Inference, Computation, and Games

Thesis by
Florian Tobias Schäfer

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2021
Defended May 19, 2021

ACKNOWLEDGEMENTS

I first and foremost want to thank Houman Owhadi for being the best advisor I could have hoped for. Houman gave me complete freedom to pursue my academic interests, whether I was collaborating with him or with others. Yet, whenever I needed his advice, he would make time to meet almost immediately. Over the course of my graduate studies, I came to greatly appreciate his openness to asking unconventional questions, his endless optimism, and the integrity and unconditional generosity with which he treats the people around him.

I want to thank my committee members: Anima Anandkumar, Mathieu Desbrun, Peter Schröder, and Joel Tropp, not primarily for serving on my committee, but for teaching, mentorship, and scientific collaborations that enriched my time at Caltech.

The work in this thesis would not have been possible without my great collaborators, especially Jiong Chen, Matthias Katzfuss, Tim Sullivan, and Hongkai Zheng. It would also not have been possible without the amazing Caltech staff, in particular Diana Bohler and Carmen Nemer-Sirois, to whom I owe the privilege of being able to fully concentrate on my studies. I also want to thank Laura Flower Kim and Daniel Yoder from ISP for their tremendous effort to support us international students, in particular during the most recent, tumultuous years.

Through all the high highs and low lows of grad school, the one thing that remained unchanged is how immensely grateful I am for the wonderful friends I have, at Caltech and elsewhere. More than anything else, you make my life worth living!

Finally, I want to thank my family, especially my parents Birgit and Christian, and my sister Valerie. For as long as I can remember, they have encouraged me to develop my own interests and pursue my own path. Without their love and support, I would not be who I am today.

ABSTRACT

In this thesis, we use statistical inference and competitive games to design algorithms for computational mathematics.

In the first part, comprising chapters two through six, we use ideas from Gaussian process statistics to obtain fast solvers for differential and integral equations. We begin by observing the equivalence of conditional (near-)independence of Gaussian processes and the (near-)sparsity of the Cholesky factors of its precision and covariance matrices. This implies the existence of a large class of *dense* matrices with almost *sparse* Cholesky factors, thereby greatly increasing the scope of application of sparse Cholesky factorization. Using an elimination ordering and sparsity pattern motivated by the *screening effect* in spatial statistics, we can compute approximate Cholesky factors of the covariance matrices of Gaussian processes admitting a screening effect in near-linear computational complexity. These include many popular smoothness priors such as the Matérn class of covariance functions. In the special case of Green's matrices of elliptic boundary value problems (with possibly unknown elliptic operators of arbitrarily high order, with possibly rough coefficients), we can use tools from numerical homogenization to prove the exponential accuracy of our method. This result improves the state-of-the-art for solving general elliptic integral equations and provides the first proof of an exponential screening effect. We also derive a fast solver for elliptic partial differential equations, with accuracy-vs-complexity guarantees that improve upon the state-of-the-art. Furthermore, the resulting solver is performant in practice, frequently beating established algebraic multigrid libraries such as AMGCL and Trilinos on a series of challenging problems in two and three dimensions. Finally, for any given covariance matrix, we obtain a closed-form expression for its *optimal* (in terms of Kullback-Leibler divergence) approximate inverse-Cholesky factorization subject to a sparsity constraint, recovering Vecchia approximation and factorized sparse approximate inverses. Our method is highly robust, embarrassingly parallel, and further improves our asymptotic results on the solution of elliptic integral equations. We also provide a way to apply our techniques to sums of independent Gaussian processes, resolving a major limitation of existing methods based on the screening effect. As a result, we obtain fast algorithms for large-scale Gaussian process regression problems with possibly noisy measurements.

