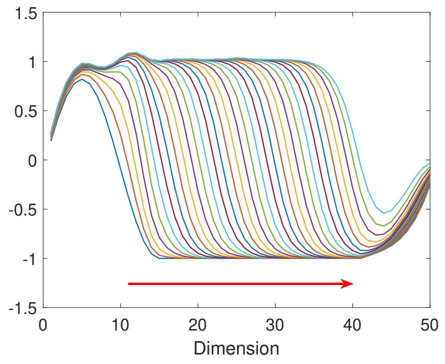


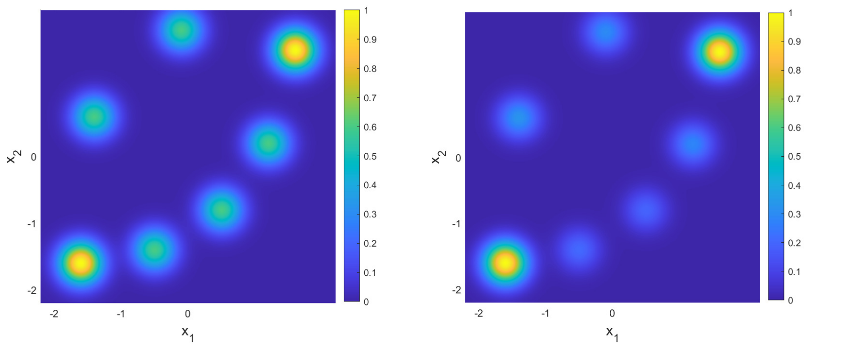
Figure 4.4.9: Visualization of one transition path for $T = 8$.

4.4.3 Gaussian Mixture Equilibrium Density

To evaluate the model performance on rugged energy landscape, we consider constructing a more complicated equilibrium distribution with several isolated local maxima using Gaussian mixture models. In this example we use a mixture of 7 Gaussian densities,

$$p(\boldsymbol{x}) = \exp(-\beta\|\boldsymbol{x} - C_A\|) + \exp(-\beta\|\boldsymbol{x} - C_B\|) + 0.6\sum_{i=1}^{3}\exp(-\beta\|\boldsymbol{x} - C_i\|) + \eta\sum_{i=4}^{5}\exp(-\beta\|\boldsymbol{x} - C_i\|). \tag{4.4.9}$$

where C_A, C_B and C_i for $i = 1, \cdots, 5$ are the 7 Gaussian centers. η is some parameter. In this example let the problem dimension $d = 10$ and all the centers be roughly contained in the first two dimensions for visualization purpose. Specifically for the first two dimensions, $C_A = (-1.6, -1.6)$, $C_B = (1.6, 1.6)$, $C_1 = (-0.5, -1.4)$, $C_2 = (0.5, -0.8)$, $C_3 = (1.2, 0.2)$, $C_4 = (-1.4, 0.6)$, $C_5 = (-0.1, 1.9)$. If we pad all 8 other dimensions with 0, the centers strictly lie on a 2-dimensional subspace. In Fig. 4.4.10 (a) we show the density in 2D. To make the problem more difficult, we perturb the mean of these Gaussians with small independent Gaussian noise $0.1N(0, 1)$. The magnitude of the perturbation is chosen such that the Gaussian centers can still be visualized in the first two dimensions. In Fig. 4.4.10 (b) we show the "perturbed" density, which is be used in the following numerical tests.

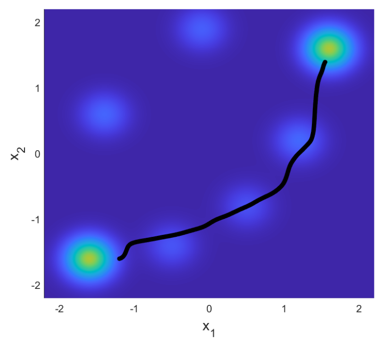


(a) Equilibrium density in 2D (b) Equilibrium density in high dimension

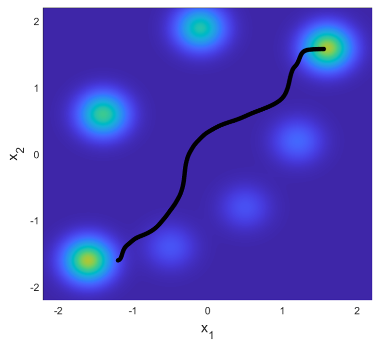
Figure 4.4.10: The Gaussian mixture model equilibrium density visualized in the first two dimensions. (a) Equilibrium density in 2D. (b) Equilibrium density where the mean of Gaussians are in high dimension.

Note the five Gaussian centers in the middle are roughly aligned on two curves: a lower curve containing C_1, C_2, C_3 and an upper curve containing C_4, C_5. Both curves share C_A and C_B as their end points. Similarly we study the transition paths between the two boundaries by simulating the reactive flow (4.4.8). Specifically we study how the transition path changes with the additional local maxima of the equilbrium density. To this end, we pick two η values 0.6 and 1.6 to alter the magnitude the top two Gaussian densities.

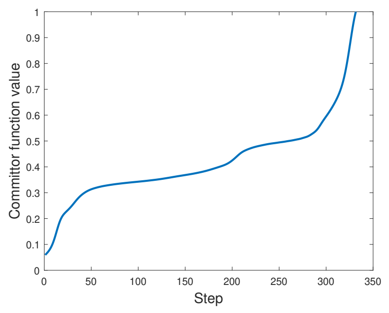
Let the boundaries A and B be the balls centered at two centers $\{\boldsymbol{x} : \|\boldsymbol{x} - C_A\| \leq R\}$ and $\{\boldsymbol{x} : \|\boldsymbol{x} - C_B\| \leq R\}$, respectively. In this example $R = 0.22$, chosen such that balls A and B roughly contained the regions of high equilibrium density around the two centers. The temperature is set to be $T = 0.1$. We define the solution domain to be the hypercube $\Omega = [-2.4, 2.4]^{10}$, which guarantees the equilibrium density has negligible mass outside the solution domain. We use the first 60 Fourier basis to parameterize the committor function. The ranks of the coefficient MPS/TT $\mathcal{Q}$ are all taken to be 4.



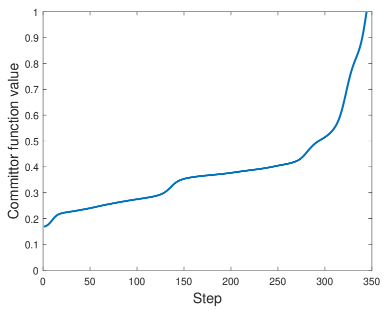
(a) $\eta = 0.6$ transition trajectory



(b) $\eta = 1.6$ transition trajectory



(c) $\eta = 0.6$ committor function values along the path



(d) $\eta = 1.6$ committor function values along the path

Figure 4.4.11: Transition paths (black curves) overlayed on the Gaussian mixture model equilibrium density visualized in the first two dimensions for (a) $\eta = 0.6$ and (b) $\eta = 1.6$. Committor function values along the transition path for (c) $\eta = 0.6$ and (d) $\eta = 1.6$.

In Fig. 4.4.11, we show the transition path between boundaries A, B for $\eta = 0.6, 1.6$. Both the transition paths share the same starting point $[-1.2, -1.6]$. We can observe that the transition path is shifting towards the top as the magnitude η increases.

4.5 Conclusion

In this chapter, we propose a novel approach for computing high-dimensional committor functions using MPS/TT. In particular, we start from the variational formulation (4.2.7) for the soft committor function, which can be viewed as an approximation of the committor function but which also enjoys a probabilistic interpretation in its own right. To compute high-dimensional integrals, we approximate the equilibrium density and soft boundary measures in MPS/TT format. Meanwhile, the unknown committor function q is also parametrized in MPS/TT format. The variational problem can then be reformulated using standard MPO and MPS/TT operations, and the optimization of q can be performed with $O(d)$ complexity. Extensive numerical experiments demonstrate the computational efficiency and accuracy of the proposed method.

Notice that our proposed method relies on the assumption of certain structures of the variables in order to approximate the equilibrium density efficiently in MPS/TT format. One expects the MPS/TT-based approximation to be successful in a very high-dimensional limit when there is a 1D or quasi-1D graphical model structure underlying the equilibrium measure. Already we believe that the capacity to treat such systems with high accuracy is a meaningful contribution, since many previous works in the literature have considered problems with such structure, such as the double-well and rugged-Muller potentials (embedded in high dimensions) and the Ginzburg-Landau potential [95, 107, 113, 155]. Even in the case without obvious 1D ordering, MPS/TT have demonstrated excellent empirical performance in high-dimensional PDE and chemistry [152, 11, 164, 15].

However in principle our method can also be extended to more complicated systems and more general network with significant interactions between any particles, provided that suitable tensor network contractions can be performed. It has become an active research field of extending the tensor network algorithm to more general systems, see for example [88]. Our complexity analysis relies on the assumption that the rank of q remains constant as the number of dimension increases, which may not hold for complicated networks. As a future work, one can monitor the rank of the committor function q via a two-site alternating minimization scheme. If we see any signs of rapidly growing rank, it may indicate the method is more computationally expensive than expected and MPS/TT is not the most efficient format to represent the true committor function. As a future work, one can monitor the rank of the committor function q via a two-site alternating minimization scheme [65].

CHAPTER 5
STATISTICAL MECHANICS AND QUANTUM MANY BODY SIMULATION WITH AUXILIARY-FIELD MONTE CARLO AND MATRIX PRODUCT STATE

5.1 Introduction

Statistical mechanics [142, 99] is a branch of physics that deals with the behavior of systems consisting of a large number of particles. It seeks to explain the observed macroscopic phenomena, such as temperature, pressure, and entropy, in terms of the statistical properties of the underlying particles. It provides a mathematical framework to understand the statistical properties of particle ensembles from the microscopic laws of physics. Statistical mechanics has broad applications in modeling a wide range of phenomena, including phase transitions [72, 41], chemical reactions [103, 222], and the behavior of complex physical/biological systems [92, 166].

One of the fundamental concepts in statistical mechanics is the Boltzmann distribution [118], which describes the probability of the particle ensemble in a given energy state in thermal equilibrium. The Boltzmann distribution is derived from the principle of maximum entropy, which states that in thermal equilibrium, a system will be in the state with the maximum number of microstates consistent with the macroscopic constraints of the system.

While the Boltzmann distribution is a powerful tool for understanding the behavior of small-scale particle systems, it becomes increasingly difficult to calculate due to the exponential increase in the number of microstates as the number of particles increases. On the other hand, the interactions between particles in large-scale systems can be complex and difficult to model accurately, further complicating the calculation of the Boltzmann distribution.

These obstacles have motivated researchers to develop various approaches to approximate the density and circumvent the exponential scaling. Monte Carlo simulations have been widely used in statistical mechanics [108, 19], which involve generating a large number of random samples of the system and using statistical methods to estimate the distribution. However, the convergence rate of simulating large systems often makes it challenging to obtain accurate approximations. Another approach is to use machine learning techniques to learn the Boltzmann distribution directly from data. This approach, known as Boltzmann machine learning [172, 169, 161], has shown promising results in several applications, including modeling of protein folding and predicting material properties. Similar to the Monte Carlo approach, the performance of this data-driven technique depends crucially on the data quality and the number of samples available. The mean-field theory [203, 181, 89] assumes that each particle interacts with an average field that is determined by the other particles, simplifying the dynamics of the system and the calculation of the distribution. However, mean-field theory is limited by its assumptions and may not accurately capture the behavior of some systems with complex interactions.

The auxiliary field Monte Carlo (AFMC) method [14, 12, 13] is a powerful numerical technique that has been developed to overcome some of the limitations of traditional Monte Carlo simulations. AFMC is based on the idea of introducing auxiliary fields to decouple the correlations between particles by means of the application of the Hubbard–Stratonovich transformation [85]. This reduces the many-body problem to the calculation of a sum or integral over all possible auxiliary-field configurations. The method and its quantum version has been successfully applied to a wide range of problems in statistical mechanics, including lattice field theory, quantum chromodynamics, and condensed matter physics [220, 42, 168, 145, 110].

On the other hand, tensor network methods [61, 139, 193, 138, 64, 97] have emerged as an efficient tool to model the high-dimensional particle system in modern quantum physics and beyond. Specifically, tensor network methods exhibit extraordinary performances for systems with low intrinsic dimensions or special coupling structures. Recently, the authors [50] propose to solve the committor function for high-dimensional interacting systems given the equilibrium Boltzmann distribution can be efficiently compressed in MPS/TT format. Indeed, the MPS/TT format significantly reduces the storage requirements and computational costs associated with evaluating the distribution, making it more accessible and practical for large-scale simulations. It also provides a compact representation of the distribution that is amenable to analysis and manipulation, allowing researchers to extract meaningful insights from the data. Furthermore, authors [86, 151, 182] show that one can use tensor networks to do density estimations given available samples from the target distribution.

Additionally, we aim to provide a unified framework for particle-based time evolution algorithms based on tensor network sketching. Assume the propagator is denoted by G and the quantity we want to propagate is denoted by ϕ, our goal is to iteratively approximate $\phi_{t+1} = G\phi_t$. For example, in classical statistical mechanics, G can be the Fokker-Planck operator and ϕ_t is the density of a given time t. In quantum mechanics G can be the imaginary time propagator and ϕ is the wavefunction. For large scale and high-dimensional problems, significant efforts have been spent in approximating the propagator G with stochastic particles, e.g. classical Monte Carlo method, AFMC, etc. and approximating the quantity of interest using tensor networks. In this work, we provide a general framework of combining the two approaches together, i.e. propagating the tensor networks with stochastic samples and recompressing the tensor networks with sketching.

In this chapter, we aim to explore the potential of compressing the Boltzmann distribution in an efficient low-complexity MPS/TT format given samples from the auxiliary-field. On the one hand, AFMC provides a powerful approach to generate necessary samples for TT sketching from the auxiliary field. On the other hand, the low-complexity MPS/TT format implicitly regularize the problem to reduce the number of required samples to achieve accurate approximations.

5.2 Proposed Method

5.2.1 *Recap of AFMC*

One of the most widely studied probabilistic models in statistical physics and machine learning is the *Ising model*, which is a probability distribution on the hypercube $\{\pm 1\}^d$. Consider a set of d lattice sites, there is a discrete variable x_i such that $x_i \in \{\pm 1\}$ represents the site's spin. The energy of a spin configuration for the system is given by the Hamiltonian function

$$H(x) = -\sum_{i \neq j} J_{i,j} x_i x_j - \mu \sum_i h_i x_i = x^T J x + \mu h^T x, \tag{5.2.1}$$

where the first sum defines the interactions between adjacent spins, μ denotes the magnetic moment. Note we have absorbed the negative sign to interaction matrix J and vector h respectively for simplicity. The sign of h controls the direction of the external magnetic field. The probability of a given configuration is given by the Boltzmann distribution

$$P(x) := \frac{1}{Z} \exp\left(-\beta H(x)\right) = \frac{1}{Z} \exp\left(-\beta x^T J x - \beta \mu h^T x\right), \tag{5.2.2}$$

where $\beta = (k_{\mathrm{B}} T)^{-1}$ is the inverse of temperature and $Z = \sum_{x \in \{\pm 1\}^d} \exp\left(-\beta H(x)\right)$ is the partition function. In this chapter we will use the Ising model extensively to demonstrate our approach. But the procedure can be generally extended to systems with more complicated pairwise interactions.

The key idea of AFMC is to decouple the interactions on the spin space by applying the Hubbard-Stratonovich transformation. In the classical setting, it is necessary to have the interaction matrix J be positive-definite to perform the transformation. This can be easily achieved by adding a diagonal nugget to J. Since $(\pm 1)^2 = 1$, the nugget effect can be undone by dividing the resulting Boltzmann distribution by a positive constant. Therefore, without loss of generality, we assume J is positive-definite for the remainder of the chapter. To decouple the interactions through orthogonal transformations, we first perform an eigenvalue decomposition $J = U\Sigma U^T$ and compute

$$\begin{aligned}\exp\left(-\beta H(x)\right) &= \exp\left(-\beta\mu h^T x\right)\exp\left(-\beta x^T U\Sigma U^T x\right)\\ &= \exp\left(-\beta\mu h^T x\right)\ \Pi_{i=1}^{d}\exp(-\beta\lambda_i(u_i^T x)^2),\end{aligned}\tag{5.2.3}$$

where $\lambda_i = \Sigma_{i,i}$ are the eigenvalues of J and $U = [u_1, u_2, \cdots, u_d]$. Now we perform Fourier transform to the second term of (5.2.3),

$$\begin{aligned}\exp(-\beta H(x)) = \exp\left(-\beta\mu h^T x\right)\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty}(2\pi)^{-d/2}\left(\Pi_{i=1}^{d}\exp\left(-k_i^2/2\right)\right)\\ \left(\Pi_{i=1}^{d}\exp\left(\sqrt{-2\beta\lambda_i}k_i(u_i^T x)\right)\right)dk_1\cdots dk_d.\end{aligned}\tag{5.2.4}$$

Instead of sampling directly from the Boltzmann distribution on the spin space (5.2.2), AFMC approximates the d-dimensional integration in (5.2.4) by Monte Carlo integration from the Fourier space. Replacing the integration with Monte Carlo samples, we get the following approximation,

$$\exp(-\beta H(x)) \approx \exp\left(-\beta\mu h^T x\right)\sum_{j=1}^{N}\left(\Pi_{i=1}^{d}\exp\left(\sqrt{-2\beta\lambda_i}k_i^j(u_i^T x)\right)\right),\tag{5.2.5}$$

where N is the total number of Monte Carlo samples, k_i^j is the j-th sample for the frequency variable of the i-th dimension. All samples $\{k_i^j\}_{i=1,\cdots,d,\ j=1,\cdots,N}$ are drawn from i.i.d standard normal distribution. Note that we ignore the partition function Z from the Boltzmann distribution and focus on approximating $\exp(-\beta H(x))$ in the rest of the chapter. As we will see later, once we obtain a low-complexity representation (e.g. MPS/TT) of $\exp(-\beta H(x))$, the partition function can be easily calculated.

5.2.2 MPS/TT Density Estimations with AFMC Sampling

Detailed introductions about tensor train, general tensor networks and tensor diagrams can be found in Section 4.2.3. To get a MPS/TT representation of the approximation, we reorganize each term in the summation of (5.2.5),

$$\begin{aligned}\Pi_{i=1}^{d} \exp\left(\sqrt{-2\beta\lambda_i} k_i^j (u_i^T x)\right) =& \Pi_{i=1}^{n} \exp\left(\sum_{l=1}^{d} \sqrt{-2\beta\lambda_i} k_i^j (u_{l,i} x_l)\right) \\ =& \Pi_{l=1}^{d} \exp\left(\sum_{i=1}^{d} \sqrt{-2\beta\lambda_i} k_i^j (u_{l,i} x_l)\right) \end{aligned} \tag{5.2.6}$$

Each $\exp\left(\sum_{i=1}^{d} \sqrt{-2\beta\lambda_i} k_i^j (u_{l,i} x_l)\right)$ only depends on the spin x_l of a single dimension. The product can be chained as a rank-1 MPS/TT. Additionally, the first term in (5.2.5) is naturally a chained product of operators in individual dimensions, which can be easily written in MPS/TT representations,

$$\exp(-\beta\mu h^T x) = \Pi_{l=1}^{d} \exp(-\beta\mu h_l x_l). \tag{5.2.7}$$

Combine the two parts together, we can approximate the Boltzmann distribution as follows,

$$\exp(-\beta H(x)) \approx \sum_{j=1}^{N} \left(\Pi_{l=1}^{d} \exp\left(\sum_{i=1}^{d} \sqrt{-2\beta\lambda_i} k_i^j (u_{l,i} x_l) - \beta\mu h_l x_l\right)\right). \tag{5.2.8}$$

A tensor diagram approximation of the Boltzmann distribution is shown in Fig. 5.2.1.

5.2.3 Recompression via TT Sketching

In Fig. 5.2.1, we approximate the original Boltzmann distribution $\exp(-\beta H(x))$ as a sequential summation of N rank-1 MPS/TT. This representation offers computational advantages

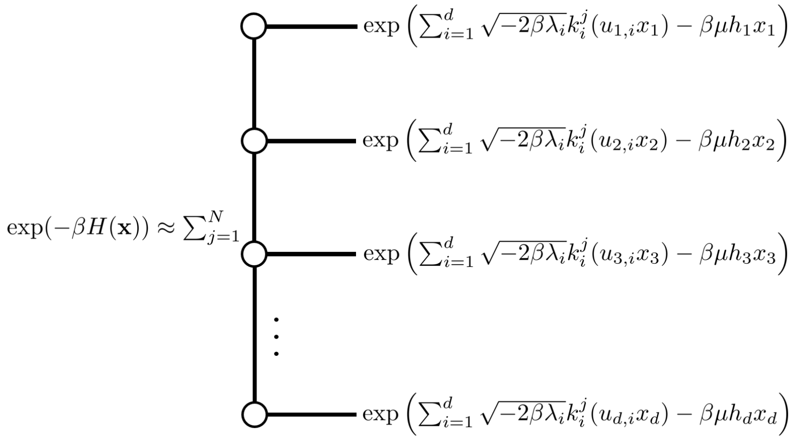

$\exp(-\beta H(\mathbf{x})) \approx \sum_{j=1}^{N}$
$\exp\left(\sum_{i=1}^{d} \sqrt{-2\beta\lambda_i} k_i^j (u_{1,i} x_1) - \beta\mu h_1 x_1\right)$
$\exp\left(\sum_{i=1}^{d} \sqrt{-2\beta\lambda_i} k_i^j (u_{2,i} x_2) - \beta\mu h_2 x_2\right)$
$\exp\left(\sum_{i=1}^{d} \sqrt{-2\beta\lambda_i} k_i^j (u_{3,i} x_3) - \beta\mu h_3 x_3\right)$
$\exp\left(\sum_{i=1}^{d} \sqrt{-2\beta\lambda_i} k_i^j (u_{d,i} x_d) - \beta\mu h_d x_d\right)$

Figure 5.2.1: Tensor diagram for approximating the Boltzmann distribution as a sum of MPS/TT based on (5.2.8).

using efficient tensor algorithms. However, the bottleneck of the representation is the MPS/TT rank. For instance, the sum of N rank-1 MPS/TT results in a rank-N MPS/TT. Obtaining an accurate approximation to the Monte Carlo integration may require many samples, leading to a high-rank MPS/TT representation. Nevertheless, the true rank of the MPS/TT representation should be determined by the Hamiltonian H intrinsically and independent of the number of samples used. We aim to round or recompress the resulting high-rank MPS/TT to reveal the true rank of the Boltzmann distribution and control the representation's complexity.

Recently, several works have been developed on MPS/TT density estimations from empirical samples, such as [86, 182, 151]. The main idea is to apply low-dimensional recursive sketching on the empirical distribution and solve a series of core determining equations to reconstruct the MPS/TT approximation of the true distribution. The recursive sketches can take different forms, including kernel sketches, random sketches, and random MPS/TT sketches. In this context, we illustrate the procedure using random MPS/TT sketches in Fig. 5.2.2. For more information about other sketches and the formation of the core system equations, please refer to [86].

In this example, we are using a five-dimensional tensor diagram and starting with the

AFMC + TT approximation for the Boltzmann distribution (Fig. 5.2.2a). We assume that the true Boltzmann distribution can be written as an MPS/TT (left diagram in Fig. 5.2.2a), which is approximated by the summation of N rank-1 MPS/TT (right diagram in Fig. 5.2.2a). Our goal is to estimate the unknown tensor cores $\mathcal{Q}_1, \ldots, \mathcal{Q}_5$ given the summation of MPS/TT. We do this iteratively, and in Fig. 5.2.2 we demonstrate an example of estimating the third tensor core $\mathcal{Q}_3$, which is marked as the filled red node in the diagram.

To estimate the unknown tensor core $\mathcal{Q}_3$, we perform sketching to all dimensions before and after the target dimension $l = 3$, as shown in Fig. 5.2.2b. The sketches are pieces from a random MPS/TT, highlighted with red dashed boxes. As a result, we obtain tensor $\mathcal{A}$. Next, we perform similar sketching but also include the target dimension, as illustrated in Fig. 5.2.2c, to get matrix $\mathcal{B}$. Finally, we solve the linear system in Fig. 5.2.2d to get the unknown tensor core $\mathcal{Q}_3$. This same procedure should be performed for all dimensions in order to estimate all of the unknown tensor cores.

Remark 3. *Size of the core determining equation. Here we highlight that the size of the core determining equation remains unaffected by the number of samples and the total number of dimensions. Indeed,* 3*-tensor* $\mathcal{A}$ *has dimensionality controlled by the rank of the sketching MPS/TT and the cardinality of the target dimension (*2 *in the case of Ising model). Similarly, the dimensionality of* $\mathcal{B}$ *is fully controlled by the rank of the sketching MPS/TT. Thus, the recompression step does not pose a bottleneck for the algorithm, allowing us to generate as many samples as needed without inflating the rank of the resulting MPS/TT.*

Remark 4. *Rank of the randomized sketches. As discussed previously, the intrinsic TT-rank of the Boltzmann distribution is determined solely by the system's nature. Therefore, the randomized sketches must be selected carefully to reveal the distribution's true numerical rank. Analytically, the rank should be greater than or equal to the rank of the unfolding matrix of the true distribution [141]. To estimate the rank numerically, we can gradually increase the rank of the sketches and monitor (1) the approximation's convergence and (2)* *the singular value decay of the core determining equations. The threshold for solving the core determining equations must be set accordingly to expose the true numerical rank and eliminate sampling noise.*

We loop over all dimensions to get all tensor cores. In classical cases, our ultimate goal is to find a low-complexity representation of the Boltzmann distribution that enables efficient downstream tasks. Following the procedures detailed in Section 5.2.1, Section 5.2.2, Section 5.2.3, we now have an MPS/TT approximation of $\exp(-\beta H(x))$. We can effortlessly perform a wide range of downstream tasks taking advantages of the MPS/TT structure, e.g. calculating the partition function, evaluating the density, computing the moments, drawing i.i.d samples, drawing conditional samples from the distribution, all in $O(d)$ time.

5.2.4 Extending to Quantum Setting

We demonstrate our approach with the classical Ising model. But the similar procedure can be extended to quantum case with little additional effort. Take the quantum version of the classical Ising model – transverse-field Ising model – as an example, the Hamiltonian becomes

$$H = -\sum_{i \neq j} J_{i,j} Z_i Z_j - \mu \sum_i h_i X_i, \tag{5.2.9}$$

where Z_i and X_i are representations of elements of the spin algebra (Pauli matrices, in the case of spin 1/2) acting on the spin variables of the corresponding sites. Auxiliary-field quantum Monte Carlo (AFQMC) is the quantum version of the AFMC method [28, 12]. In the quantum case, the goal is to find energy and wavefunction of the ground state of the system, which is equivalent to finding the smallest eigenvalue and eigenvector of the operator H. AFQMC performs Trotter decomposition followed by the Hubbard-Stratonovich transformation to the propagator

$$\exp(-\beta H) \approx (I - \beta H), \tag{5.2.10}$$

where β here is the infinitesimal time step. The power method will reveal the largest eigenpairs of $(I - \beta H)$, from which we can compute the smallest eigenpairs for H accordingly.

With similar derivations, one gets AFQMC approximations of the propagator as follows

$$\exp(-\beta H) \approx \sum_{j=1}^{N} \left(\Pi_{l=1}^{d} \exp\left(\sum_{i=1}^{d} \sqrt{-2\beta\lambda_i} k_i^j (u_{l,i} Z_l) - \beta\mu h_l X_l \right) \right). \tag{5.2.11}$$

The only difference here is that now each dimension consists of operators (Pauli matrices) instead of spins. Therefore we construct each sample as a rank-1 MPO instead of MPS/TT in the classical case.

One can recompress the rank of MPO with similar sketching procedure. But in the quantum case we are interested in the application of the propagator to an arbitrary wavefunction instead of the propagator itself. Assume we parametrize the wavefunction as an MPS/TT. The power method becomes MPO (propagator) - MPS (wavefunction) product and the renormalization step is efficient for MPS/TT. We use tensor diagrams to illustrate the imaginary time evolution in Fig. 5.2.3.

The test wavefunction Φ is parametrized as an MPS/TT and highlighted with red dashed boxes. Using tensor contraction, the updated wavefunction can be expressed as a sum of N MPS/TTs, with the same expression as (5.2.8) in the classical case. We can employ the same sketching technique to compress the rank of the resulting MPS/TT approximations. The updated wavefunction, represented as an MPS/TT, is then renormalized and passed on to the next iteration.

5.2.5 Extending to Parabolic PDE

The same framework can also be applied to numerical simulations of parabolic PDEs. The theory of elliptic and parabolic equations for measures is now a rapidly growing area with deep and interesting connections to many directions in real analysis, PDEs, and stochastic analysis. In this subsection, we take overdamped Langevin process and the corresponding Fokker-Planck equations as an example. But the idea is applicable to the evolution of a general parabolic PDE. We consider a particle system governed by the overdamped Langevin process,

$$d\boldsymbol{X}_t = -\nabla V(\boldsymbol{X}_t)\, dt + \sqrt{2\beta^{-1}}\, d\boldsymbol{W}_t, \tag{5.2.12}$$

where $\boldsymbol{X}_t \in \Omega \subset \mathbb{R}^d$ is the state of the system, $V : \Omega \subset \mathbb{R}^d \to \mathbb{R}$ is a smooth potential energy function, $\beta = 1/T$ is the inverse of the temperature T, and $\boldsymbol{W}_t$ is a d-dimensional Wiener processs. If the potential energy function V is confining for Ω (see, e.g., [25, Definition 4.2]), then one can show that the equilibrium probability distribution of the Langevin dynamics (5.2.12) is the Boltzmann-Gibbs distribution

$$\hat{p}(\boldsymbol{x}) = \frac{1}{Z_\beta} \exp(-\beta V(\boldsymbol{x})). \tag{5.2.13}$$

where $Z_\beta = \int_\Omega \exp(-\beta V(\boldsymbol{x}))\, d\boldsymbol{x}$ is the partition function. Moreover, the evolution of the distribution of the particle system can be described by the corresponding Fokker-Planck equation,

$$\frac{\partial p}{\partial t} = \beta^{-1} \Delta p + \nabla \cdot (\nabla V p), \quad p(\boldsymbol{x}, 0) = p_0(\boldsymbol{x}), \tag{5.2.14}$$

where p_0 is the initial distribution. For any time $t_2 > t_1 \geq 0$, one can approximate the solution of (5.2.14) at time t_2 by simulating the overdamped Langevin dynamics (5.2.12) over time interval $[t_1, t_2]$ given a collection of N initial stochastic samples $\{\boldsymbol{X}_{t_1}^i\}_{i=1}^N$ where $\boldsymbol{X}_{t_1}^i \sim p(\cdot, t_1)$. Instead of working with a large number of stochastic samples directly, we deploy the same idea of tensor train sketching to recompress the samples and find a low-complexity MPS/TT parametrization for the solution $p(\cdot, t)$ at any t.

Since each sample can be naturally written as a rank-1 MPS (Dirac delta measure is separable), the tensor diagram for this application is exactly the same as in Fig. 5.2.2. The left-hand side of Fig. 5.2.2a is the PDE solution we want to compress and the right-hand side is a Monte Carlo empirical distribution of the particle system. The only difference is we are sketching a continuous distribution instead of a discrete one in classical Ising model. We refer readers to [86, Appendix C.] for more details about tensor train sketching for continuous distributions.

5.2.6 Complexity Analysis

AFMC/AFQMC offers a significant advantage over traditional Monte Carlo methods in terms of the number of samples required to obtain an accurate approximation. In the classical case, creating the approximation as a sum of N MPS/TT takes $O(dN)$ time, while the recompression process takes $O(rdN)$ time, where r is the highest rank of the sketches. A comprehensive analysis of the complexity of the recompression process can be found in [86, Section 3.2]. The overall complexity is dominated by $O(rdN)$.

In the quantum case, forming the imaginary time evolution as a sum of N MPS/TT takes $O(dr_\Phi N)$ time, where r_Φ is the highest rank of the wavefunction Φ. The subsequent recompression process takes $O(rr_\Phi^2 dN)$ time. It's worth noting that the new wavefunction is obtained through tensor train sketching using random sketches with a maximum rank of r, and therefore $r_\Phi \leq r$. The overall complexity is dominated by $O(r^3 dN)$. Assume we perform imaginary time evolution for N_{iter} iterations. The total time complexity is $O(r^3 dN N_{\text{iter}})$.

5.3 Numerical Experiments

5.3.1 Classical Setting

As we mentioned above, the goal for the classical setting is to find a low-complexity MPS/TT representation of the Boltzmann distribution $\exp(-\beta H)$. We consider two models in this subsection: a $d = 16$ dimensional 1D Ising model and a 2D Ising model of 4×4 lattices with the same Hamiltonian

$$H(x) = -\sum_{\langle ij \rangle} x_i x_j - 0.5 \sum_{i=1}^{d} x_i, \tag{5.3.1}$$

where $\langle ij \rangle$ indicates sites i and j are nearest neighbors. In the 1D case, all the sites are connected as a chain. In the 2D case, we use space filling curve [159] to order the sites. For example, we show the space filling curve and the ordering of the dimensions in Fig. 5.3.1.

We apply several metrics to assess the performance of the proposed algorithm,

- ϵ: relative 2-norm error in $\exp(-\beta H)$,
- ϵ_p: relative error in the log partition function of $\exp(-\beta H)$,
- ϵ_1, ϵ_2, ϵ_3: relative error in $1, 2, 3$-marginals of $\exp(-\beta H)$, as we may be interested in the marginals of the distributions, or based on [86], the marginals can also be used to reconstruct the full distribution.

The rest of the parameters are set as follows: $\beta = 0.01$, we use 10^5 number of samples from AFMC, the random sketching TT is a random TT with a universal $r = 20$ rank and we use 10^{-3} as the singular value threshold to solve the core determining equations. Both of the models have exactly the same hyperparameter settings. We note that all the errors are decaying in a $\sqrt{N}$ rate with the number of parameters and the results are fairly robust across a broad range of choices of random sketches and singular value thresholds. The performance is summarized in Table 5.1.

Models	ϵ	ϵ_p	ϵ_1	ϵ_2	ϵ_3
$d = 16$, 1D Ising	3.9×10^{-3}	3.9×10^{-3}	1.0×10^{-3}	1.3×10^{-3}	1.6×10^{-3}
4×4, 2D Ising	8.0×10^{-3}	8.0×10^{-3}	1.9×10^{-3}	2.1×10^{-3}	2.3×10^{-3}

Table 5.1: Approximation performance in classical setting.

5.3.2 Quantum Setting

In this subsection, we use the transversal-field Ising model of the following quantum Hamiltonian,

$$H = -\sum_{\langle ij \rangle} Z_i Z_j - \sum_i X_i, \tag{5.3.2}$$

where the system undergoes a quantum phase transition. Similar to Section 5.3.1, we use a $d = 16$ 1D transversal-field Ising model and a 4×4 lattices 2D transversal-field Ising model. The same ordering of sites is used for the 2D model as illustrated in Fig. 5.3.1. We set the infinitesimal time step to be 0.01 and use 2000 samples in each power method iteration to approximate the propagator $\exp(-\beta H)$. The rest of the parameters are set as follows: we use a $r = 20$ random TT for sketching and we use 10^{-3} as the singular value threshold to solve the core determining equations. We initialize the test wavefunction as a random TT. The imaginary time evolution energy is shown in Fig. 5.3.2. Here we use the symmetric energy estimator given by

$$E_{\text{symmetric}} = \frac{\langle \phi_t, H, \phi_t \rangle}{\langle \phi_t, \phi_t \rangle}, \tag{5.3.3}$$

where ϕ_t is the wavefunction of the t-th iteration. Theoretically the energy given by the symmetric estimator can only be larger than the ground state energy. Researchers also use mixed estimators,

$$E_{\text{mixed}} = \frac{\langle \phi, H, \phi_t \rangle}{\langle \phi, \phi_t \rangle}, \tag{5.3.4}$$

where ϕ is a fixed wavefunction. The mixed estimator will oscillate around the ground state energy after convergence so one can further take the average of the mixed energy estimators of several iterations to reduce the variance.

We report the energies based on the symmetric estimator in Fig. 5.3.2. For the 1D model, the ground truth energy is 16.5432 and our approach converges to energy 16.6293, with a relative error of 5.2×10^{-3}. In the 2D case, the ground truth energy is 14.5798 and our approach converges to energy 14.5910, with a relative error of 7.7×10^{-4}. Our approach has already achieved stable convergence and very accurate ground state energy estimation with only the symmetric estimators.

Tensor network sketching makes it possible to work with large number of samples while maintaining an algorithm with memory complexity that is independent from the number of samples. We argue that enough number of samples is critical to reduce the stochastic noise and ensure the convergence of the algorithm. As a natural alternative to obtain an algorithm with constant memory usage, we can force to round the MPS/TT to certain given rank once the rank exceeds a prespecified threshold. In comparison, we follow the same idea of approximating the imarginary time evoluation using AFQMC as discussed in Fig. 5.2.3. However, instead of recompressing with sketching, we iteratively add the MPS/TTs together and force round the resulting MPS/TT to $r = 50$ once it exceeds the upper bound $r = 100$. The energy evoluation is shown in Fig. 5.3.3. We show the energy evolutions for both the symmetric and mixed estimators. We can clearly observe that the energy convergence is more stable and even faster with tensor network sketching, due to the denoising effect from large number of samples.

5.3.3 Evolving Fokker-Planck Equations

In this numerical experiment, we consider the following double-well potential

$$V(\boldsymbol{x}) = (x_1^2 - 1)^2 + 0.3\sum_{i=2}^{d} x_i^2, \tag{5.3.5}$$

and the induced particle system governed by the overdamped Langevin process (5.2.12). Since the potential function is easily separable, the equilibrium Boltzmann density is a product of univariate densities for each dimension, i.e.

$$\exp(-\beta V(\boldsymbol{x})) = \exp\left(-\beta(x_1^2 - 1)^2\right) \Pi_{i=2}^{d} \exp\left(-0.3\beta x_i^2\right), \tag{5.3.6}$$

which gives us a nice way to visualize the distribution. In this example, we use $\beta = 1$ and $d = 10$. To obtain a continuous MPS/TT approximation, we use Gaussian kernel function as univariate basis functions for each dimension. We use the same set of 20 basis functions for each dimension and we visualize all the basis functions in Fig. 5.3.4.

We start from the uniform distribution over the hypercube $[-2.5, 2.5]^d$ and evolve the distribution via particle systems. For each time interval $[t_1, t_2]$ and given the MPS/TT representation of the initial density $p(\cdot, t_1)$, we first sample from the initial distribution taking advantage of the efficient sampling algorithms for MPS/TT. Then we simulate the overdamped Langevin process for the sampled particles up to time t_2. Then we recompress these samples back to MPS/TT format to obtain the MPS/TT representation of density $p(\cdot, t_2)$. The same procedure is repeated until convergence. More specifically, we perform a full iteration of the procedure (sampling, evolving and recompression) for every $\Delta t = 0.02$ time. And we use $N = 10^4$ samples for all iterations.

In Fig. 5.3.5 we visualize the simulated Langevin particles, the fitted continuous MPS/TT density and the targeting equilibrium density for the first dimension at iteration $1, 3, 5, 7, 20, 30$, as the convergence gets slower and slower. We can observe that the particle distribution gets evolved effectively by the Langevin dynamics and the fitted continuous MPS/TT density accurately captures the histograms of the particle samples. The low-complexity continuous MPS/TT format also serves as an extra regularization and as a result, is not prone to overfitting. To quantify the performance of our algorithm, we evaluate the relative error metric $E = \|p_1 - \hat{p}_1\|/\|\hat{p}_1\|$ where p_1 and $\hat{p}_1$ are the marginal distribution of the first dimension for the fitted continuous MPS/TT density and the ground truth Boltzmann distribution, respectively. At iteration 30, the relative error $E = 3.8 \times 10^{-2}$. We can further improve the performance of the algorithm by choosing more basis functions and generating more stochastic samples.

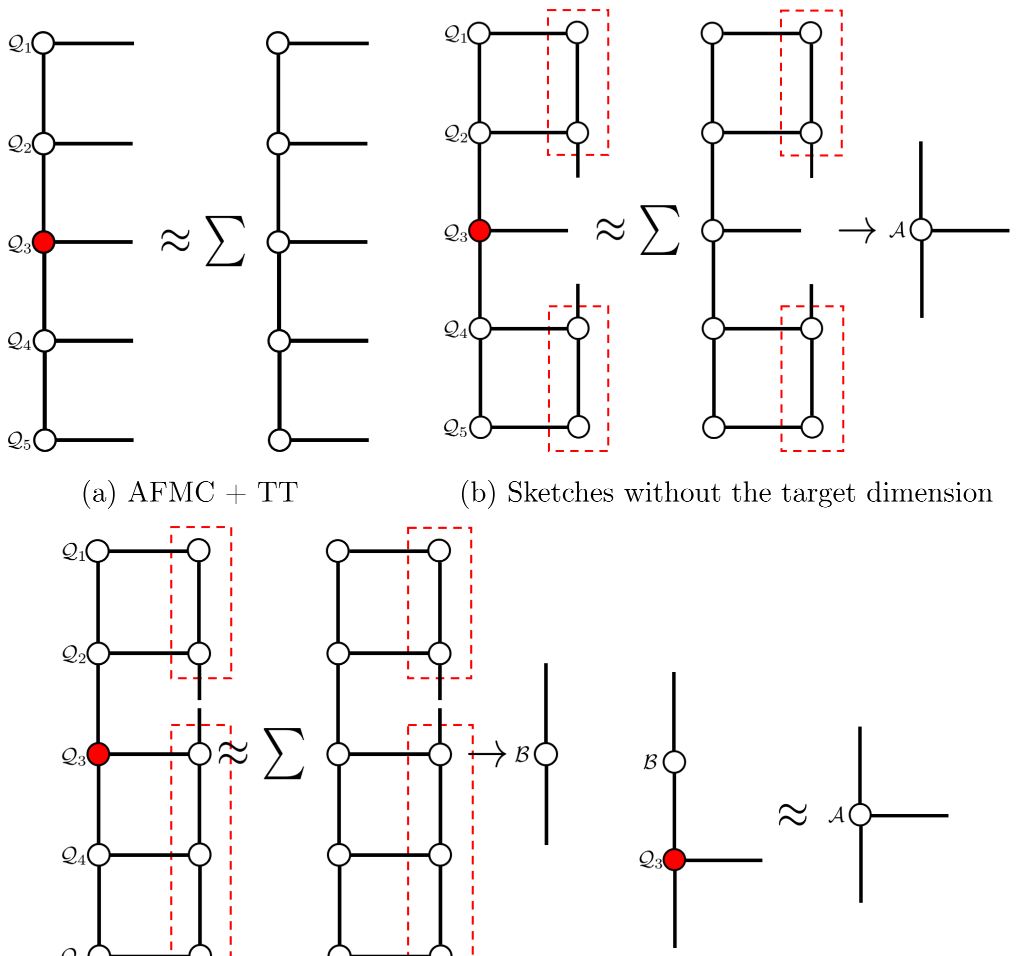
5.4 Discussion

In this chapter, we propose a novel approach to efficiently compress the Boltzmann distribution of a many-body system into an MPS/TT format in the classical setting or an MPO format in the quantum setting. Our starting point is the AFMC/AFQMC method. By leveraging the decoupling effect of the AFMC/AFQMC method, we are able to convert the samples into efficient MPS/MPO formats. To control the rank of the resulting MPS/MPO, we further employ randomized sketching and reconstruct the compressed representation. Through extensive experiments, we demonstrate the numerical accuracy of our approximation and its usefulness in downstream tasks using both classical Ising models and quantum transverse-field Ising models.