In the second part of this thesis, comprising chapters seven through nine, we study continuous optimization through the lens of competitive games. In particular, we consider *competitive optimization*, where multiple agents attempt to minimize conflicting objectives. In the single-agent case, the updates of gradient descent are minimizers of quadratically regularized linearizations of the loss function. We propose to generalize this idea by using the Nash equilibria of quadratically regularized linearizations of the competitive game as updates (*linearize the game*). We provide fundamental reasons why the natural notion of linearization for competitive optimization problems is given by the *multilinear* (as opposed to linear) approximation of the agents' loss functions. The resulting algorithm, which we call *competitive gradient descent*, thus provides a natural generalization of gradient descent to competitive optimization. By using ideas from information geometry, we extend CGD to competitive mirror descent (CMD) that can be applied to a vast range of constrained competitive optimization problems. CGD and CMD resolve the cycling problem of simultaneous gradient descent and show promising results on problems arising in constrained optimization, robust control theory, and generative adversarial networks. Finally, we point out the *GAN-dilemma* that refutes the common interpretation of GANs as approximate minimizers of a divergence obtained in the limit of a fully trained discriminator. Instead, we argue that GAN performance relies on the *implicit competitive regularization* (ICR) due to the simultaneous optimization of generator and discriminator and support this hypothesis with results on low-dimensional model problems and GANs on CIFAR10.

PUBLISHED CONTENT AND CONTRIBUTIONS

[1] Pierre-Luc Bacon, Florian Schäfer, Clement Gehring, Animashree Anandkumar, and Emma Brunskill. A Lagrangian method for inverse problems in reinforcement learning. F.S. contributed to the development of the method.

[2] Jiong Chen, Florian Schäfer, Jin Huang, and Mathieu Desbrun. Multiscale Cholesky preconditioning for ill-conditioned problems. 2021. F.S. contributed to the writing and the development of the method.

[3] Houman Owhadi, Clint Scovel, and Florian Schäfer. Statistical numerical approximation. *Notices of the AMS*, 2019. F.S. contributed to the writing, mostly in the section on the sparse factorization of dense kernel matrices.

[4] Florian Schaefer and Anima Anandkumar. Competitive Gradient Descent. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/56c51a39a7c77d8084838cc920585bd0-Paper.pdf. F.S. is the first author and main contributor of this work.

[5] Florian Schaefer, Hongkai Zheng, and Animashree Anandkumar. Implicit competitive regularization in GANs. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8533–8544. PMLR, 13–18 Jul 2020. URL http://proceedings.mlr.press/v119/schaefer20a.html. F.S. and H.Z. are joint first authors of this work. F.S. primarily contributed to the writing and conceptual/theoretical part of the project.

[6] Florian Schäfer, Anima Anandkumar, and Houman Owhadi. Competitive Mirror Descent. *arXiv preprint arXiv:2006.10179*, 2020. F.S. is the first author and main contributor of this work.

[7] Florian Schäfer, Matthias Katzfuss, and Houman Owhadi. Sparse Cholesky factorization by Kullback-Leibler minimization. *arXiv preprint arXiv:2004.14455*, 2020. F.S. is the first author and main contributor of this work.

[8] Florian Schäfer, T. J. Sullivan, and Houman Owhadi. Compression, Inversion, and Approximate PCA of Dense Kernel Matrices at Near-Linear Computational Complexity. *Multiscale Model. Simul.*, 19(2):688–730, 2021. ISSN 1540-3459. doi: 10.1137/19M129526X. URL https://doi.org/10.1137/19M129526X. F.S. is the first author and main contributor of this work.

[9] Jing Yu, Clement Gehring, Florian Schäfer, and Anima Anandkumar. Robust reinforcement learning: A constrained game-theoretic approach. *To be presented at Learning for Decision and Control (L4DC) 2021.*, 2021. F.S. advised first author J.Y. on the use of CMD and contributed to the writing.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 Numerical algorithms as games and estimators

This thesis studies the following question:

Can we design better numerical algorithms by interpreting computation in terms of statistical inference and game theory?

At face value, this question is somewhat ambiguous: If we apply an algorithm for solving linear systems to a linear system arising from a partial differential equation, what significance does it have that this algorithm was designed "by interpreting computation in terms of statistical inference"? Is it then nothing more than a decoration for results that, while useful, are "just linear algebra"? This justification would be easier if our method returns random results, such as in randomized linear algebra [165], or if it returns a Bayesian posterior of possible solutions, as in probabilistic numerics [116] thus solving a different problem than classical methods. But for the work presented in this thesis, this is not the case, and the resulting algorithms could just as well be characterized in terms of linear algebra and optimization.