Like other Monte Carlo methods, the accuracy of our algorithm also depends crucially on the number of samples used. However, our algorithm still scales linearly with the number of samples, as the rank of the final representation depends solely on the potential field and the system. Moreover, the most time-consuming sampling and sketching procedures of the algorithm can be fully parallelized, further improving its efficiency. With more computational resources, the algorithm's performance can be significantly enhanced.

The performance of our algorithm is determined by two factors: the rank of the randomized sketches and the number of samples used. Our algorithm is expected to succeed when the true Boltzmann distribution can be well-approximated by a low-complexity MPS/MPO format, indicating that the correlations exhibit low-dimensional graphical structures underlying the true distribution. This holds true for many statistical mechanics models of interests. However, if the correlation of the true model is complicated, the effective rank of the representation may increase, potentially even with the number of dimensions. In such cases, more samples are needed to mitigate the sampling error resulting from complicated interactions. Empirical observations indicate an $O(\sqrt{N})$ Monte Carlo convergence rate of the error with respect to the number of samples.

Another way to improve the algorithm's performance is to apply similar techniques used in AFMC/AFQMC applications [220]. For instance, one can first solve the mean-field approximation of the system and then demean the operators based on the mean-field solutions. In the quantum setting, the importance mean of the operator can be computed based on the wavefunction of the current iteration, and importance samples can be drawn from the auxiliary field. While this step increases the computational cost as we may need to sample and recompress the representation for every iteration, it improves the convergence and reduces the number of samples required for an accurate approximation.



(a) AFMC + TT (b) Sketches without the target dimension

(c) Sketches with the target dimension (d) Core determining equations

Figure 5.2.2: Tensor diagram for recompressing AFMC + TT approximations. The unknown tensor core to estimate is marked as filled red node. In practice, the same procedure should be repeated for all dimensions to get all cores $\mathcal{Q}_1, \cdots, \mathcal{Q}_d$. The sketching tensors are highlighted with red dashed boxes.

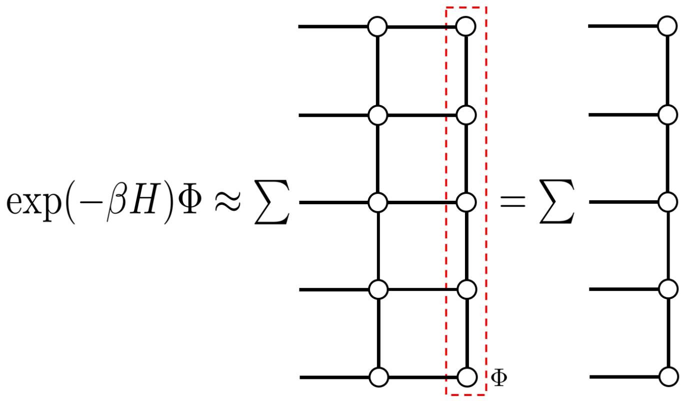


Figure 5.2.3: Tensor diagram for imaginary time evolution in the quantum case. The test wavefunction Φ is parametrized as an MPS/TT and highlighted with red dashed boxes.

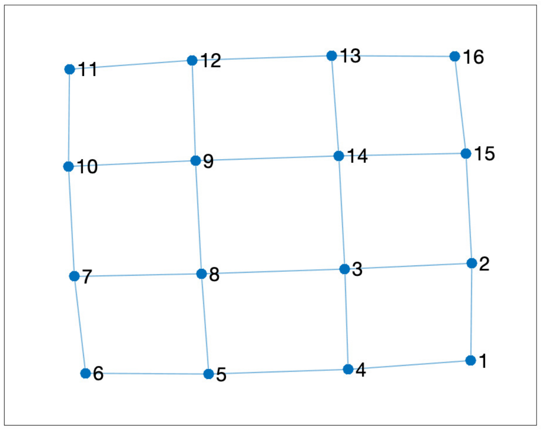
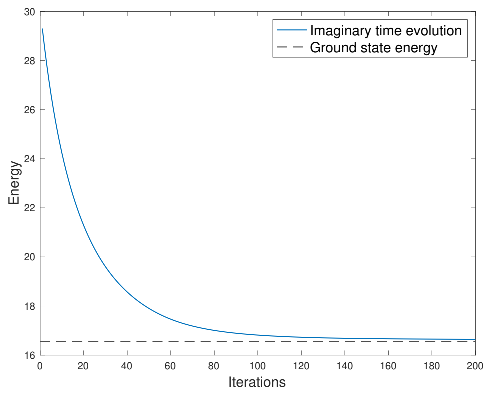
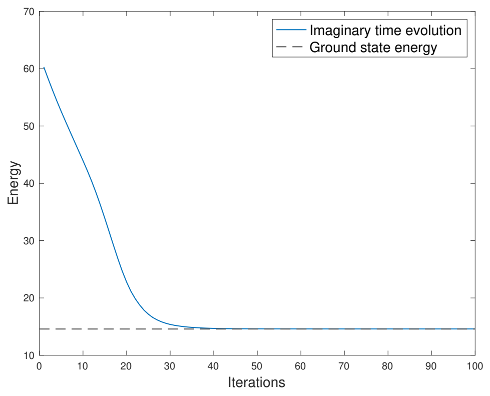


Figure 5.3.1: Example of 2D space filling curve for Ising model of 4×4 lattices.

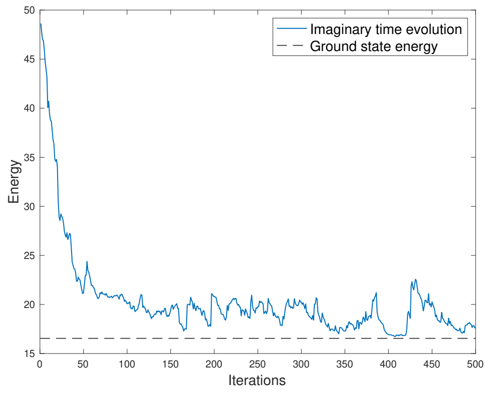


(a) $d = 16$ 1D transversal-field Ising symmetric estimator

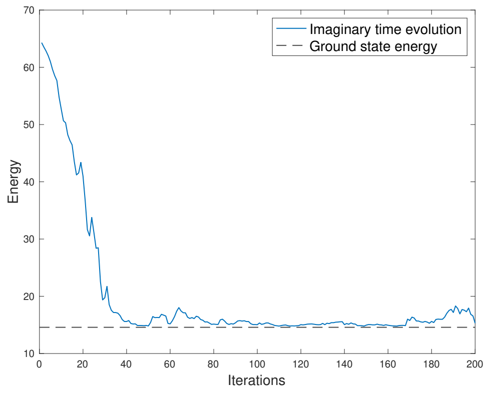


(b) 4×4 2D transversal-field Ising symmetric estimator

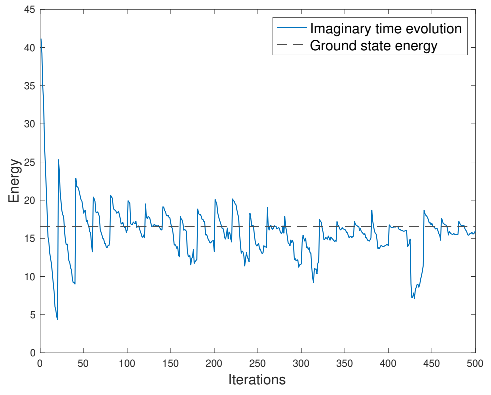
Figure 5.3.2: Imaginary time evolution energy plots.



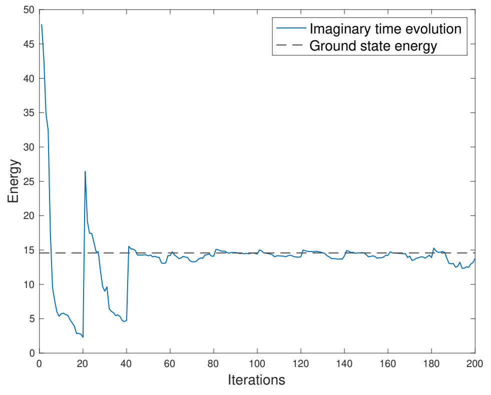
(a) $d = 16$ 1D transversal-field Ising symmetric estimator



(b) 4×4 2D transversal-field Ising symmetric estimator



(c) $d = 16$ 1D transversal-field Ising mixed estimator



(d) 4×4 2D transversal-field Ising mixed estimator

Figure 5.3.3: Imaginary time evolution energy plots when iteratively adding and rounding the MPS/TT to constant rank.

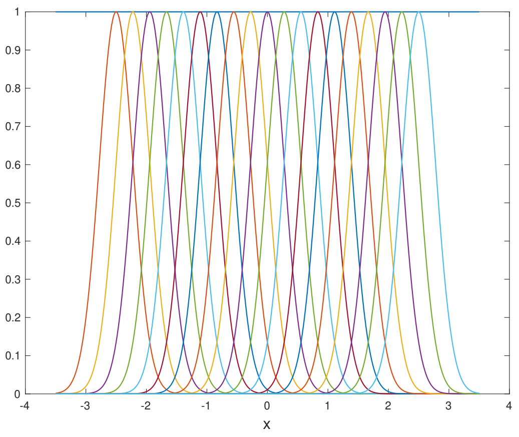
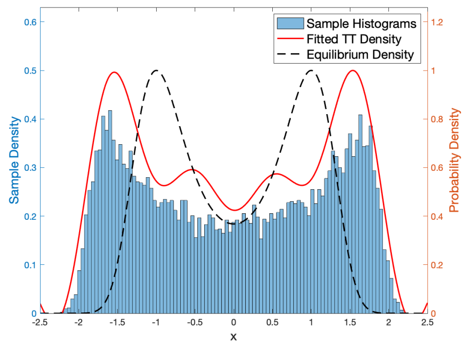
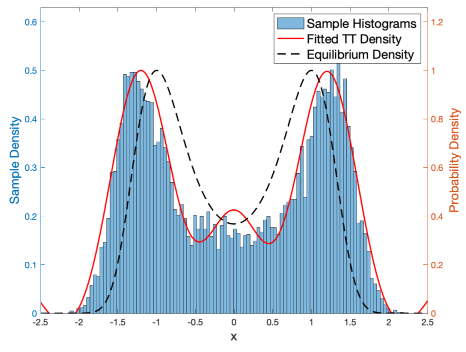


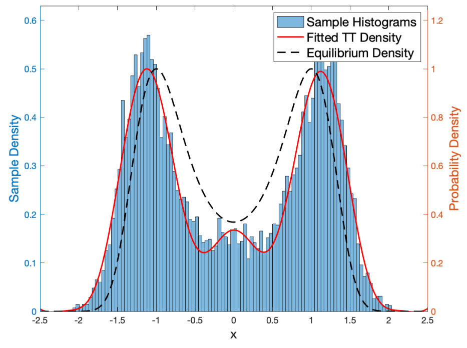
Figure 5.3.4: Visualization of univariate basis functions for each dimension. Here we use Gaussian kernel functions as our basis.



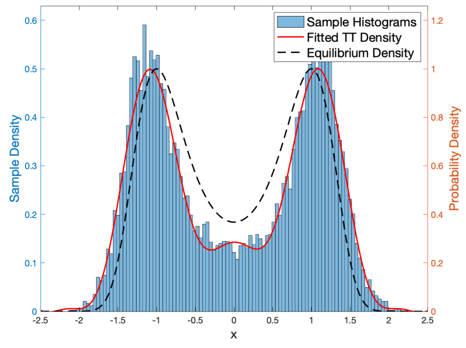
(a) Iteration 1



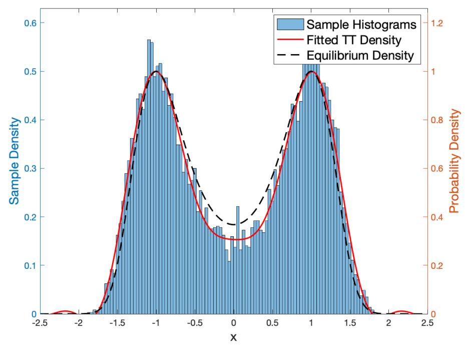
(b) Iteration 3



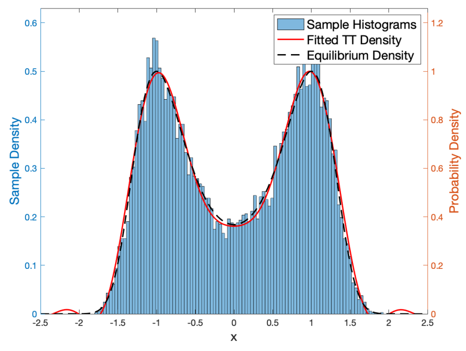
(c) Iteration 5



(d) Iteration 7



(e) Iteration 20



(f) Iteration 30

Figure 5.3.5: Visualization of the density evolution for double-well system in the first dimension. The blue bar plots correspond to the sample histograms after Langevin simulations at each iteration. The fitted continuous TT density and the targeting equilibrium density are represented with red solid lines and black dashed lines, respectively.

REFERENCES

[1] Ian Affleck, Tom Kennedy, Elliott H Lieb, and Hal Tasaki. Valence bond ground states in isotropic quantum antiferromagnets. In *Condensed matter physics and exactly soluble models*, pages 253–304. Springer, 1988.

[2] Alen Alexanderian, Noemi Petra, Georg Stadler, and Omar Ghattas. A-optimal design of experiments for infinite-dimensional Bayesian linear inverse problems with regularized \ell_0-sparsification. *SIAM Journal on Scientific Computing*, 36(5):A2122–A2148, 2014.

[3] Ilona Ambartsumyan, Wajih Boukaram, Tan Bui-Thanh, Omar Ghattas, David Keyes, Georg Stadler, George Turkiyyah, and Stefano Zampini. Hierarchical matrix approximations of Hessians arising in inverse problems governed by PDEs. *arXiv preprint arXiv:2003.10173*, 2020.

[4] Sivaram Ambikasaran and Eric Darve. An $o(n \log N)$ fast direct solver for partial hierarchically semi-separable matrices. *Journal of Scientific Computing*, 57(3):477–501, 2013.

[5] Sivaram Ambikasaran, Daniel Foreman-Mackey, Leslie Greengard, David W Hogg, and Michael O'Neil. Fast direct methods for Gaussian processes. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):252–265, 2015.

[6] Sivaram Ambikasaran, Michael O'Neil, and Karan Raj Singh. Fast symmetric factorization of hierarchical matrices with applications. *arXiv preprint arXiv:1405.0223*, 2014.

[7] Mihai Anitescu, Jie Chen, and Michael L Stein. An inversion-free estimating equations approach for gaussian process models. *Journal of Computational and Graphical Statistics*, 26(1):98–107, 2017.

[8] Mihai Anitescu, Jie Chen, and Lei Wang. A matrix-free approach for solving the parametric Gaussian process maximum likelihood problem. *SIAM Journal on Scientific Computing*, 34(1):A240–A262, 2012.

[9] Amit Apte, Didier Auroux, and R Mythily. Variational data assimilation for discrete Burgers equation. In *Electronic Journal of Differential Equations Conference*, pages 15–30. Texas State Univ., 2010.

[10] Haim Avron and Sivan Toledo. Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix. *Journal of the ACM (JACM)*, 58(2):1–34, 2011.

[11] Markus Bachmayr, Reinhold Schneider, and André Uschmajew. Tensor networks and hierarchical tensors for the solution of high-dimensional partial differential equations. *Foundations of Computational Mathematics*, 16(6):1423–1472, 2016.

[12] SA Baeurle. Computation within the auxiliary field approach. *Journal of Computational Physics*, 184(2):540–558, 2003.

[13] SA Baeurle. Grand canonical auxiliary field monte carlo: a new technique for simulating open systems at high density. *Computer physics communications*, 157(3):201–206, 2004.

[14] Stephan A Baeurle, Roman Martoňák, and Michele Parrinello. A field-theoretical approach to simulation in the classical canonical and grand canonical ensemble. *The Journal of chemical physics*, 117(7):3027–3039, 2002.

[15] Alberto Baiardi and Markus Reiher. The density matrix renormalization group in chemistry and molecular physics: Recent developments and new challenges. *The Journal of Chemical Physics*, 152(4):040903, 2020.

[16] Sudipto Banerjee, Alan E Gelfand, Andrew O Finley, and Huiyan Sang. Gaussian predictive process models for large spatial data sets. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(4):825–848, 2008.

[17] Mário Basto, Viriato Semiao, and Francisco Calheiros. Dynamics and synchronization of numerical solutions of the Burgers equation. *Journal of computational and applied mathematics*, 231(2):793–806, 2009.

[18] Mario Bebendorf and Sergej Rjasanow. Adaptive low-rank approximation of collocation matrices. *Computing*, 70:1–24, 2003.

[19] Nicolas Béreux, Aurélien Decelle, Cyril Furtlehner, and Beatriz Seoane. Learning a restricted boltzmann machine using biased monte carlo sampling. *arXiv preprint arXiv:2206.01310*, 2022.

[20] L Mark Berliner. Hierarchical Bayesian time series models. In *Maximum entropy and Bayesian methods*, pages 15–22. Springer, 1996.

[21] L Mark Berliner. Physical-statistical modeling in geophysics. *Journal of Geophysical Research: Atmospheres*, 108(D24), 2003.

[22] L. Mark Berliner, Ralph F. Milliff, and Christopher K. Wikle. Bayesian hierarchical modeling of air-sea interaction. *Journal of Geophysical Research (Oceans)*, 108(C4):3104, April 2003.

[23] L Mark Berliner, J Andrew Royle, Christopher K Wikle, and Ralph F Milliff. Bayesian methods in the atmospheric sciences. *Bayesian statistics*, 6:83–100, 1999.

[24] Anna Berteotti, Andrea Cavalli, Davide Branduardi, Francesco Luigi Gervasio, Maurizio Recanatini, and Michele Parrinello. Protein conformational transitions: the closure mechanism of a kinase explored by atomistic simulations. *Journal of the American Chemical Society*, 131(1):244–250, 2009.

[25] Rabi N Bhattacharya and Edward C Waymire. *Stochastic processes with applications.* SIAM, 2009.

[26] Daniele Bigoni, Allan P Engsig-Karup, and Youssef M Marzouk. Spectral tensor-train decomposition. *SIAM Journal on Scientific Computing*, 38(4):A2405–A2439, 2016.

[27] Pratik Biswas, T-C Liang, K-C Toh, Yinyu Ye, and T-C Wang. Semidefinite programming approaches for sensor network localization with noisy distance measurements. *IEEE transactions on automation science and engineering*, 3(4):360–371, 2006.

[28] Richard Blankenbecler, DJ Scalapino, and RL Sugar. Monte carlo calculations of coupled boson-fermion systems. i. *Physical Review D*, 24(8):2278, 1981.

[29] Avrim Blum, John Hopcroft, and Ravindran Kannan. Foundations of data science. *Vorabversion eines Lehrbuchs*, 5, 2016.

[30] David Bolin and Kristin Kirchner. The rational SPDE approach for Gaussian random fields with general smoothness. *Journal of Computational and Graphical Statistics*, 29(2):274–285, 2020.

[31] Steffen Börm. Data-sparse approximation of non-local operators by $\mathcal{H}^2$-matrices. *Linear algebra and its applications*, 422(2-3):380–403, 2007.

[32] Steffen Börm. Directional-matrix compression for high-frequency problems. *Numerical Linear Algebra with Applications*, 24(6):e2112, 2017.

[33] Steffen Börm and Jochen Garcke. Approximating Gaussian processes with $\mathcal{H}^2$-matrices. In *European Conference on Machine Learning*, pages 42–53. Springer, 2007.

[34] Wajih Boukaram, George Turkiyyah, and David Keyes. Randomized GPU algorithms for the construction of hierarchical matrices from matrix-vector operations. *SIAM Journal on Scientific Computing*, 41(4):C339–C366, 2019.

[35] Phillip Boyle and Marcus Frean. Dependent Gaussian processes. *Advances in neural information processing systems*, 17:217–224, 2004.

[36] Mike Brookes. The matrix reference manual. *Imperial College London*, 3, 2005.

[37] Charles G Broyden. A class of methods for solving nonlinear simultaneous equations. *Mathematics of computation*, 19(92):577–593, 1965.

[38] Tan Bui-Thanh, Omar Ghattas, James Martin, and Georg Stadler. A computational framework for infinite-dimensional Bayesian inverse problems part i: The linearized case, with application to global seismic inversion. *SIAM Journal on Scientific Computing*, 35(6):A2494–A2523, 2013.

[39] Tony Cai, Zongming Ma, and Yihong Wu. Optimal estimation and rank detection for sparse spiked covariance matrices. *Probability theory and related fields*, 161(3):781–815, 2015.

[40] Jian Cao, Marc G Genton, David E Keyes, and George M Turkiyyah. Hierarchical-block conditioning approximations for high-dimensional multivariate normal probabilities. *Statistics and Computing*, 29(3):585–598, 2019.

[41] John Cardy. *Scaling and renormalization in statistical physics*, volume 5. Cambridge university press, 1996.

[42] J Carlson, Stefano Gandolfi, Kevin E Schmidt, and Shiwei Zhang. Auxiliary-field quantum monte carlo method for strongly paired fermions. *Physical Review A*, 84(6):061602, 2011.

[43] George Casella. An introduction to empirical Bayes data analysis. *The American Statistician*, 39(2):83–87, 1985.

[44] Julio E Castrillon-Candás, Marc G Genton, and Rio Yokota. Multi-level restricted maximum likelihood covariance estimation and kriging for large non-gridded spatial datasets. *Spatial Statistics*, 18:105–124, 2016.

[45] David Ceperley, Geoffrey V Chester, and Malvin H Kalos. Monte carlo simulation of a many-fermion study. *Physical Review B*, 16(7):3081, 1977.

[46] Xiao-Wen Chang, Christopher C Paige, and GW Stewart. Perturbation analyses for the QR factorization. *SIAM Journal on Matrix Analysis and Applications*, 18(3):775–791, 1997.

[47] Jie Chen and Michael L Stein. Linear-cost covariance functions for Gaussian random fields. *arXiv preprint arXiv:1711.05895*, 2017.

[48] Nan Chen and Andrew J Majda. Filtering the stochastic skeleton model for the Madden–Julian oscillation. *Monthly Weather Review*, 144(2):501–527, 2016.

[49] Yian Chen and Mihai Anitescu. Scalable Gaussian process analysis for implicit physics-based covariance models. *International Journal for Uncertainty Quantification*, 11(6), 2021.

[50] Yian Chen, Jeremy Hoskins, Yuehaw Khoo, and Michael Lindsey. Committor functions via tensor networks. *Journal of Computational Physics*, 472:111646, 2023.

[51] Hongwei Cheng, Zydrunas Gimbutas, Per-Gunnar Martinsson, and Vladimir Rokhlin. On the compression of low rank matrices. *SIAM Journal on Scientific Computing*, 26(4):1389–1404, 2005.

[52] D Yu Chenhan, Severin Reiz, and George Biros. Distributed-memory hierarchical compression of dense SPD matrices. In *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 183–197. IEEE, 2018.

[53] Jean-Paul Chilès. How to adapt kriging to non-classical problems: three case studies. In *Advanced geostatistics in the mining industry*, pages 69–89. Springer, 1976.

[54] Jean-Paul Chiles and Pierre Delfiner. *Geostatistics: modeling spatial uncertainty*, volume 497. John Wiley & Sons, 2009.

[55] James S Clark and Alan E Gelfand. *Hierarchical modelling for the environmental sciences: statistical methods and applications*. OUP Oxford, 2006.

[56] Ronald R Coifman, Ioannis G Kevrekidis, Stéphane Lafon, Mauro Maggioni, and Boaz Nadler. Diffusion maps, reduction coordinates, and low dimensional representation of stochastic systems. *Multiscale Modeling & Simulation*, 7(2):842–864, 2008.

[57] Emil M Constantinescu and Mihai Anitescu. Physics-based covariance models for Gaussian processes with multiple outputs. *International Journal for Uncertainty Quantification*, 3(1), 2013.

[58] Noel Cressie. *Statistics for spatial data*. John Wiley & Sons, 2015.

[59] Noel Cressie and Gardar Johannesson. Fixed rank kriging for very large spatial data sets. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(1):209–226, 2008.

[60] Martin J Crowder. Maximum likelihood estimation for dependent observations. *Journal of the Royal Statistical Society: Series B (Methodological)*, 38(1):45–53, 1976.

[61] David Elieser Deutsch. Quantum computational networks. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 425(1868):73–90, 1989.

[62] Jürgen Dölz, Helmut Harbrecht, and Michael D Multerer. On the best approximation of the hierarchical matrix product. *SIAM Journal on Matrix Analysis and Applications*, 40(1):147–174, 2019.

[63] Iain S Duff, Albert Maurice Erisman, and John Ker Reid. *Direct methods for sparse matrices*. Oxford University Press, 2017.

[64] Glen Evenbly and Guifré Vidal. Tensor network states and geometry. *Journal of Statistical Physics*, 145(4):891–918, 2011.

[65] Matthew Fishman, Steven R. White, and E. Miles Stoudenmire. The ITensor software library for tensor network calculations, 2020.

[66] Roger Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2013.

[67] Geir-Arne Fuglstad, Daniel Simpson, Finn Lindgren, and Håvard Rue. Does non-stationary spatial data always require non-stationary random fields? *Spatial Statistics*, 14:505–531, 2015.

[68] Reinhard Furrer, Marc G Genton, and Douglas Nychka. Covariance tapering for interpolation of large spatial datasets. *Journal of Computational and Graphical Statistics*, 15(3):502–523, 2006.

[69] Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. CRC press, 2013.

[70] Marc G Genton, David E Keyes, and George Turkiyyah. Hierarchical decompositions for the computation of high-dimensional multivariate normal probabilities. *Journal of Computational and Graphical Statistics*, 27(2):268–277, 2018.

[71] Christopher J Geoga, Mihai Anitescu, and Michael L Stein. Scalable Gaussian process computations using hierarchical matrices. *Journal of Computational and Graphical Statistics*, 29(2):227–237, 2020.

[72] Nigel Goldenfeld. *Lectures on phase transitions and the renormalization group*. CRC Press, 2018.

[73] Alex Gorodetsky, Sertac Karaman, and Youssef Marzouk. A continuous analogue of the tensor-train decomposition. *Computer Methods in Applied Mechanics and Engineering*, 347:59–84, 2019.

[74] Barry J Grant, Alemayehu A Gorfe, and J Andrew McCammon. Large conformational changes in proteins: signaling and other functions. *Current opinion in structural biology*, 20(2):142–147, 2010.

[75] Lars Grasedyck. Hierarchical low rank approximation of tensors and multivariate functions. *Lecture notes of the Zürich summer school on Sparse Tensor Discretizations of High-Dimensional Problems*, 2010.

[76] Lars Grasedyck, Daniel Kressner, and Christine Tobler. A literature survey of low-rank tensor approximation techniques. *GAMM-Mitteilungen*, 36(1):53–78, 2013.

[77] Wolfgang Hackbusch. *Hierarchical matrices: algorithms and analysis*, volume 49. Springer, 2015.

[78] Wolfgang Hackbusch, Boris N Khoromskij, and Eugene E Tyrtyshnikov. Hierarchical kronecker tensor-product approximations. 2005.

[79] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[80] Jiequn Han, Arnulf Jentzen, and E Weinan. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

[81] Jouni Hartikainen and Simo Särkkä. Kalman filtering and smoothing solutions to temporal Gaussian process regression models. In *2010 IEEE international workshop on machine learning for signal processing*, pages 379–384. IEEE, 2010.

[82] Rüdiger Hewer, Petra Friederichs, Andreas Hense, and Martin Schlather. A Matérn-based multivariate Gaussian random process for a consistent model of the horizontal wind components and related variables. *Journal of the Atmospheric Sciences*, 74(11):3833–3845, 2017.

[83] Frank L Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189, 1927.

[84] K-H Hoffmann and Qi Tang. *Ginzburg-Landau phase transition theory and superconductivity*, volume 134. Birkhäuser, 2012.

[85] John Hubbard. Calculation of partition functions. *Physical Review Letters*, 3(2):77, 1959.

[86] Yoonhaeng Hur, Jeremy G Hoskins, Michael Lindsey, E Miles Stoudenmire, and Yuehaw Khoo. Generative modeling via tensor train sketching. *arXiv preprint arXiv:2202.11788*, 2022.

[87] Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 19(2):433–450, 1990.

[88] Katharine Hyatt and E Miles Stoudenmire. Dmrg approach to optimizing two-dimensional tensor networks. *arXiv preprint arXiv:1908.08833*, 2019.

[89] Leo P Kadanoff. More is the same; phase transitions and mean field theories. *Journal of Statistical Physics*, 137:777–797, 2009.

[90] Eugenia Kalnay. *Atmospheric modeling, data assimilation and predictability*. Cambridge university press, 2003.

[91] Eugenia Kalnay, Masao Kanamitsu, Robert Kistler, William Collins, Dennis Deaven, Lev Gandin, Mark Iredell, Suranjana Saha, Glenn White, John Woollen, et al. The NCEP/NCAR 40-year reanalysis project. *Bulletin of the American meteorological Society*, 77(3):437–472, 1996.

[92] Mehran Kardar. *Statistical physics of fields*. Cambridge University Press, 2007.

[93] Cari G Kaufman, Mark J Schervish, and Douglas W Nychka. Covariance tapering for likelihood-based estimation in large spatial data sets. *Journal of the American Statistical Association*, 103(484):1545–1555, 2008.

[94] Yuehaw Khoo, Jianfeng Lu, and Lexing Ying. Solving parametric PDE problems with artificial neural networks. *arXiv preprint arXiv:1707.03351*, 2017.

[95] Yuehaw Khoo, Jianfeng Lu, and Lexing Ying. Solving for high-dimensional committor functions using artificial neural networks. *Research in the Mathematical Sciences*, 6(1):1–13, 2019.

[96] Boris N Khoromskij. Tensor numerical methods for multidimensional pdes: theoretical analysis and initial applications. *ESAIM: Proceedings and Surveys*, 48:1–28, 2015.

[97] Boris N. Khoromskij and Christoph Schwab. Tensor-structured galerkin approximation of parametric and stochastic elliptic pdes. *SIAM Journal on Scientific Computing*, 33(1):364–385, 2011.

[98] Boris N Khoromskij and Christoph Schwab. Tensor-structured galerkin approximation of parametric and stochastic elliptic pdes. *SIAM Journal on Scientific Computing*, 33(1):364–385, 2011.

[99] Charles Kittel and Herbert Kroemer. Thermal physics, 1998.

[100] Peter E Kloeden and Eckhard Platen. *Numerical solution of stochastic differential equations*, volume 23. Springer Science & Business Media, 2013.

[101] Juš Kocijan, Roderick Murray-Smith, Carl Edward Rasmussen, and Agathe Girard. Gaussian process model based predictive control. In *Proceedings of the 2004 American control conference*, volume 3, pages 2214–2219. IEEE, 2004.

[102] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[103] Dilip Kondepudi and Ilya Prigogine. *Modern thermodynamics: from heat engines to dissipative structures.* John Wiley & Sons, 2014.

[104] Daniel Kressner and Ana Susnjara. Fast QR decomposition of HODLR matrices. *arXiv preprint arXiv:1809.10585*, 2018.

[105] Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. Ensemble Nystrom method. In *Advances in Neural Information Processing Systems*, pages 1060–1068, 2009.

[106] Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5):987–1000, 1998.

[107] Rongjie Lai and Jianfeng Lu. Point cloud discretization of fokker–planck operators for committor functions. *Multiscale Modeling & Simulation*, 16(2):710–726, 2018.

[108] David Landau and Kurt Binder. *A guide to Monte Carlo simulations in statistical physics.* Cambridge university press, 2021.

[109] Antonio C Lasaga. 2. transition state theory. In *Kinetic Theory in the Earth Sciences*, pages 152–219. Princeton University Press, 2014.

[110] Joonho Lee, Hung Q Pham, and David R Reichman. Twenty years of auxiliary-field quantum monte carlo in quantum chemistry: An overview and assessment on main group chemistry and bond-breaking. *Journal of Chemical Theory and Computation*, 18(12):7024–7042, 2022.

[111] Tine Lefebvre*, Herman Bruyninckx, and Joris De Schutter. Kalman filters for non-linear systems: a comparison of performance. *International journal of Control*, 77(7):639–653, 2004.

[112] John M Lewis, Sivaramakrishnan Lakshmivarahan, and Sudarshan Dhall. *Dynamic data assimilation: a least squares approach*, volume 13. Cambridge University Press, 2006.

[113] Haoya Li, Yuehaw Khoo, Yinuo Ren, and Lexing Ying. A semigroup method for high dimensional committor functions based on neural network. *arXiv preprint arXiv:2012.06727*, 2020.

[114] Haoya Li, Yuehaw Khoo, Yinuo Ren, and Lexing Ying. Solving for high dimensional committor functions using neural network with online approximation to derivatives. *arXiv preprint arXiv:2012.06727*, 2020.

[115] Qianxiao Li, Bo Lin, and Weiqing Ren. Computing committor functions for the study of rare events using deep learning. *The Journal of Chemical Physics*, 151:054112, 2019.

[116] Edo Liberty, Franco Woolfe, Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert. Randomized algorithms for the low-rank approximation of matrices. *Proceedings of the National Academy of Sciences*, 104(51):20167–20172, 2007.

[117] Brant Liebmann and Catherine A Smith. Description of a complete (interpolated) outgoing longwave radiation dataset. *Bulletin of the American Meteorological Society*, 77(6):1275–1277, 1996.

[118] EM Lifshitz and Lev Davidovich Landau. Statistical physics (course of theoretical physics, volume 5), 1984.

[119] Lin Lin, Jianfeng Lu, and Lexing Ying. Fast construction of hierarchical matrix representation from matrix–vector multiplication. *Journal of Computational Physics*, 230(10):4071–4087, 2011.

[120] Finn Lindgren, Håvard Rue, and Johan Lindström. An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(4):423–498, 2011.

[121] G. Little and J.B. Reade. Eigenvalues of analytic kernels. *SIAM J. Math. Anal.*, 15(1):133–136, 1984.

[122] Alexander Litvinenko, Ying Sun, Marc G Genton, and David E Keyes. Likelihood approximation with hierarchical matrices for large spatial datasets. *Computational Statistics & Data Analysis*, 137:115–132, 2019.

[123] Jianfeng Lu and James Nolen. Reactive trajectories and the transition path process. *Probability Theory and Related Fields*, 161(1):195–244, 2015.

[124] Jianfeng Lu and Eric Vanden-Eijnden. Exact dynamical coarse-graining without time-scale separation. *The Journal of chemical physics*, 141(4):07B619_1, 2014.

[125] David JC MacKay. Introduction to Gaussian processes. *NATO ASI Series F Computer and Systems Sciences*, 168:133–166, 1998.

[126] Azamat Mametjanov, Boyana Norris, Xiaoyan Zeng, Beth Drewniak, Jean Utke, Mihai Anitescu, and Paul Hovland. Applying automatic differentiation to the Community Land Model. In *Recent Advances in Algorithmic Differentiation*, pages 47–57. Springer, 2012.

[127] Luca Maragliano, Alexander Fischer, Eric Vanden-Eijnden, and Giovanni Ciccotti. String method in collective variables: Minimum free energy paths and isocommittor surfaces. *The Journal of chemical physics*, 125(2):024106, 2006.

[128] Per-Gunnar Martinsson. A fast randomized algorithm for computing a hierarchically semiseparable representation of a matrix. *SIAM Journal on Matrix Analysis and Applications*, 32(4):1251–1274, 2011.

[129] Per-Gunnar Martinsson. Compressing rank-structured matrices via randomized sampling. *SIAM Journal on Scientific Computing*, 38(4):A1959–A1986, 2016.

[130] Stefano Massei, Leonardo Robol, and Daniel Kressner. hm-toolbox: MATLAB software for HODLR and HSS matrices. *SIAM Journal on Scientific Computing*, 42(2):C43–C68, 2020.

[131] Victor Minden, Anil Damle, Kenneth L Ho, and Lexing Ying. Fast spatial Gaussian process maximum likelihood estimation via skeletonization factorizations. *Multiscale Modeling & Simulation*, 15(4):1584–1611, 2017.

[132] Mohammad Amin Nabian and Hadi Meidani. A deep neural network surrogate for high-dimensional random partial differential equations. *arXiv preprint arXiv:1806.02957*, 2018.

[133] Richard D Neidinger. Introduction to automatic differentiation and MATLAB object-oriented programming. *SIAM review*, 52(3):545–563, 2010.

[134] Duy Nguyen-Tuong, Jan R Peters, and Matthias Seeger. Local Gaussian process regression for real time online model learning. In *Advances in Neural Information Processing Systems*, pages 1193–1200, 2009.

[135] Wolfgang Nowak and Alexander Litvinenko. Kriging and spatial design accelerated by orders of magnitude: Combining low-rank covariance approximations with FFT-techniques. *Mathematical Geosciences*, 45(4):411–435, 2013.

[136] H Reed Ogrosky and Samuel N Stechmann. Identifying convectively coupled equatorial waves using theoretical wave eigenvectors. *Monthly Weather Review*, 144(6):2235–2264, 2016.

[137] Naoto Okuyama-Yoshida, Masataka Nagaoka, and Tokio Yamabe. Transition-state optimization on free energy surface: Toward solution chemical reaction ergodography. *International journal of quantum chemistry*, 70(1):95–103, 1998.

[138] Román Orús. Advances on tensor network theory: symmetries, fermions, entanglement, and holography. *The European Physical Journal B*, 87(11):1–18, 2014.

[139] Román Orús. A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Annals of Physics*, 349:117–158, 2014.

[140] Ivan Oseledets and Eugene Tyrtyshnikov. TT-cross approximation for multidimensional arrays. *Linear Algebra and its Applications*, 432(1):70–88, 2010.

[141] Ivan V Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

[142] Raj Kumar Pathria. *Statistical mechanics*. Elsevier, 2016.

[143] David Perez-Garcia, Frank Verstraete, Michael M Wolf, and J Ignacio Cirac. Matrix product state representations. *arXiv preprint quant-ph/0608197*, 2006.

[144] Noemi Petra, Cosmin G Petra, Zheng Zhang, Emil M Constantinescu, and Mihai Anitescu. A Bayesian approach for parameter estimation with uncertainty for dynamic power systems. *IEEE Transactions on Power Systems*, 32(4):2735–2743, 2016.

[145] Mingpu Qin, Hao Shi, and Shiwei Zhang. Benchmark study of the two-dimensional hubbard model with auxiliary-field quantum monte carlo method. *Physical Review B*, 94(8):085103, 2016.

[146] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6(Dec):1939–1959, 2005.

[147] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[148] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

[149] Konrad Reif, Stefan Gunther, Engin Yaz, and Rolf Unbehauen. Stochastic stability of the discrete-time extended Kalman filter. *IEEE Transactions on Automatic control*, 44(4):714–728, 1999.

[150] Weiqing Ren, Eric Vanden-Eijnden, Paul Maragakis, and Weinan E. Transition pathways in complex systems: Application of the finite-temperature string method to the alanine dipeptide. *The Journal of chemical physics*, 123(13):134109, 2005.

[151] Yinuo Ren, Hongli Zhao, Yuehaw Khoo, and Lexing Ying. High-dimensional density estimation with tensorizing flow. *arXiv preprint arXiv:2212.00759*, 2022.

[152] Lorenz Richter, Leon Sallandt, and Nikolas Nüsken. Solving high-dimensional parabolic pdes using the tensor train format. In *International Conference on Machine Learning*, pages 8998–9009. PMLR, 2021.

[153] Vladimir Rokhlin. Rapid solution of integral equations of classical potential theory. *Journal of computational physics*, 60(2):187–207, 1985.

[154] Grant M Rotskoff, Andrew R Mitchell, and Eric Vanden-Eijnden. Active importance sampling for variational objectives dominated by rare events: Consequences for optimization and generalization. *arXiv preprint arXiv:2008.06334*, 2020.

[155] Grant M Rotskoff and Eric Vanden-Eijnden. Learning with rare data: using active importance sampling to optimize objectives dominated by rare events. *Preprint at arXiv https://arxiv. org/abs/2008.06334*, 2020.

[156] JA Royle, LM Berliner, CK Wikle, and R Milliff. A hierarchical spatial model for constructing wind fields from scatterometer data in the labrador sea. In *Case Studies in Bayesian Statistics*, pages 367–382. Springer, 1999.

[157] Hååvard Rue and Hååkon Tjelmeland. Fitting Gaussian Markov random fields to Gaussian fields. *Scandinavian journal of Statistics*, 29(1):31–49, 2002.

[158] Havard Rue and Leonhard Held. *Gaussian Markov random fields: theory and applications*. CRC press, 2005.

[159] Hans Sagan. *Space-filling curves*. Springer Science & Business Media, 2012.

[160] Arvind K Saibaba and Peter K Kitanidis. Efficient methods for large-scale linear inversion using a geostatistical approach. *Water Resources Research*, 48(5), 2012.

[161] Ruslan Salakhutdinov and Hugo Larochelle. Efficient learning of deep boltzmann machines. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 693–700. JMLR Workshop and Conference Proceedings, 2010.

[162] Simo Sarkka and Arno Solin. *Applied stochastic differential equations*, volume 10. Cambridge University Press, 2019.

[163] Simo Sarkka, Arno Solin, and Jouni Hartikainen. Spatiotemporal learning via infinite-dimensional Bayesian filtering and smoothing: A look at Gaussian process regression through Kalman filtering. *IEEE Signal Processing Magazine*, 30(4):51–61, 2013.

[164] Dmitry V Savostyanov, SV Dolgov, JM Werner, and Ilya Kuprov. Exact nmr simulation of protein-size spin systems using tensor train formalism. *Physical Review B*, 90(8):085139, 2014.

[165] Martin Schlather, Alexander Malinowski, Peter J Menck, Marco Oesting, and Kirstin Strokorb. Analysis, simulation and prediction of multivariate random fields with package random fields. *Journal of Statistical Software*, 63(8):1–25, 2015.

[166] James P Sethna. *Statistical mechanics: entropy, order parameters, and complexity*, volume 14. Oxford University Press, USA, 2021.

[167] Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[168] Hao Shi and Shiwei Zhang. Some recent developments in auxiliary-field quantum monte carlo for real materials. *The Journal of Chemical Physics*, 154(2):024107, 2021.

[169] Yuta Shingu, Yuya Seki, Shohei Watabe, Suguru Endo, Yuichiro Matsuzaki, Shiro Kawabata, Tetsuro Nikuni, and Hideaki Hakoshima. Boltzmann machine learning with a variational quantum algorithm. *Physical Review A*, 104(3):032413, 2021.

[170] Si Si, Cho-Jui Hsieh, and Inderjit S Dhillon. Memory efficient kernel approximation. *The Journal of Machine Learning Research*, 18(1):682–713, 2017.

[171] Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018.

[172] Nitish Srivastava and Russ R Salakhutdinov. Multimodal learning with deep boltzmann machines. *Advances in neural information processing systems*, 25, 2012.

[173] Samuel N Stechmann and Scott Hottovy. Unified spectrum of tropical rainfall and waves in a simple stochastic model. *Geophysical Research Letters*, 44(20):10–713, 2017.

[174] Samuel N Stechmann and Andrew J Majda. Identifying the skeleton of the Madden–Julian oscillation in observational data. *Monthly Weather Review*, 143(1):395–416, 2015.

[175] Samuel N Stechmann and H Reed Ogrosky. The Walker circulation, diabatic heating, and outgoing longwave radiation. *Geophysical Research Letters*, 41(24):9097–9105, 2014.

[176] Michael L Stein. *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media, 2012.

[177] Michael L Stein, Jie Chen, Mihai Anitescu, et al. Stochastic approximation of score functions for Gaussian processes. *The Annals of Applied Statistics*, 7(2):1162–1191, 2013.

[178] Defeng Sun, Kim-Chuan Toh, Yancheng Yuan, and Xin-Yuan Zhao. SDPNAL+: A Matlab software for semidefinite programming with bound constraints (version 1.0). *Optimization Methods and Software*, 35(1):87–115, 2020.

[179] Ludovico Sutto and Francesco Luigi Gervasio. Effects of oncogenic mutations on the conformational free-energy landscape of egfr kinase. *Proceedings of the National Academy of Sciences*, 110(26):10616–10621, 2013.

[180] James Joseph Sylvester. Xxxvii. on the relation between the minor determinants of linearly equivalent quadratic functions. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 1(4):295–305, 1851.

[181] Toshiyuki Tanaka. A theory of mean field approximation. *Advances in Neural Information Processing Systems*, 11, 1998.

[182] Xun Tang, Yoonhaeng Hur, Yuehaw Khoo, and Lexing Ying. Generative modeling via tree tensor network states. *arXiv preprint arXiv:2209.01341*, 2022.

[183] Reginald P Tewarson and Reginald P Tewarson. *Sparse matrices*, volume 69. Academic press New York, 1973.

[184] Erik H. Thiede, Dimitrios Giannakis, Aaron R. Dinner, and Jonathan Weare. Galerkin approximation of dynamical quantities using trajectory data. *The Journal of Chemical Physics*, 150:244111, 2019.

[185] L. N. Trefethen. *Approximation Theory and Approximation Practice*. SIAM, 2013.

[186] Lloyd N Trefethen. *Approximation Theory and Approximation Practice, Extended Edition*. SIAM, 2019.

[187] Volker Tresp. A bayesian committee machine. *Neural computation*, 12(11):2719–2741, 2000.

[188] Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[189] Francesco Uboldi and Masafumi Kamachi. Time-space weak-constraint data assimilation for nonlinear models. *Tellus A*, 52(4):412–421, 2000.

[190] Eric Vanden-Eijnden et al. Transition-path theory and path-finding algorithms for the study of rare events. *Annual review of physical chemistry*, 61:391–420, 2010.

[191] Eric Vanden-Eijnden and Maddalena Venturoli. Revisiting the finite temperature string method for the calculation of reaction tubes and free energies. *The Journal of chemical physics*, 130(19):05B605, 2009.

[192] Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

[193] Frank Verstraete, Valentin Murg, and J Ignacio Cirac. Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems. *Advances in Physics*, 57(2):143–224, 2008.

[194] Gary R Waissi. Network flows: Theory, algorithms, and applications, 1994.

[195] David J. Wales and Jonathan P.K. Doye. Global optimization by basin-hopping and the lowest energy structures of Lennard-Jones clusters containing up to 110 atoms. *J. Phys. Chem. A*, 101(28):5111–5116, 1997.

[196] Duane Waliser, K Sperber, H Hendon, D Kim, E Maloney, M Wheeler, K Weickmann, Chidong Zhang, L Donner, J Gottschalck, et al. MJO simulation diagnostics. *Journal of Climate*, 22(11):3006–3030, 2009.

[197] Shusen Wang, Luo Luo, and Zhihua Zhang. SPSD matrix approximation vis column selection: theories, algorithms, and extensions. *The Journal of Machine Learning Research*, 17(1):1697–1745, 2016.

[198] Shusen Wang and Zhihua Zhang. Efficient algorithms and error analysis for the modified Nystrom method. In *Artificial Intelligence and Statistics*, pages 996–1004, 2014.

[199] E Weinan, Weiqing Ren, and Eric Vanden-Eijnden. Transition pathways in complex systems: Reaction coordinates, isocommittor surfaces, and transition tubes. *Chemical Physics Letters*, 413(1-3):242–247, 2005.

[200] E Weinan, Weiqing Ren, Eric Vanden-Eijnden, et al. Finite temperature string method for the study of rare events. *J. Phys. Chem. B*, 109(14):6688–6693, 2005.

[201] E Weinan and Eric Vanden-Eijnden. Towards a theory of transition paths. *Journal of statistical physics*, 123(3):503–523, 2006.

[202] E Weinan and Bing Yu. The deep ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018.

[203] Pierre Weiss. L'hypothèse du champ moléculaire et la propriété ferromagnétique. *J. Phys. Theor. Appl.*, 6(1):661–690, 1907.

[204] Steven R White. Density matrix formulation for quantum renormalization groups. *Physical review letters*, 69(19):2863, 1992.

[205] Peter Whittle. On stationary processes in the plane. *Biometrika*, pages 434–449, 1954.

[206] Peter Whittle. Stochastic-processes in several dimensions. *Bulletin of the International Statistical Institute*, 40(2):974–994, 1963.

[207] Christopher K Wikle, L Mark Berliner, and Noel Cressie. Hierarchical Bayesian space-time models. *Environmental and Ecological Statistics*, 5(2):117–154, 1998.

[208] Christopher K Wikle, Ralph F Milliff, Doug Nychka, and L Mark Berliner. Spatiotemporal hierarchical Bayesian modeling tropical ocean surface winds. *Journal of the American Statistical Association*, 96(454):382–397, 2001.

[209] Christopher KI Williams. Prediction with Gaussian processes: From linear regression to linear prediction and beyond. In *Learning in graphical models*, pages 599–621. Springer, 1998.

[210] Max A Woodbury. Inverting modified matrices. *Memorandum report*, 42(106):336, 1950.

[211] Franco Woolfe, Edo Liberty, Vladimir Rokhlin, and Mark Tygert. A fast randomized algorithm for the approximation of matrices. *Applied and Computational Harmonic Analysis*, 25(3):335–366, 2008.

[212] Jianlin Xia, Shivkumar Chandrasekaran, Ming Gu, and Xiaoye S Li. Fast algorithms for hierarchically semiseparable matrices. *Numerical Linear Algebra with Applications*, 17(6):953–976, 2010.

[213] Wanting Xu and Mihai Anitescu. A limited-memory multiple shooting method for weakly constrained variational data assimilation. *SIAM Journal on Numerical Analysis*, 54(6):3300–3331, 2016.

[214] Zixi Xu, Léopold Cambier, François-Henry Rouet, Pierre L'Eplatennier, Yun Huang, Cleve Ashcraft, and Eric Darve. Low-rank kernel matrix approximation using skeletonized interpolation with endo-or exo-vertices. *arXiv preprint arXiv:1807.04787*, 2018.

[215] Chenhan D Yu, James Levitt, Severin Reiz, and George Biros. Geometry-oblivious FMM for compressing dense SPD matrices. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–14, 2017.

[216] Jing Yu and Mihai Anitescu. Multidimensional sum-up rounding for integer programming in optimal experimental design. *Mathematical Programming*, 185(1):37–76, 2021.

[217] Dirk Zahn and Stefano Leoni. Nucleation and growth in pressure-induced phase transitions from molecular dynamics simulations: Mechanism of the reconstructive transformation of nacl to the cscl-type structure. *Physical review letters*, 92(25):250201, 2004.

[218] Hao Zhang. Inconsistent estimation and asymptotically equal interpolations in model-based geostatistics. *Journal of the American Statistical Association*, 99(465):250–261, 2004.

[219] Hao Zhang and Dale L Zimmerman. Towards reconciling two asymptotic frameworks in spatial statistics. *Biometrika*, 92(4):921–936, 2005.

[220] Shiwei Zhang. Auxiliary-field quantum monte carlo for correlated electron systems. emergent phenomena in correlated matter: Autumn school organized by the forschungszentrum jülich and the german research school for simulation sciences at forschungszentrum jülich 23-27 september 2013. *Lecture Notes of the Autumn School Correlated Electrons*, 3:2013, 2013.

[221] Bingge Zhao, Linfang Li, Fenggui Lu, Qijie Zhai, Bin Yang, Christoph Schick, and Yulai Gao. Phase transitions and nucleation mechanisms in metals studied by nanocalorimetry: A review. *Thermochimica Acta*, 603:2–23, 2015.

[222] Robert Zwanzig. *Nonequilibrium statistical mechanics*. Oxford university press, 2001.

APPENDIX A
SUPPLEMENT TO CHAPTER 2

A.1 Joint Process Kriging with Higher-Order Terms

In summary, based on our assumption of observation noise covariance matrices, the entire approach takes $O(N+N_l)$ forward solves and $O(N+N_l)$ observation noise covariance matrix linear system solves, and the dominant computational cost takes $O((m+n)(N+N_l)^2)$ time, although the workflow is highly parallelizable. The cost is quasilinear with n if $N, N_l = log(n)$.

A.2 Tensor Algebra Conventions

Computing the higher-order terms in the physics-based covariance model requires tensor operations. We use the tensor algebra with the following convention throughout the paper. Assume the vector-valued function F(z) defines a mapping $\mathbb{R}^n \to \mathbb{R}^m$ and the Jacobian is an $m \times n$ matrix where each entry $L_{ij} = \frac{\partial F_i(z)}{\partial z_j}$. The Hessian of F is a rank-three tensor H. We arrange the order of indices of the tensor by $H_{ijk} = \frac{\partial^2 F_j(z)}{\partial y_i \partial y_k}$. Therefore, H is a $n \times m \times n$ tensor. The Hessian tensor can be viewed as an array of Hessian matrices of each components of the function

$$H = \begin{pmatrix} H_1 \\ \vdots \\ H_m \end{pmatrix}, \tag{A.2.1}$$

where each $H_i \in \mathbb{R}^{n \times n}$ is the symmetric Hessian matrix of the ith component of F. The transpose of the tensor is defined by permuting the first and last indices as well as the entire array:

$$H^T = (H_1^T, H_2^T, \cdots, H_m^T). \tag{A.2.2}$$

Following this convention, the tensor-tensor, tensor-vector, and tensor-matrix product can be defined in an "elementwise sense": for any vector u and matrix U of proper size, we define

$$Hu = \begin{pmatrix} H_1 u \\ \vdots \\ H_m u \end{pmatrix}, \ HU = \begin{pmatrix} H_1 U \\ \vdots \\ H_m U \end{pmatrix}. \tag{A.2.3}$$

Additionally, we define the following tensor-tensor product, which is useful in the quadratic form computation. Assume $\hat{H}$ is an $n \times \hat{m} \times n$ tensor,

$$HU\hat{H}^T U = \begin{pmatrix} H_1 U \hat{H}_1^T U & \cdots & H_1 U \hat{H}_{\hat{m}}^T U \\ \vdots & \ddots & \vdots \\ H_m U \hat{H}_1^T U & \cdots & H_m U \hat{H}_{\hat{m}}^T U \end{pmatrix}. \tag{A.2.4}$$

The left multiplications are defined in the same elementwise sense, with left multiplications in each entry instead. As a result, bilinear forms of such tensors on vectors $u, v, \in \mathbb{R}^m$ become a contraction along indices 1,3, and thus the vector

$$v^T H u = \begin{pmatrix} v^T H_1 u \\ \vdots \\ v^T H_m u \end{pmatrix} \in \mathbb{R}^m. \tag{A.2.5}$$

The trace operation of the rank-three tensor is a tensor contraction to a vector taking over the first and last indices defined by

$$\mathrm{tr}(H) = [(\sum_{i,k} H_{ijk}\delta_{ik})_j] = \begin{pmatrix} \mathrm{tr}(H_1) \\ \vdots \\ \mathrm{tr}(H_m) \end{pmatrix}. \tag{A.2.6}$$

In particular, the trace of a rank-four tensor of form (A.2.4) can be defined:

$$\mathrm{tr}(HU\hat{H}^T U) = \begin{pmatrix} \mathrm{tr}(H_1 U \hat{H}_1^T U) & \cdots & \mathrm{tr}(H_1 U \hat{H}_{\hat{m}}^T U) \\ \vdots & \ddots & \vdots \\ \mathrm{tr}(H_m U \hat{H}_1^T U) & \cdots & \mathrm{tr}(H_m U \hat{H}_{\hat{m}}^T U) \end{pmatrix} \in \mathbb{R}^{m \times \hat{m}}. \tag{A.2.7}$$

A.3 Derivation of Higher-Order Terms

Here the expectation of the quadratic form of centered multivariate Gaussian random variable is extensively used. Since $\delta z \in \mathbb{R}^n$ is a zero-mean Gaussian random variable, we have the following important properties [36]: For any $n \times n$ symmetric matrix A, B,

$$\mathrm{E}(\delta z^T A \delta z) = \mathrm{tr}(A\mathrm{Cov}(z,z)) \tag{A.3.1}$$

$$\mathrm{E}[(\delta z^T A \delta z)(\delta z^T B \delta z)] = 2\mathrm{tr}(A\mathrm{Cov}(z,z)B\mathrm{Cov}(z,z)) + \mathrm{tr}(A\mathrm{Cov}(z,z))\mathrm{tr}(B\mathrm{Cov}(z,z)). \tag{A.3.2}$$

Proof. Since δz is a mean zero multivariate random variable,

$$\begin{aligned} \mathrm{E}(\delta z^T A \delta z) &= \mathrm{E}(\mathrm{tr}(\delta z^T A \delta z)) \\ &= \mathrm{E}(\mathrm{tr}(A \delta z \delta z^T)) \\ &= \mathrm{tr}(A\mathrm{E}(\delta z \delta z^T)) \\ &= \mathrm{tr}(A\mathrm{Cov}(z, z)). \end{aligned} \tag{A.3.3}$$

For Equation (A.3.2), we start from a simpler case. Assume $x \in \mathbb{R}^n$ is a standard normal random vector $x_i \sim \mathcal{N}(0, 1)$ i.i.d. We first consider for any symmetric $n \times n$ matrices $\tilde{A}, \tilde{B}$,

$$\begin{aligned} \mathrm{E}(x^T \tilde{A} x x^T \tilde{B} x) =& \mathrm{E}\left(\sum_{i,j,k,l=1}^{n} \tilde{A}_{ij}\tilde{B}_{kl} x_i x_j x_k x_l\right) \\ =& \mathrm{E}\left(\sum_{i \neq j} \tilde{A}_{ii}\tilde{B}_{jj} x_i^2 x_j^2\right) + \mathrm{E}\left(\sum_{i \neq j} \tilde{A}_{ij}\tilde{B}_{ij} x_i^2 x_j^2\right) + \mathrm{E}\left(\sum_{i \neq j} \tilde{A}_{ij}\tilde{B}_{ji} x_i^2 x_j^2\right) \\ & + \mathrm{E}\left(\sum_{i=1}^{n} \tilde{A}_{ii}\tilde{B}_{ii} x_i^4\right) \\ =& \mathrm{E}\left(\sum_{i \neq j} \tilde{A}_{ii}\tilde{B}_{jj}\right) + 2\mathrm{E}\left(\sum_{i \neq j} \tilde{A}_{ij}\tilde{B}_{ji}\right) + 3\mathrm{E}\left(\sum_{i=1}^{n} \tilde{A}_{ii}\tilde{B}_{ii}\right). \end{aligned} \tag{A.3.4}$$

Note that the last step comes from the fact that $\tilde{B}$ is a symmetric matrix and the Isserlis' theorem. On the other hand, note that

$$\mathrm{tr}(\tilde{A}\tilde{B}) = \sum_{i,j=1}^{n} \tilde{A}_{ij}\tilde{B}_{ji} = \sum_{i \neq j} \tilde{A}_{ij}\tilde{B}_{ji} + \sum_{i=1}^{n} \tilde{A}_{ii}\tilde{B}_{ii} \tag{A.3.5}$$

$$\mathrm{tr}(\tilde{A})\mathrm{tr}(\tilde{B}) = \sum_{i \neq j} \tilde{A}_{ii}\tilde{B}_{jj} + \sum_{i=1}^{n} \tilde{A}_{ii}\tilde{B}_{ii}. \tag{A.3.6}$$

Comparing (A.3.4) with (A.3.5) and (A.3.6), we have

$$\mathrm{E}(x^T \tilde{A} x x^T \tilde{B} x) = 2\mathrm{tr}(\tilde{A}\tilde{B}) + \mathrm{tr}(\tilde{A})\mathrm{tr}(\tilde{B}). \tag{A.3.7}$$

Now since δz is a mean zero multivariate normal vector, there exists a matrix L such that $\mathrm{Cov}(z,z) = LL^T$ and $\delta z = Lx$. Using (A.3.7), we have

$$\begin{aligned}
\mathrm{E}[(\delta z^T A \delta z)(\delta z^T B \delta z)] &= \mathrm{E}[(x^T(L^T A L)x)(x^T(L^T B L)x)] \\
&= 2\mathrm{tr}(L^T A L L^T B L) + \mathrm{tr}(L^T A L)\mathrm{tr}(L^T B L) \\
&= 2\mathrm{tr}(A L L^T B L L^T) + \mathrm{tr}(A L L^T)\mathrm{tr}(B L L^T) \\
&= 2\mathrm{tr}(A\mathrm{Cov}(z,z)B\mathrm{Cov}(z,z)) + \mathrm{tr}(A\mathrm{Cov}(z,z))\mathrm{tr}(B\mathrm{Cov}(z,z)).
\end{aligned} \tag{A.3.8}$$

We thus have proved (A.3.2).

$\square$

Following our convention of tensor algebra for Hessian tensor H, we get

$$\begin{aligned}
\overline{\delta z^T H \delta z} &= \mathrm{E}(\delta z^T H \delta z) \\
&= \begin{pmatrix} \mathrm{E}(\delta z^T H_1 \delta z) \\ \vdots \\ \mathrm{E}(\delta z^T H_m \delta z) \end{pmatrix} \\
&= \begin{pmatrix} \mathrm{tr}(H_1 \mathrm{Cov}(z,z)) \\ \vdots \\ \mathrm{tr}(H_m \mathrm{Cov}(z,z)) \end{pmatrix} \\
&= \mathrm{tr}(H\mathrm{Cov}(z,z)).
\end{aligned} \tag{A.3.9}$$

Then for Hessian tensors $H \in \mathbb{R}^{n\times m\times n}$, $\hat{H} \in \mathbb{R}^{n\times \hat{m}\times n}$,

$$\overline{\delta z^T H \delta z}\;\overline{\delta z^T \hat{H}^T \delta z} = \operatorname{tr}(H\operatorname{Cov}(z,z))\operatorname{tr}(\hat{H}\operatorname{Cov}(z,z))^T. \tag{A.3.10}$$

We thus have

$$\overline{\delta z^T H \delta z \delta z^T \hat{H}^T \delta z} = \begin{pmatrix} \mathrm{E}(\delta z^T H_1 \delta z \delta z^T \hat{H}_1^T \delta z) & \cdots & \mathrm{E}(\delta z^T H_1 \delta z \delta z^T \hat{H}_{\hat{m}}^T \delta z) \\ \vdots & \ddots & \vdots \\ \mathrm{E}(\delta z^T H_m \delta z \delta z^T \hat{H}_1^T \delta z) & \cdots & \mathrm{E}(\delta z^T H_m \delta z \delta z^T \hat{H}_{\hat{m}}^T \delta z) \end{pmatrix}$$
$$\overset{(A.3.2)}{=} 2\operatorname{tr}(H\operatorname{Cov}(z,z)\hat{H}^T\operatorname{Cov}(z,z)) + \operatorname{tr}(H\operatorname{Cov}(z,z))\operatorname{tr}(\hat{H}\operatorname{Cov}(z,z))^T. \tag{A.3.11}$$

Algorithm 8: Joint Process Kriging with Higher-Order Terms $\mathcal{M}_2^+$

```
Use the low-rank approximation with $N = O(\log n)$. Specifically, $\mathrm{Cov}(z_o, z_o) \approx C_{z_o}^T K C_{z_o}, \mathrm{Cov}(z_o, z) \approx C_{z_o}^T K C_z, \mathrm{Cov}(z_p, z) \approx C_{z_p}^T K C_z, \mathrm{Cov}(z_p, z_o) \approx C_{z_p}^T K C_{z_o}$.
Step 1. Compute $A_1 = L_o C_z^T \in \mathbb{R}^{d\times N}$, $A_2 = L_p C_z^T \in \mathbb{R}^{(m-d)\times N}$ by $O(N)$ forward solves.
Step 2. Compute the Cholesky factorization $K = P^T P$, which takes $O(N^3)$ time.
Step 3. Draw $2N_l$ independent Rademacher vectors $\{u_i\}$,$\{v_i\} \in \mathbb{R}^N$, $i = 1, 2, \ldots, N_l$. Compute
    $$\phi_i = C_z^T P^T u_i, \psi_i = C_z^T P^T v_i. \tag{A.1.1}$$
    This takes $O(2N_l(N^2 + nN))$ time.
Step 4. Compute vector-Hessian-vector product by approximation (2.2.15). This approximation takes $O(N_l)$ forward solves. Denote $w_i = \psi_i^T H_o \phi_i$, $w_i' = \psi_i^T H_p \phi_i$. Then set the matrices $A_3 \leftarrow \frac{1}{\sqrt{2N_l}}(w_1, w_2, \cdots, w_{N_l}) \in \mathbb{R}^{d\times N_l}$,
    $A_4 \leftarrow \frac{1}{\sqrt{2N_l}}(w_1', w_2', \cdots, w_{N_l}') \in \mathbb{R}^{(m-d)\times N_l}$. Using (2.3.22), we will approximate higher-order terms by
    $$\frac{1}{2}\mathrm{tr}(H_o \mathrm{Cov(z,z)} H_o^T \mathrm{Cov(z,z)}) \approx A_3 A_3^T, \tag{A.1.2}$$
    $$\frac{1}{2}\mathrm{tr}(H_p \mathrm{Cov(z,z)} H_o^T \mathrm{Cov(z,z)}) \approx A_4 A_3^T. \tag{A.1.3}$$
Step 5. Carry out the kriging computation (2.3.15) with components (2.3.16) and (2.3.17) in two steps. First, solve the modified inverse problem
    $$\begin{aligned}\alpha &= \left(\begin{pmatrix} K_{\epsilon_y} & 0 \\ 0 & K_{\epsilon_z}\end{pmatrix} + \begin{pmatrix} A_3A_3^T & 0 \\ 0 & 0\end{pmatrix} + \begin{pmatrix} A_1 & 0 \\ 0 & C_{z_o}^T\end{pmatrix}\begin{pmatrix} K & K \\ K & K\end{pmatrix}\begin{pmatrix} A_1^T & 0 \\ 0 & C_{z_o}\end{pmatrix}\right)^{-1}\begin{pmatrix} y_o - m(y_o) \\ z_o - m(z_o)\end{pmatrix}, \\ &= \left(\begin{pmatrix} K_{\epsilon_y} & 0 \\ 0 & K_{\epsilon_z}\end{pmatrix} + \begin{pmatrix} A_3 & A_1P^T \\ 0 & C_{z_o}^T P^T\end{pmatrix}\begin{pmatrix} A_3^T & 0 \\ PA_1^T & PC_{z_o}\end{pmatrix}\right)^{-1}\begin{pmatrix} y_o - m(y_o) \\ z_o - m(z_o)\end{pmatrix} \end{aligned} \tag{A.1.4}$$
    by applying the SMW formula. The time cost is dominated by computing the matrix products $\begin{pmatrix} A_3^T & 0 \\ PA_1^T & PC_{z_o}\end{pmatrix}\begin{pmatrix} K_{\epsilon_y}^{-1} & 0 \\ 0 & K_{\epsilon_z}^{-1}\end{pmatrix}\begin{pmatrix} A_3 & A_1P^T \\ 0 & C_{z_o}^T P^T\end{pmatrix}$, which take $O(N + N_l)$ linear system solves with the observation noise matrix and $O((n+m)(N + N_l)^2)$ assembly time.
Step 6. **Return** the final solution
    $$\begin{pmatrix} m(y_p) \\ m(z_p)\end{pmatrix} + \left(\begin{pmatrix} A_4A_3^T & 0 \\ 0 & 0\end{pmatrix} + \begin{pmatrix} A_2 & 0 \\ 0 & C_{z_p}^T\end{pmatrix}\begin{pmatrix} K & K \\ K & K\end{pmatrix}\begin{pmatrix} A_1^T & 0 \\ 0 & C_{z_o}\end{pmatrix}\right)\alpha, \tag{A.1.5}$$
    by matrix-vector product. It takes $O(2(n+m)N + 2mN_l + 4N^2)$ time.
```

APPENDIX B

SUPPLEMENT TO CHAPTER 3

B.1 Motivating Example for Full Dataset Fitting

Before moving forward to sophisticated covariance and physical models, we demonstrate the benefit of performing MLE over the entire large datasets of measurements. Here we use a nonstationary Gaussian process on the $1D$ interval $[0, 10]$. The covariance model of the process is the square exponential covariance

$$K(x_i, x_j) = \sigma \exp\left(-\frac{(x_i - x_j)^2}{2l_il_j}\right), \text{ where } x_i,\ x_j \in [0, 10] \tag{B.1.1}$$

where the length scale parameter l_i, l_j are given by a linear function of x, i.e. $l_i = l(x_i) = a + bx_i$. We set the ground truth parameters $\sigma = 1$, $a = 0.1$, $b = 0.6$. Therefore the true length scale parameter l ranges from 0.1 to 6.1 over the entire domain. Here we treat the magnitude parameter σ as known. By simulating synthetic observations from the given Gaussian process with an equally spaced grid points of mesh size $\Delta x = 0.05$, we attempt to recover the unknown parameters a and b via MLE.

We consider the following three settings:

- Setting 1: full observations from $[0, 10]$ with resolution $\Delta x = 0.05$.
- Setting 2: subsampled observations from $[0, 10]$ with a coarser resolution $\Delta x = 0.2$.
- Setting 3: truncated observations from $[0, 2.5]$ with full resolution $\Delta x = 0.05$.

The three settings corresponds to three scenarios when dealing with large datasets of raw observations. In setting 1, we perform MLE on the entire dataset and possibly apply approximation methods to confront the scaling issue. Alternatively we can subsample the dataset by taking a coarser subset or restricting the observations to a partial domain. These approaches will effectively reduce the problem size, but as a result, inducing difficulties and inaccuracy in parameter identifications, as what we will see next.

We simulate 20 independent realizations of the same process as observations and perform MLE for the three settings above. We show the parameter point estimations and the corresponding 95% confidence intervals in Fig. B.1.1. We observe that with full observations we can identify both parameters accurately (with small confidence intervals). When we subsample the observations as in setting 2, the observation density ($\Delta x = 0.2$) is not enough to capture the finest correlation scale ($l_{\text{min}} = 0.1$). We can still successfully identify the true parameter but the confidence intervals are much wider. In setting 3, the truncated subdomain contains information about finer scale correlations, as the correlation length scale increases from 0 to 10. We see the intercept term a can be estimated accurately but estimating b poses much more uncertainties. In summary, it is indeed possible to subsample the observations to circumvent the scaling issue when dealing with large datasets. However this comes with a cost as partial observations may fail to provide enough information to identify the parameters. A better alternative is to use approximation method e.g. our approach in this paper, which introduces a clearer framework for complexity / accuracy tradeoff.

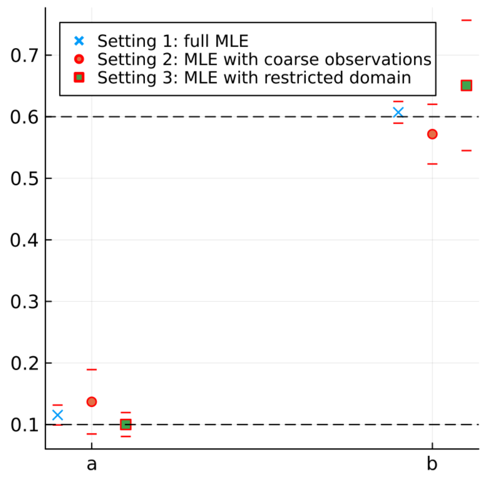


Figure B.1.1: MLE point estimation and confidence intervals for parameters a and b. Blue crosses, red circles and green squares correspond to settings 1, 2, 3, respectively.

B.2 Two Quasilinear Trace Operations for HODLR Matrices

Assume A and B are both HODLR matrices with number of levels τ and fixed local rank k. Based on the basic factorization (3.3.3) we have

$$A = \bar{A}(I + U^{(\tau)}V^{(\tau)T}) \cdots (I + U^{(1)}V^{(1)T}), \tag{B.2.1}$$

where $\bar{A}$ is a block-diagonal matrix containing all the leaf level blocks of A. $(I + U^{(i)}V^{(i)T})$ is a block-diagonal matrix with 2^{i-1} blocks where each block is a rank-$2k$ update to identity. More details and the connection between off-diagonal blocks and low-rank update to identity can be found in Appendix B.3.

We assume the level grows as $\log n$, i.e. $\tau = O(\log n)$. (B.2.1) is the original form of factorization in [5]. By factorizing the finer level diagonal factors to the other side, A can also be factorized in the following "transposed" form in the same $O(n \log^2 n)$ complexity,

$$A = (I + V^{(1)}U^{(1)T}) \cdots (I + V^{(\tau)}U^{(\tau)T})\bar{A}. \tag{B.2.2}$$

Note that here we assume A is symmetric. For asymmetric matrix the two factorizations (B.2.1), (B.2.2) also exist, but the factors generally do not have explicit correspondence.

B.2.1 Matrix-Matrix Product AB And $A^{-1}B$

Now we discuss in detail the algorithm for performing matrix-matrix product AB and $A^{-1}B$ in HODLR format. Instead of forming the resulting product matrix explicitly, we write it as a sum of matrices. We start from HODLR matrices A, B with the same number of levels τ and fixed local rank k. Additionally we assume A, B have exactly the same size and hierarchical partitioning. Consider product AB. We use the factorization form (B.2.1).

Step 1 The first step is to multiply the rightmost factor of A with HODLR matrix B. We have

$$(I + U^{(1)}V^{(1)T})B = B + U^{(1)}V^{(1)T}B = B + \bar{U}^{(1)}\bar{V}^{(1)T}, \tag{B.2.3}$$

where $\bar{U}^{(1)} = U^{(1)}$ and $\bar{V}(1) = B^T V^{(1)}$. Note that the second term is of rank k. We can store the two low-rank factors $U_1^{(1)}$, $V_1^{(1)}$ to avoid computing it explicitly.

The result now is written as the sum of an HODLR matrix and a low-rank component. The next step is to apply subsequent factors of A. This is done by two steps.

Step 2 Notice that the second level factor has the following block diagonal format [5],

$$(I + U^{(2)}V^{(2)T}) = \begin{bmatrix} I + U_1^{(2)}V_1^{(2)T} & 0 \\ 0 & I + U_2^{(2)}V_2^{(2)T} \end{bmatrix}. \tag{B.2.4}$$

where each block has size one half of the entire matrix and the sized of the identity matrices is chose to match. To apply the second level factor to the product, we first update the low-rank components. In this case we have

$$\bar{U}^{(1)}\bar{V}^{(1)T} \leftarrow (I + U^{(2)}V^{(2)T})\bar{U}^{(1)}\bar{V}^{(1)T}, \tag{B.2.5}$$

which can be done by updating the existing low-rank factors $\bar{U}^{(1)} \leftarrow (I+U^{(2)}V^{(2)T})\bar{U}^{(1)}$ and $\bar{V}^{(1)} \leftarrow \bar{V}^{(1)}$ via applying the two diagonal blocks in (B.2.4) to the corresponding rows of $\bar{U}^{(1)}$.

Step 3 The next step is to apply the second level factor to the HODLR matrix. Divide the HODLR matrix into its first level, B can be written as

$$B = \begin{bmatrix} B_1^{(1)} & G_1^{(1)}H_1^{(1)T} \\ H_2^{(1)}G_2^{(1)T} & B_2^{(1)} \end{bmatrix}. \tag{B.2.6}$$

Then we can compute

$$
\begin{aligned}
(I+U^{(2)}V^{(2)T})B &= \begin{bmatrix} I+U_1^{(2)}V_1^{(2)T} & 0 \\ 0 & I+U_2^{(2)}V_2^{(2)T} \end{bmatrix} \begin{bmatrix} B_1^{(1)} & G_1^{(1)}H_1^{(1)T} \\ H_2^{(1)}G_2^{(1)T} & B_2^{(1)} \end{bmatrix}, \\
&= \begin{bmatrix} B_1^{(1)} & \bar{G}_1^{(1)}\bar{H}_1^{(1)T} \\ \bar{H}_2^{(1)}\bar{G}_2^{(1)T} & B_2^{(1)} \end{bmatrix} + \begin{bmatrix} U_1^{(2)}V_1^{(2)T}B_1^{(1)} & 0 \\ 0 & U_2^{(2)}V_2^{(2)T}B_2^{(1)} \end{bmatrix}.
\end{aligned}
\tag{B.2.7}
$$

where $\bar{G}_1^{(1)} = (I+U_1^{(2)}V_1^{(2)T})G_1^{(1)}$, $\bar{H}_1^{(1)} = H_1^{(1)}$, $\bar{G}_2^{(1)} = G_2^{(1)}$, $\bar{H}_2^{(1)} = (I+U_2^{(2)}V_2^{(2)T})H_2^{(1)}$. Notice the first term is still an HODLR matrix and the second term is block-diagonally low-rank. Further denote

$$
\begin{bmatrix} U_1^{(2)}V_1^{(2)T}B_1^{(1)} & 0 \\ 0 & U_2^{(2)}V_2^{(2)T}B_2^{(1)} \end{bmatrix} = \begin{bmatrix} \bar{U}_1^{(2)}\bar{V}_1^{(2)T} & 0 \\ 0 & \bar{U}_2^{(2)}\bar{V}_2^{(2)T} \end{bmatrix} = \bar{U}^{(2)}\bar{V}^{(2)T},
\tag{B.2.8}
$$

and define the HODLR matrix $\tilde{B}$ in the multiplier by

$$
\tilde{B} \leftarrow \begin{bmatrix} B_1^{(1)} & \bar{G}_1^{(1)}\bar{H}_1^{(1)T} \\ \bar{H}_2^{(1)}\bar{G}_2^{(1)T} & B_2^{(1)} \end{bmatrix}.
\tag{B.2.9}
$$

Note that B and $\tilde{B}$ are HODLR matrices with the same size and hierarchical structure. However the off-diagonal blocks have been updated. Now the product can be written as

$$
(I+U^{(2)}V^{(2)T})(I+U^{(1)}V^{(1)T})B = \tilde{B} + \bar{U}^{(1)}\bar{V}^{(1)T} + \bar{U}^{(2)}\bar{V}^{(2)T}.
\tag{B.2.10}
$$

We notice not only the HODLR matrix has been updated, an extra second level diagonal low-rank component ($\bar{U}^{(2)}\bar{V}^{(2)T}$ in our case) has been generated as well.

Step 4 Steps 2 through 4 are repeated until we reach level τ. For each level, both the HODLR matrix $\bar{B}$ and all the low-rank components from the previous levels need to be updated. Finally we have

$$(I + U^{(\tau)}V^{(\tau)T}) \cdots (I + U^{(1)}V^{(1)T})B = \tilde{B} + \sum_{i=1}^{\tau} \bar{U}^{(i)}\bar{V}^{(i)T}. \tag{B.2.11}$$

Here $\tilde{B}$ is a HODLR matrix with the same structure as B. Note that each $\bar{U}^{(i)}$ and $\bar{V}^{(i)}$ have the same dimension and structure as $U^{(i)}$ and $V^{(i)}$: they are block diagonal with $n/2^{i-1} \times 2k$ blocks.

Step 5 The final step is to apply the leaf level block diagonal matrix $\bar{A}$ to the product we have gotten. Since we assume A, B have the same hierarchical partitioning, it can be done by applying each diagonal block of $\bar{A}$ blockwisely to all low-rank components and the low-rank factors of $\bar{B}$ at all levels. The operation has been extensively used when factorizing the HODLR matrix, see [6] for an example.

Now we analyze the computational complexity of the given workflow. Using (B.2.1), we apply the level i factor of A. This requires multiplying the block-diagonal matrix $(I + U^{(i)}V^{(i)T})$ with all $(i-1)$ existing left block-diagonally low-rank components $\bar{U}^{(1)}, \cdots, \bar{U}^{(i-1)}$. *The essential observation that makes this efficient is that block diagonal structures at level i can be mapped into block diagonal structures at all other levels k, $k \leq i$ since they have the finest structures among the latter.* Take $\bar{U}^{(i-1)}$ as an example, exploring the preceding observation, we can express level i in block diagonal form for the $i-1$ partition and compute the matrix multiplication as follows,

$$\begin{aligned}
&(I + U^{(i)}V^{(i)T})\bar{U}^{(i-1)}, && \text{(B.2.12)}\\
&= \operatorname{diag}\left(\begin{bmatrix} I + U_1^{(i)}V_1^{(i)T} & \cdots & I + U_{2^{i-1}}^{(i)}V_{2^{i-1}}^{(i)T}\end{bmatrix}\right)\cdot \operatorname{diag}\left(\begin{bmatrix}\bar{U}_1^{(i-1)} & \cdots & \bar{U}_{2^{i-2}}^{(i-1)}\end{bmatrix}\right),\\
&= \operatorname{diag}\left(\begin{bmatrix}\left(I + U_1^{(i)}V_1^{(i)T}\right)\bar{U}_{1,1}^{(i-1)} & \left(I + U_2^{(i)}V_2^{(i)T}\right)\bar{U}_{1,2}^{(i-1)} & \cdots\end{bmatrix}\right)\left(I + U_{2^{i-1}}^{(i)}V_{2^{i-1}}^{(i)T}\right)\bar{U}_{2^{i-2},2}^{(i-1)}.
\end{aligned}$$

where diag(v) denotes the block-diagonal matrix whose diagonal blocks are given by the component block matrices v. $\bar{U}_1^{(i-1)} = \begin{bmatrix}\bar{U}_{1,1}^{(i-1)}\\ \bar{U}_{1,2}^{(i-1)}\end{bmatrix}$ is a partition of matrix $\bar{U}_1^{(i-1)}$ compatible with the finer partition i. To evaluate the complexity consequences of our approach, we need to investigate the size of the block matrices in detail. For simplicity we assume bipartition for all levels in the hierarchical partitioning structure and therefore that n is divisible by 2^{τ}. In this case, $(I+U^{(i)}V^{(i)T})$ has diagonal blocks of size $n/2^{i-1} \times n/2^{i-1}$. In contrast, $\bar{U}^{(i-1)}$ has diagonal blocks of size $n/2^{i-2}\times 2k$. Therefore $\bar{U}_{1,1}^{(i-1)}, \bar{U}_{1,2}^{(i-1)}$ are both of size $n/2^{i-1}\times 2k$. To solve (B.2.12), we need to compute matrix multiplications of type $\left(I + U_1^{(i)}V_1^{(i)T}\right)\bar{U}_{1,1}^{(i-1)}$ for all 2^{i-1} blocks. By multiplying the low-rank factors from right to left, each term takes $8nk^2/2^{i-1}$, yielding a total cost of $8nk^2$. At the following level, $\bar{U}^{(i-2)}$ has diagonal blocks of size $n/2^{i-3} \times 2k$. Therefore each diagonal block of of should be quartered to match the multiplier $(I + U^{(i)}V^{(i)T})$. Ultimately there are still 2^{i-1} block multiplications of the same size as $\left(I + U_1^{(i)}V_1^{(i)T}\right)\bar{U}_{1,1}^{(i-1)}$. The computational cost is $8nk^2$ as well. Repeating the updating procedure for all $(i-1)$ existing low-rank components, the total cost is $8(i-1)nk^2$.

Next, updating $\tilde{B}$ requires applying $(I+U^{(i)}V^{(i)T})$, $i \geq 2$, which has 2^{i-1} blocks (B.2.21), to each of the left low-rank off-diagonal components of $\tilde{B}$ at levels $1, \cdots, i-1$, (B.2.7) . The same trick detailed in (B.2.12) can be applied as well. For example in level 1, two low-rank factors of size $n/2 \times k$ need to be updated ($\bar{G}_1^{(1)}$ and $\bar{H}_2^{(1)}$ in (B.2.7)). By partitioning each low rank component of $\tilde{B}$ into 2^{i-2} blocks (since $i = 2$ no splitting was required in (B.2.7)), we can match them with the diagonal blocks in $(I + U^{(i)}V^{(i)T})$. The resulting 2^i block multiplications take $4nk^2$ time. For all $(i-1)$ levels, the cost is $4(i-1)nk^2$.

Additionally to generate the new low-rank terms, $\bar{U}^{(i)}$, $\bar{V}^{(i)}$ in (B.2.10) and (B.2.11), we need to apply the diagonal low-rank components of $(I + U^{(i)}V^{(i)T})$, i.e. $U_j^{(i)}V_j^{(i)T}$ in (B.2.21) to the diagonal components of $\tilde{B}$ on level i (i.e the computation of (B.2.8)). This step can be done by regular HODLR-vector products (after transposing the computation). The complexity of HODLR-vector products has been extensively studied, see [77]. Note that the diagonal component matrices of $\tilde{B}$ at level i have size $n/2^i \times n/2^i$ and $(\tau - i)$ levels of their own. The total computational cost is thus upper bounded $O(nk^2\tau)$.

Adding all these operations, we obtain a total computational cost,

$$\sum_{i=1}^{\tau} 8(i-1)nk^2 + 4(i-1)nk^2 + O(nk^2\tau) = O(nk^2\tau^2) = O(n\log^2 n). \tag{B.2.13}$$

Here we used the assumption $\tau = O(\log n)$ and that k is a fixed constant. Finally we need to apply all 2^τ leaf blocks of $\bar{A}$ to $\tilde{B} + \sum_{i=1}^{\tau} \bar{U}^{(i)}\bar{V}^{(i)T}$. That is equivalent to applying all leaf blocks of $\bar{A}$ to all the left low-rank components and leaf blocks of $\tilde{B}$, and all $\bar{U}^{(i)}$. By blockwise application, each diagonal leaf block of $\bar{A}$ will be applied to $2k$ vectors (left low-rank components of off-diagonal blocks) in each level and an extra $O(k)$ vectors for the leaf blocks of $\tilde{B}$, yielding $O(k^3\tau)$ complexity. Similarly, to multiply with $\sum_{i=1}^{\tau} \bar{U}^{(i)}\bar{V}^{(i)T}$, each $\bar{A}$ leaf block needs to be applied to $2k$ vectors (diagonal blocks of $\bar{U}^{(i)}$), yielding $O(k^3\tau)$ complexity. For all blocks, the total complexity is $O(k^3\tau \times 2^\tau) = O(nk^2\tau) = O(n\log n)$. Summarizing everything up, the total cost of computing AB for two HODLR matrices in the format of the right hand side of (B.2.11) is $O(n\log^2 n)$.

Next we discuss the algorithm for computing $A^{-1}B$ in a format of the right hand side of (B.2.11). Since the inverse will reverse the order of the factors, we factorize A in form

(B.2.2). Using the Woodbury identity, we have

$$\begin{aligned} A^{-1}B &= \bar{A}^{-1}(I + U^{(\tau)}V^{(\tau)T})^{-1} \cdots (I + U^{(1)}V^{(1)T})^{-1}B \\ &= \bar{A}^{-1}(I - U^{(\tau)}(I + V^{(\tau)T}U^{(\tau)})^{-1}V^{(\tau)T}) \cdots (I - U^{(1)}(I + V^{(1)T}U^{(1)})^{-1}V^{(1)T})B. \end{aligned} \tag{B.2.14}$$

Comparing (B.2.11) and (B.2.14), each factor is still a low-rank update to identity. The only differences are now we need to apply the inverse of a matrix $(I + V^{(i)T}U^{(i)})$ and the inverse of leaf blocks of $\bar{A}$. Luckily, $V^{(i)T}U^{(i)}$ consists of only diagonal blocks of size $O(k)$ which makes the linear system efficiently computable. If we denote $\tilde{V}^{(i)T} = (I + V^{(i)T}U^{(i)})^{-1}V^{(i)T}$ we can apply the same algorithm we did to compute AB to compute $A^{-1}B$ in the format of the right hand side of (B.2.11).

Now we analyze the additional computational complexity compared to the original AB product algorithm. One of the extra operations is to obtain $\tilde{V}^{(i)T} = (I + V^{(i)T}U^{(i)})^{-1}V^{(i)T}$. Recall that both $U^{(i)}$ and $V^{(i)}$ are block-diagonal with a total number of 2^{i-1} diagonal blocks of rank $2k$. For each level i, the diagonal blocks of $V^{(i)}$ are of size $n/2^{i-1} \times 2k$. Therefore the complexity of computing each block of $\tilde{V}^{(i)}$ is $8nk^3/2^{i-1}$. For all blocks the complexity becomes $8nk^3$. Repeating the same operation to obtain all $\tilde{V}^{(i)}$, $i = 1, \cdots, \tau$ requires $O(nk^3\tau) = O(n \log n)$ complexity. Another difference is that now we need to apply the inverse of leaf blocks in $\bar{A}$. Based on similar considerations, each leaf block needs to be applied to $O(k\tau)$ vectors, yielding $O(k^4\tau)$ complexity. Taking all blocks into consideration, the total extra complexity is $O(k^4\tau \times 2^\tau) = O(nk^3\tau) = O(n \log n)$.

In summary, the total cost of computing either AB or $A^{-1}B$ for two HODLR matrices scales as $O(n \log^2 n)$ and the result can be expressed as the sum of an HODLR matrix and τ terms with low rank blocks, as in the right hand side of (B.2.11). The extra memory required to store the low-rank terms is $O(nk\tau) = O(n \log n)$.

B.2.2 Product of Form $A^{-1}BC^{-1}D$

In the same vein, we can compute the product of form $A^{-1}BC^{-1}D$ given four HODLR matrices A, B, C, D. Assume all four HODLR matrices have exactly the same size of hierarchical partitioning. The local rank of all off-diagonal blocks is fixed at k. Using B.2.1 we are able to compute $A^{-1}B$ and $C^{-1}D$ separately. Assume

$$A^{-1}B = \bar{B} + \sum_{i=1}^{\tau} \bar{U}^{(i)}\bar{V}^{(i)T}, \tag{B.2.15}$$

$$C^{-1}D = \bar{D} + \sum_{i=1}^{\tau} \bar{\mathbf{U}}^{(i)}\bar{\mathbf{V}}^{(i)T}. \tag{B.2.16}$$

By multiplying each term separately, we have

$$A^{-1}BC^{-1}D = \bar{B}\bar{D} + \sum_{i=1}^{\tau} \bar{U}^{(i)}\bar{V}^{(i)T}\bar{D} + \sum_{i=1}^{\tau} \bar{B}\bar{\mathbf{U}}^{(i)}\bar{\mathbf{V}}^{(i)T} + \sum_{i=1}^{\tau}\sum_{j=1}^{\tau} \bar{U}^{(i)}\bar{V}^{(i)T}\bar{\mathbf{U}}^{(j)}\bar{\mathbf{V}}^{(j)T}. \tag{B.2.17}$$

The first term is the product of two HODLR matrices. Applying the algorithms in B.2.1 again, we can write

$$\bar{B}\bar{D} = \tilde{D} + \sum_{i=1}^{\tau} \bar{\mathcal{U}}^{(i)}\bar{\mathcal{V}}^{(i)T}. \tag{B.2.18}$$

In summary, the product can be written as

$$\begin{aligned} A^{-1}BC^{-1}D =& \tilde{D} + \sum_{i=1}^{\tau} \bar{\mathcal{U}}^{(i)}\bar{\mathcal{V}}^{(i)T} + \sum_{i=1}^{\tau} \bar{U}^{(i)}\bar{V}^{(i)T}\bar{D} + \sum_{i=1}^{\tau} \bar{B}\bar{\mathbf{U}}^{(i)}\bar{\mathbf{V}}^{(i)T} \\ &+ \sum_{i=1}^{\tau}\sum_{j=1}^{\tau} \bar{U}^{(i)}\bar{V}^{(i)T}\bar{\mathbf{U}}^{(j)}\bar{\mathbf{V}}^{(j)T}. \end{aligned} \tag{B.2.19}$$

From B.2.1, computing $A^{-1}B$, $C^{-1}D$, $\bar{B}\bar{D}$ all take $O(n \log^2 n)$ time. In total, given

HODLR matrices A, B, C, D we can write the product of $A^{-1}BC^{-1}D$ in form (B.2.19) with a cost of $O(n \log^2 n)$ complexity. Though still complicated, we will show how to combine (B.2.19) with the trace operation to speed up the computation.

B.2.3 Computation of $\mathrm{tr}(\mathrm{A}^{-1}\mathrm{B})$ *And* $\mathrm{tr}(\mathrm{A}^{-1}\mathrm{B}\mathrm{C}^{-1}\mathrm{D})$

Taking the trace of form (B.2.15), we have

$$\mathrm{tr}(A^{-1}B) = \mathrm{tr}(\bar{\mathrm{B}}) + \sum_{i=1}^{\tau} \mathrm{tr}(\bar{U}^{(i)}\bar{V}^{(i)T}). \tag{B.2.20}$$

Further recall that $\bar{U}^{(i)}\bar{V}^{(i)T}$ is a block diagonal low-rank matrix, which can be written as

$$\bar{U}^{(i)}\bar{V}^{(i)T} = \begin{bmatrix} \bar{U}_1^{(i)}\bar{V}_1^{(i)T} & 0 & \cdots & 0 \\ 0 & \bar{U}_2^{(i)}\bar{V}_2^{(i)T} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \bar{U}_{2^{i-1}}^{(i)}\bar{V}_{2^{i-1}}^{(i)T} \end{bmatrix}. \tag{B.2.21}$$

Now using the properties of the trace operator, we have

$$\mathrm{tr}(A^{-1}B) = \mathrm{tr}(\bar{\mathrm{B}}) + \sum_{i=1}^{\tau} \mathrm{tr}(\bar{V}^{(i)T}\bar{U}^{(i)}), \tag{B.2.22}$$

$$= \mathrm{tr}(\bar{\mathrm{B}}) + \sum_{i=1}^{\tau}\sum_{j=1}^{2^i} \mathrm{tr}(\bar{V}_j^{(i)T}\bar{U}_j^{(i)}). \tag{B.2.23}$$

Notice that both $\bar{U}_j^{(i)}, \bar{V}_j^{(i)} \in \mathbb{R}^{n/2^{i-1} \times 2k}$. Computing the product $\bar{V}_j^{(i)T}\bar{U}_j^{(i)}$ can be done in $(8nk^2/2^{i-1})$ time, extracting its trace requires $k-1$ additions, and it is asymptotically negligible. Repeating the calculation for all 2^{i-1} terms at all levels $i = 1, \ldots, \tau$, the total computational cost is $O(nk^2\tau) = O(n \log n)$. Taking into account the cost of producing (B.2.15), the total computational complexity of computing $\mathrm{tr}(A^{-1}B)$ for two HODLR matrices is $O(n\log^2 n)$.

Next we consider taking the trace of (B.2.19). Similarly utilizing the basic trace properties, we can write

$$\begin{aligned}\mathrm{tr}(A^{-1}BC^{-1}D) =&\mathrm{tr}(\tilde{D}) + \sum_{i=1}^{\tau}\sum_{j=1}^{2^i}\mathrm{tr}(\bar{\mathcal{V}}_j^{(i)T}\bar{\mathcal{U}}_j^{(i)}) + \sum_{i=1}^{\tau}\mathrm{tr}(\bar{V}^{(i)T}\bar{D}\bar{U}^{(i)})\\ &+\sum_{i=1}^{\tau}\mathrm{tr}(\bar{\mathbf{V}}^{(i)T}\bar{B}\bar{\mathbf{U}}^{(i)}) + \sum_{i=1}^{\tau}\sum_{j=1}^{\tau}\mathrm{tr}(\bar{\mathbf{V}}^{(j)T}\bar{U}^{(i)}\bar{V}^{(i)T}\bar{\mathbf{U}}^{(j)}). \qquad \text{(B.2.24)}\end{aligned}$$

Since $\tilde{D}$ is of HODLR form, taking its trace is very efficient and can be negligible. Next we consider the second term. Following the same procedure in (B.2.21), we can compute the second term with cost $O(n\log n)$.

Moving to the third and fourth terms, they share the same structure. Take the third term $\sum_{i=1}^{\tau}\mathrm{tr}(\bar{V}^{(i)T}\bar{D}\bar{U}^{(i)})$ as an example. Note $\bar{D}$ is an HODLR matrix and $\bar{U}^{(i)}$, $\bar{V}^{(i)}$ are block-diagonal matrices with 2^{i-1} diagonal blocks of size $(n/2^{i-1})\times 2k$. Therefore the trace operations depends only on the diagonal blocks of $\bar{D}$ at level $i-1$ of size $(n/2^{i-1})\times(n/2^{i-1})$. There are 2^{i-1} of them in total, each having HODLR structure with $\tau - i + 1$ levels:

$$\bar{D}_1^{(i-1)}, \bar{D}_2^{(i-1)}, \cdots, \bar{D}_{2^{i-1}}^{(i-1)}. \qquad \text{(B.2.25)}$$

We then obtain

$$\mathrm{tr}(\bar{V}^{(i)T}\bar{D}\bar{U}^{(i)}) = \sum_{j=1}^{2^{i-1}}\mathrm{tr}(\bar{V}_j^{(i)T}\bar{D}_j^{(i-1)}\bar{U}_j^{(i)}). \qquad \text{(B.2.26)}$$

The computations can be conducted efficiently by computing $\bar{D}_j^{(i-1)}\bar{U}_j^{(i)}$ first which can be done by HODLR-vector product. Similar operations have been analyzed in detail in B.2.1. The sum-total complexity for all 2^{i-1} terms is upper bounded by $O(nk^2\tau)$ and each product is of size $(n/2^{i-1}) \times 2k$. Then we can left-multiply $\bar{V}_j^{(i)T}$ with the result via regular matrix-matrix multiplication. The sum-total complexity is $O(nk^2)$ for all 2^{i-1} terms. Each product is now of size $2k \times 2k$ and the complexity of the trace operation is $O(k2^{i-1})$, which thus can be ignored. Repeating the same operations for all τ terms in $\sum_{i=1}^{\tau} \operatorname{tr}(\bar{V}^{(i)T}\bar{D}\bar{U}^{(i)})$, the total complexity is given by $O(nk^2\tau^2) = O(n\log^2 n)$. The fourth term $\sum_{i=1}^{\tau} \operatorname{tr}(\bar{\mathbf{V}}^{(i)T}\bar{B}\bar{\mathbf{U}}^{(i)})$ can be computed similarly with the same scaling.

For the last term, we compute $\bar{\mathbf{V}}^{(j)T}\bar{U}^{(i)}$ and $\bar{V}^{(i)T}\bar{\mathbf{U}}^{(j)}$ for each pair of (i,j) separately. By applying blockwise matrix multiplications, both can be computed in $O(nk^2)$ time. Now we analyze their output matrices to conduct the following operations.

If $i \geq j$, the product $\bar{\mathbf{V}}^{(j)T}\bar{U}^{(i)}$ is a block-diagonal matrix with 2^{j-1} diagonal blocks. Each diagonal block is of size $k \times 2^{(i-j)}k$. $\bar{V}^{(i)T}\bar{\mathbf{U}}^{(j)}$ is also a block-diagonal matrix with 2^{j-1} diagonal blocks. Its block has size $2^{(i-j)}k \times k$. In this case we compute their product directly by multiplying the corresponding diagonal blocks. Each pair of diagonal blocks takes $O(2^{(i-j)}k^3)$ time. In total, all pairs take $O(2^i k^3)$ time.

If $j \geq i$, we swap the order of the two matrices in the trace, i.e. $\operatorname{tr}(\bar{\mathbf{V}}^{(j)T}\bar{U}^{(i)}\bar{V}^{(i)T}\bar{\mathbf{U}}^{(j)}) = \operatorname{tr}(\bar{V}^{(i)T}\bar{\mathbf{U}}^{(j)}\bar{\mathbf{V}}^{(j)T}\bar{U}^{(i)})$. Now similarly, $\bar{V}^{(i)T}\bar{\mathbf{U}}^{(j)}$ is block-diagonal with 2^{i-1} blocks. Each block is of size $k \times 2^{(j-i)}k$. $\bar{\mathbf{V}}^{(j)T}\bar{U}^{(i)}$ is block-diagonal with 2^{i-1} blocks. Each block is of size $2^{(j-i)}k \times k$. Now we multiply all the pairs of diagonal blocks. The total cost is $O(2^j k^3)$.

Combining two cases together, the cost of computing the block-diagonal matrix inside the trace operator takes $O(2^{\max(i,j)}k^3)$ time. The complexity for the following trace operations can be ignored. In total, evaluating the last term takes

$$\sum_{i=1}^{\tau}\sum_{j=1}^{\tau} O(nk^2 + k^3 2^{\max(i,j)}) = O(nk^2\tau^2 + 2^\tau k^3 \tau) = O(n\log^2 n). \tag{B.2.27}$$

In summary, the total cost of evaluating (B.2.24) is $O(n\log^2 n)$. B.2.1, B.2.2, B.2.3 streamlined an exact approach of evaluating operations of form $\operatorname{tr}(\mathrm{A}^{-1}\mathrm{B})$ and $\operatorname{tr}(\mathrm{A}^{-1}\mathrm{B}\mathrm{C}^{-1}\mathrm{D})$

for HODLR matrices. Given the HODLR form, both operations take $O(n \log^2 n)$ time and an extra memory of $O(n \log n)$.

B.3 HODLR factorization

HODLR matrices admit several fast factorization algorithms. Particularly, factorization of form (3.3.3) is extensively used in this work. Here we use an example to explain the algorithm in detail and more importantly, establish a relationship between HODLR form (3.3.2) and its block-diagonal factors in (3.3.3). We use the same example of a 2-level HODLR matrix as in (3.3.2),

$$A = \begin{bmatrix} \begin{bmatrix} A_1^{(2)} & W_1^{(2)} X_1^{(2)T} \\ X_1^{(2)} W_1^{(2)T} & A_2^{(2)} \end{bmatrix} & W_1^{(1)} X_1^{(1)T} \\ X_1^{(1)} W_1^{(1)T} & \begin{bmatrix} A_3^{(2)} & W_2^{(2)} X_2^{(2)T} \\ X_2^{(2)} W_2^{(2)T} & A_4^{(2)} \end{bmatrix} \end{bmatrix}, \tag{B.3.1}$$

where diagonal blocks $A_i^{(2)}$ in the leaf level (level 2) are assumed to be dense. The first step is to factor out the dense leaf blocks from A. For example we can factor out the leaf blocks from the left. Let us denote

$$\bar{A} = \begin{bmatrix} A_1^{(2)} & 0 & 0 & 0 \\ 0 & A_2^{(2)} & 0 & 0 \\ 0 & 0 & A_3^{(2)} & 0 \\ 0 & 0 & 0 & A_4^{(2)} \end{bmatrix}, \tag{B.3.2}$$

then we can compute blockwisely by

$$A = \bar{A} \begin{bmatrix} \begin{bmatrix} I_{n/4} & \bar{W}_1^{(2)} X_1^{(2)T} \\ \bar{X}_1^{(2)} W_1^{(2)T} & I_{n/4} \end{bmatrix} & \bar{W}_1^{(1)} X_1^{(1)T} \\ \bar{X}_1^{(1)} W_1^{(1)T} & \begin{bmatrix} I_{n/4} & \bar{W}_2^{(2)} X_2^{(2)T} \\ \bar{X}_2^{(2)} W_2^{(2)T} & I_{n/4} \end{bmatrix} \end{bmatrix}, \tag{B.3.3}$$

where $I_{n/4}$ denotes identity matrix of size $n/4 \times n/4$. (B.3.3) can be obtained by updating $\bar{W}_1^{(2)} = \left(A_1^{(2)}\right)^{-1} W_1^{(2)}$, $\bar{X}_1^{(2)} = \left(A_2^{(2)}\right)^{-1} X_1^{(2)}$, $\bar{W}_2^{(2)} = \left(A_3^{(2)}\right)^{-1} W_2^{(2)}$, $\bar{X}_2^{(2)} = \left(A_4^{(2)}\right)^{-1} X_2^{(2)}$ and

$$\bar{W}_1^{(1)} = \begin{bmatrix} \left(A_1^{(2)}\right)^{-1} & 0 \\ 0 & \left(A_2^{(2)}\right)^{-1} \end{bmatrix} W_1^{(1)}, \ \bar{X}_1^{(1)} = \begin{bmatrix} \left(A_3^{(2)}\right)^{-1} & 0 \\ 0 & \left(A_4^{(2)}\right)^{-1} \end{bmatrix} X_1^{(1)}.$$

We note that there is an explicit relationship between off-diagonal blocks and low-rank updates. Take the upper left block of (B.3.3) as an example, we can extract the off-diagonal blocks by writing

$$\begin{bmatrix} I_{n/4} & \bar{W}_1^{(2)} X_1^{(2)T} \\ \bar{X}_1^{(2)} W_1^{(2)T} & I_{n/4} \end{bmatrix} = I_{n/2} + \begin{bmatrix} \bar{W}_1^{(2)} & 0 \\ 0 & \bar{X}_1^{(2)} \end{bmatrix} \begin{bmatrix} 0 & X_1^{(2)T} \\ W_1^{(2)T} & 0 \end{bmatrix}, \tag{B.3.4}$$

$$= I_{n/2} + U_1^{(2)} V_1^{(2)T}, \tag{B.3.5}$$

where $U_1^{(2)}$ and $V_1^{(2)}$ are both of size $n/2 \times 2k$ and defined by

$$U_1^{(2)} = \begin{bmatrix} \bar{W}_1^{(2)} & 0 \\ 0 & \bar{X}_1^{(2)} \end{bmatrix}, \ V_1^{(2)} = \begin{bmatrix} 0 & X_1^{(2)} \\ W_1^{(2)} & 0 \end{bmatrix}. \tag{B.3.6}$$

That is, if one matrix has off-diagonal blocks of rank k, we can extract the off-diagonal blocks and represent them as a rank-$2k$ update to its diagonal blocks. Similarly we can write

$$\begin{bmatrix} I_{n/4} & \bar{W}_2^{(2)} X_2^{(2)T} \\ \bar{X}_2^{(2)} W_2^{(2)T} & I_{n/4} \end{bmatrix} = I_{n/2} + \begin{bmatrix} \bar{W}_2^{(2)} & 0 \\ 0 & \bar{X}_2^{(2)} \end{bmatrix} \begin{bmatrix} 0 & X_2^{(2)T} \\ W_2^{(2)T} & 0 \end{bmatrix}, \tag{B.3.7}$$

$$= I_{n/2} + U_2^{(2)} V_2^{(2)T}. \tag{B.3.8}$$

Now we can further factor out (B.3.5) and (B.3.8) from left of the second term in (B.3.3) and we can further rewrite

$$A = \bar{A} \begin{bmatrix} I_{n/2} + U_1^{(2)} V_1^{(2)T} & 0 \\ 0 & I_{n/2} + U_2^{(2)} V_2^{(2)T} \end{bmatrix} \tag{B.3.9}$$

$$\begin{bmatrix} I_{n/2} & \left(I_{n/2} + U_1^{(2)} V_1^{(2)T}\right)^{-1} \bar{W}_1^{(1)} X_1^{(1)T} \\ \left(I_{n/2} + U_2^{(2)} V_2^{(2)T}\right)^{-1} \bar{X}_1^{(1)} W_1^{(1)T} & I_{n/2} \end{bmatrix}, \tag{B.3.10}$$

$$= \bar{A} \left(I + U^{(2)} V^{(2)T}\right) \begin{bmatrix} I_{n/2} & \tilde{W}_1^{(1)} X_1^{(1)T} \\ \tilde{X}_1^{(1)} W_1^{(1)T} & I_{n/2} \end{bmatrix}. \tag{B.3.11}$$

by updating $\tilde{W}_1^{(1)} = \left(I_{n/2} + U_1^{(2)} V_1^{(2)T}\right)^{-1} \bar{W}_1^{(1)}$, $\tilde{X}_1^{(1)} = \left(I_{n/2} + U_2^{(2)} V_2^{(2)T}\right)^{-1} \bar{X}_1^{(1)}$. This step can be computed efficiently via the Woodbury identity. We denote

$$\left(I + U^{(2)} V^{(2)T}\right) = \begin{bmatrix} I_{n/2} + U_1^{(2)} V_1^{(2)T} & 0 \\ 0 & I_{n/2} + U_2^{(2)} V_2^{(2)T} \end{bmatrix}. \tag{B.3.12}$$

Using the same trick, we can write the last term of (B.3.11) as low-rank update,

$$\begin{bmatrix} I_{n/2} & \tilde{W}_1^{(1)} X_1^{(1)T} \\ \tilde{X}_1^{(1)} W_1^{(1)T} & I_{n/2} \end{bmatrix} = I_n + \begin{bmatrix} \tilde{W}_1^{(1)} & 0 \\ 0 & \tilde{X}_1^{(1)} \end{bmatrix} \begin{bmatrix} 0 & X_1^{(1)T} \\ W_1^{(1)T} & 0 \end{bmatrix}, \tag{B.3.13}$$

$$= I_n + U^{(1)} V^{(1)T}. \tag{B.3.14}$$

Put everything together, we have

$$A = \bar{A} \left(I + U^{(2)} V^{(2)T} \right) \left(I + U^{(1)} V^{(1)T} \right), \tag{B.3.15}$$

where $\bar{A}$ is a block-diagonal matrix containing all leaf level blocks of A. $\left(I + U^{(1)} V^{(1)T} \right)$ and $\left(I + U^{(2)} V^{(2)T} \right)$ are also two block-diagonal matrices, for which each diagonal block is a rank-$2k$ update to identity. For a more general τ-level HODLR matrix, we can extend the algorithm to all levels to get

$$A = \bar{A}(I + U^{(\tau)} V^{(\tau)T}) \cdots (I + U^{(1)} V^{(1)T}), \tag{B.3.16}$$

which is exactly (B.2.1). Each term $(I+U^{(i)}V^{(i)T})$ is block-diagonal matrix with 2^{i-1} blocks where each diagonal block is a rank-$2k$ update to identity. For more details and complexity analysis, we refer readers to [5] for a complete discussion.

APPENDIX C

SUPPLEMENT TO CHAPTER 4

C.1 Soft committor functions

Consider the optimization problem (4.2.7). We offer a probabilistic interpretation of the optimizer q, which we call a 'soft committor function.' This probabilistic interpretation will justify its use qualitatively and quantitatively as a proxy for the committor function.

Define

$$f := \frac{\rho \cdot p_A}{\beta \cdot p}, \quad g := \frac{\rho \cdot p_B}{\beta \cdot p},$$

and note that the Euler-Lagrange equation associated to (4.2.7) then reads as

$$-\beta^{-1}\Delta q(\boldsymbol{x}) + \nabla V(\boldsymbol{x}) \cdot \nabla q(\boldsymbol{x}) + (f(\boldsymbol{x}) + g(\boldsymbol{x}))\, q(\boldsymbol{x}) = g(\boldsymbol{x}). \tag{C.1.1}$$

We shall now rederive this PDE (C.1.1) via a probabilistic construction. In particular this construction will imply that the solution q satisfies $0 \leq q(\boldsymbol{x}) \leq 1$ for all $\boldsymbol{x} \in \Omega$.

Consider a stochastic process $\boldsymbol{X}_t$ that modifies the standard overdamped Langevin diffusion

$$d\boldsymbol{X}_t = -\nabla V(\boldsymbol{X}_t)\, dt + \sqrt{2\beta^{-1}}\, d\boldsymbol{W}_t,$$

where $\boldsymbol{W}_t$ is a Wiener processs, by adding jumps to one of *two* possible 'cemetery states' c_A, c_B with state-dependent rates specified by f, g, respectively.

In other words, we view $\boldsymbol{X}_t \in \Omega \cup \{c_A\} \cup \{c_B\}$ as the continuous-time limit of the discrete-time Markov chain (also denoted $\boldsymbol{X}_t$, abusing notation slightly) defined by the update

$$\boldsymbol{X}_{t+\Delta t} = \begin{cases} c_A, & \text{if } U_t^A < f(\boldsymbol{X}_t)\,\Delta t, \\ c_B, & \text{otherwise if } U_t^B < g(\boldsymbol{X}_t)\,\Delta t\,, \\ \boldsymbol{X}_t - \nabla V(\boldsymbol{X}_t)\,\Delta t + \sqrt{2\beta^{-1}\Delta t}\,\boldsymbol{Z}_t & \text{otherwise,} \end{cases}$$

whenever $\boldsymbol{X}_t \in \Omega$. Here the U_t^A, U_t^B are i.i.d. uniformly distributed random variables on $[0, 1]$, and the $\boldsymbol{Z}_t$ are i.i.d. standard Gaussian random variables. In other words, at each time step the stochastic process is sent to the cemetery state c_A with probability $f(\mathbf{X}_t)\,\Delta t$ and the cemetery state c_B with probability $g(\mathbf{X}_t)\,\Delta t$, else it is advanced by the usual overdamped Langevin dynamics. Note that in the limit of Δt small, it is unlikely for both $U_t^A < f(\boldsymbol{X}_t)\,\Delta t$ and $U_t^B < g(\boldsymbol{X}_t)\,\Delta t$ to hold. Specifically, the probability of this is only $O(\Delta t^2)$ and does not affect the continuous-time limit.

Moreover, if $\boldsymbol{X}_t = c_A$ (resp., c_B), then we define $\boldsymbol{X}_{t+\Delta t} = c_A$ (resp., c_B) deterministically. Then define the stopping time τ as

$$\tau = \inf\left\{t \,:\, \boldsymbol{X}_t \in \{c_A, c_B\}\right\},$$

and define $q : \Omega \to [0, 1]$ by

$$q(\boldsymbol{x}) = \mathbb{P}\left(\boldsymbol{X}_\tau = c_B \,|\, \boldsymbol{X}_0 = \boldsymbol{x}\right).$$

We claim that q so defined (in the continuous-time limit $\Delta t \to 0$) satisfies the PDE (C.1.1). Note that this construction of q coincides with the usual probabilistic construction for the committor function, modulo a change in the underlying stochastic process. We justify the claim only formally. A rigorous argument can be made by analogy to arguments made for the ordinary committor function [25].

C.1.1 Formal PDE derivation

The PDE is derived by conditioning on $\boldsymbol{X}_0 = \boldsymbol{x}$, and writing

$$\begin{aligned} q(\boldsymbol{x}) =& \mathbb{P}(\boldsymbol{X}_\tau = c_B) \\ =& \mathbb{P}(\boldsymbol{X}_{\Delta t} = c_B) + \mathbb{P}(\boldsymbol{X}_{\Delta t} \notin \{c_A, c_B\})\, \mathbb{E}\left[\mathbb{P}\left(\boldsymbol{X}'_\tau = c_B \,|\, \boldsymbol{X}'_0 = \boldsymbol{X}_{\Delta t}\right)\right] + O(\Delta t^2), \end{aligned}$$

where $\boldsymbol{X}'_t$ is an indepenent dummy stochastic process with the same law as $\boldsymbol{X}_t$. But then

$$q(\boldsymbol{x}) = \mathbb{P}(\boldsymbol{X}_{\Delta t} = c_B) + \mathbb{P}(\boldsymbol{X}_{\Delta t} \notin \{c_A, c_B\})\, \mathbb{E}\left[q(\boldsymbol{X}_{\Delta t})\right] + O(\Delta t^2).$$

Expanding further we obtain

$$\begin{aligned} q(\boldsymbol{x}) =& g(\boldsymbol{x})\, \Delta t + \ (1 - f(\boldsymbol{x})\, \Delta t - g(\boldsymbol{x})\, \Delta t)\ \mathbb{E}\left[q\left(\boldsymbol{x} - \nabla V(\boldsymbol{x})\, \Delta t + \sqrt{2\beta^{-1}\Delta t}\boldsymbol{Z}_0\right)\right] \\ &+ O(\Delta t^2). \end{aligned} \tag{C.1.2}$$

Then we can expand q via Taylor expansion:

$$\begin{aligned} q\left(\boldsymbol{x} - \nabla V(\boldsymbol{x})\, \Delta t + \sqrt{2\beta^{-1}\Delta t}\boldsymbol{Z}_0\right) =& q(\boldsymbol{x}) - \nabla V(\boldsymbol{x}) \cdot \nabla q(\boldsymbol{x})\, \Delta t + \sqrt{2\beta^{-1}\Delta t}\, \boldsymbol{Z}_0 \cdot \nabla V(\boldsymbol{x}) \\ &+ \beta^{-1} \Delta t\, \boldsymbol{Z}_0^\top \nabla^2 q(\boldsymbol{x}) \boldsymbol{Z}_0 + o(\Delta t), \end{aligned}$$

from which we obtain

$$\begin{aligned} \mathbb{E}\left[q\left(\boldsymbol{x} - \nabla V(\boldsymbol{x})\, \Delta t + \sqrt{2\beta^{-1}\Delta t}\boldsymbol{Z}_0\right)\right] =& q(\boldsymbol{x}) + \left(-\nabla V(\boldsymbol{x}) \cdot \nabla q(\boldsymbol{x}) + \beta^{-1} \Delta q(\boldsymbol{x})\right) \Delta t \\ &+ o(\Delta t). \end{aligned}$$

It follows from plugging into (C.1.2) that

$$q(\boldsymbol{x}) = q(\boldsymbol{x}) + \Big[g(\boldsymbol{x}) - \nabla V(\boldsymbol{x}) \cdot \nabla q(\boldsymbol{x}) + \beta^{-1}\Delta q(\boldsymbol{x}) - (f(\boldsymbol{x}) + g(\boldsymbol{x}))q(\boldsymbol{x})\Big]\,\Delta t + o(\Delta t).$$

Cancelling $q(\boldsymbol{x})$ from boths sides, dividing by Δt, and taking the limit as $\Delta t \to 0$, we obtain precisely (C.1.1), as desired.

C.2 Discretization of the Ginzburg-Landau density

In this section, we show that one can approximate the equilibrium distribution of the Ginzburg-Landau potential as (4.3.7) via the eigenfunctions of a certain kernel.

The computation of the committor function for the Ginzburg-Landau potential requires the numerical approximation of the operator $\mathcal{H} : L^2([-R,R])^d \to \mathbb{R}$ defined by

$$\begin{aligned}\mathcal{H}[\phi_1,\ldots,\phi_d] =& c_\lambda \int_{-R}^{R} \cdots \int_{-R}^{R} K(0,x_1)\,K(x_1,x_2)\,K(x_2,x_3)\,\ldots\,K(x_{d-1},x_d) \\ &\times K(x_d,0)\,\phi_1(x_1)\ldots\phi_d(x_d)\,dx_1\,\ldots\,dx_d,\end{aligned}$$

where

$$K(x,y) = e^{-\frac{1}{8\lambda}(1-x^2)^2} e^{\frac{\lambda}{2h^2}(x-y)^2} e^{-\frac{1}{8\lambda}(1-y^2)^2},$$

$c_\lambda = e^{-\frac{1}{4\lambda}}$, and R is some fixed positive constant.

Considering the operator $\mathcal{H}_0 : L^2([-R,R]) \to L^2([-R,R])$ defined by

$$\mathcal{H}_0[\phi](x) = \int_\Omega K(x,y)\,\phi(y)\,dy,$$

we observe that it is compact, symmetric, and positive semi-definite. In particular, it has an eigendecomposition consisting of a countable basis of orthonormal eigenfunctions $u_1, u_2, \ldots \in L^2([-R,R])$, together with corresponding non-negative eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \geq$

$\ldots \geq 0$, such that

$$\int_{-R}^{R} K(x,y)\,\phi(y)\,dy = \sum_{j=1}^{\infty} \lambda_j u_j(x) \int_{-R}^{R} u_j(y)\,\phi(x)\,dy$$

for all $\phi \in L^2([-R,R])$. Moreover, for the kernel K defined above it is easily shown that $\lambda_j \in o(e^{-\alpha j})$ for any $\alpha > 0$ as $j \to \infty$ (see [121] for example). We note that the implicit constant in the previous estimate will depend on α but not on j, and increases rapidly as $\alpha \to \infty$. For convenience, let us define the re-scaled eigenfunctions $v_i = \sqrt{\lambda_i} u_i$. Upon substitution of the eigendecomposition of $\mathcal{H}_0$ into the definition of $\mathcal{H}$, we obtain

$$\mathcal{H}[\phi_1, \ldots, \phi_d] = c_\lambda \sum_{j_0,\ldots,j_d=1}^{\infty} v_{j_0}(0) A_{j_0,j_1}[\phi_1] \cdots A_{j_{d-1},j_d}[\phi_d] v_{j_d}(0), \tag{C.2.1}$$

where $A_{j,\ell}[\phi] = \int_\Omega v_j(x) v_\ell(x) \phi(x)\,dx$. We observe that

$$|A_{j,\ell}[\phi]| \leq \sqrt{\lambda_j \lambda_\ell} \|\phi\|_{L^\infty},$$

and thus, if the sums in (C.2.1) are truncated at term J, then

$$\begin{aligned} &\left| \mathcal{H}[\phi_1, \ldots, \phi_d] - \mathcal{H}^{(J)}[\phi_1, \ldots, \phi_d] \right| \\ \leq \quad & 2\, c_\lambda \|\phi_1\|_{L^\infty} \cdots \|\phi_d\|_{L^\infty} \sqrt{K(0,0) K^{(j)}(0,0)}\, Tr(\mathcal{H}_0)^{d-\frac{1}{2}} \sqrt{\sum_{j=J+1}^{\infty} \lambda_j} \\ & + (d-1)\, c_\lambda \|\phi_1\|_{L^\infty} \cdots \|\phi_d\|_{L^\infty} K(0,0) \mathrm{Tr}(\mathcal{H}_0)^{d-1} \sum_{j=J+1}^{\infty} \lambda_j, \end{aligned}$$

where $K^{(J)}(0,0) := \sum_{j=J+1}^{\infty} v_j(0)^2 \leq K(0,0)$ and $\mathcal{H}^{(J)}$ denotes the truncation of $\mathcal{H}$. The analyticity of K guarantees that $\sum_{j=J+1}^{\infty} \lambda_j$ go to zero exponentially quickly [121]) and hence for any $\alpha > 0$ there exists a constant $M_{\alpha,\lambda,d,R}$ depending on α, λ, d, and R such that

$$\left|\mathcal{H}[\phi_1,\ldots,\phi_d] - \mathcal{H}^{(J)}[\phi_1,\ldots,\phi_d]\right| \leq M_{\lambda,d,R}\, e^{-\alpha J/2} \|\phi_1\|_{L^\infty} \cdots \|\phi_d\|_{L^\infty},$$

for all $J \geq 1$.

Next, we express the input functions ϕ_i in a basis of (suitably-scaled) Chebyshev polynomials,

$$\phi_i(x) = \sum_{n=0}^{\infty} \phi_{i,n} T_n\left(\frac{x}{R}\right),$$

where T_n is the nth standard Chebyshev polynomial. Let $\phi_i^{(N)}$ denote the Nth order truncation of ϕ_i defined by

$$\phi_i^{(N)}(x) := \sum_{n=0}^{N} \phi_{i,n} T_n\left(\frac{x}{R}\right).$$

Then, substituting these Chebyshev expansions into our expression for $\mathcal{H}$, we find

$$\mathcal{H}[\phi_1,\ldots,\phi_d] = c_\lambda \sum_{j_0,\ldots,j_d=1}^{\infty} \sum_{n_1,\ldots,n_d=1}^{\infty} A^{n_1}_{j_0,j_1} \cdots A^{n_d}_{j_{d-1},j_d} v_{j_0}(0) v_{j_d}(0)\, \phi_{1,n_1} \ldots \phi_{d,n_d}, \quad \text{(C.2.2)}$$

where

$$A^n_{j,\ell} = A_{j,\ell}\left[T_n(x/R)\right].$$

In the following, we assume that $\phi_1, \ldots, \phi_d$ are in $C^{p+1}([-R,R])$ for some fixed integer $p \geq 1$ and set

$$V_p = \max_i \left\| \frac{d^{p+1}}{dx^{p+1}} \phi_i(x/R) \right\|_{L^1([-1,1])}.$$

A standard estimate from approximation theory [185] gives the following bound on the rate of decay of the coefficients of $\phi_1, \ldots, \phi_d$:

$$|\phi_{i,n}| \leq \frac{2V_p}{\pi n(n-1)\cdots(n-p)}.$$

In particular,

$$\sum_{n=N+1}^{\infty} |\phi_{i,n}| \leq \frac{2V_p}{\pi p(N-p)^p}.$$

If the sums over the Chebyshev coefficients (the n indices) in (C.2.2) are truncated at a fixed integer N and the sums over the eigenvalues (the j indices) are truncated at J, then the error is bounded by

$$\begin{aligned} & \left|\mathcal{H}[\phi_1,\ldots,\phi_d] - \mathcal{H}^{(J,N)}[\phi_1,\ldots,\phi_d]\right| \\ \leq\ & M_{\lambda,d,R}\, e^{-\alpha J/2} \|\phi_1\|_{L^\infty} \cdots \|\phi_d\|_{L^\infty} \\ & + \frac{2dc_\lambda V_p}{\pi p(N-p)^p}\left(1+\frac{2V_p}{\pi p(N-p)^p}\right)^{d-1} K(0,0)\,\mathrm{Tr}(\mathcal{H}_0)^d \max_i \|\phi_i\|_{L^\infty}^{d-1}. \end{aligned}$$

Here $\mathcal{H}^{(J,N)}$ denotes the operator obtained by truncating the sums over eigenvalues in $\mathcal{H}$ at J and projecting onto the first $N+1$ terms in the Chebyshev expansions of $\phi_1,\ldots,\phi_d$. This latter projection, along with the eigendecomposition of $\mathcal{H}_0$, can be performed easily on the computer using standard numerical integration. This yields a discrete, finite-dimensional tensor which is the object we use in our approach when approximating $\mathcal{H}$.