But just like game theory and statistical inference, linear algebra and optimization can be seen as mere decoration of operations on a long array of real numbers or, even more reductionist, on a high-dimensional Boolean hypercube. Indeed, the fascination of computational mathematics is that it expresses the vast landscape of mathematics in terms of a small common set of seemingly innocuous operations. Some mathematical concepts in computation, such as asymptotics, oracle models, and real arithmetic, replace the empirical phenomenon of computation with an idealized, more structured one. However, most of them merely re-express the original problem and thus help us to navigate the vast space of possible algorithms in search of solutions to practical problems.

This thesis develops powerful new algorithms based on statistical and game-theoretical perspectives of classical methods in computational mathematics. It thus makes the case to include them in the repertoire of viewpoints that we use to design and reason about algorithms, alongside more traditional ones such as those originating from physics, optimization, and linear algebra.

1.2 Numerical approximation, fast algorithms, and statistical inference

The first part of this thesis, comprising Chapters 3 through 6, is concerned with the interplay of statistical inference, numerical approximation, and fast solvers for partial differential equations.

1.2.1 Learners and Solvers

A fundamental difficulty in computational mathematics is that many if not most mathematical objects are infinite, whereas computation is necessarily finite. A great deal of work is therefore devoted to studying the relationship of the finite operations performed by a computer to the continuous objects they are meant to represent. This problem often occurs in different layers as we are relating arrays of binary states to real numbers, arrays of real-valued coefficients to continuous functions, and evaluations of these functions to their integrals. The fields of numerical stability, approximation theory, and numerical quadrature study the accuracy of these finite-dimensional approximations of continuous objects. Their mostly deterministic nature is surprising since probabilistic modeling and statistical inference are the methods of choice for dealing with uncertain quantities in virtually every other scientific domain.

Indeed, despite being less well-known, the idea of casting computational mathematics as statistical inference predates the development of semiconductor-based computers. As early as 1896, in his course on probability, Henri Poincaré proposes to view the numerical integration of a function as a statistical estimation problem where we try to estimate the integral based on *data* gathered from function evaluations (see [197] for a reprint and [69, Section 2] for a summary in English). Around fifty years later, in the early days of automated computation, [244] suggested the probabilistic modeling of the accumulation of round-off errors during matrix inversion. Following Poincaré's lead, the link between numerical approximation and statistical inference was further explored by Palasti and Renyi [193], Sul'din [231, 232], Sard [213], Kimeldorf and Wahba [138], and Larkin [146].

More recently, these ideas were revisited in the context of information-based complexity [235], Bayesian numerical approximation [69], and Bayesian numerical homogenization [187]. The emerging field of probabilistic numerics [116] advocates for the development of probabilistic analogs of existing numerical methods in order to better quantify the uncertainty of results of numerical computation. Meanwhile, Bayesian optimization [216] has been successfully applied to a wide range of applications. Even when dealing with finite-dimensional problems, such as in the case of numerical linear algebra, the perspective of statistical inference can improve the computational complexity of a task by computing with partial information. [188–190] adopt a decision-theoretic perspective on multigrid methods to develop new classes of fast solvers and operator adapted wavelets, named *gamblets*.

A related but different line of work replaces finite-dimensional function spaces commonly used in the numerical analysis of partial differential with model classes hailing from statistical inference and machine learning, generalizing classical meshfree discretizations such as radial basis function [80, 246, 247] and boundary element methods [214]. The authors of [52] interpret meshless collocation methods as performing Bayesian inference with a prior induced by the interpolation method of choice. Using MCMC sampling techniques, they obtain a general procedure for computing the posterior distributions of nonlinear forward and inverse problems involving partial differential equations. By using a Gauss-Newton method, [47] cast the solution of *nonlinear* partial differential equations as a series of Gaussian process regression problems, solving the nonlinear interpolation problem to high accuracy without requiring the use of sampling techniques. Motivated by the successes of deep learning in other domains, physics-informed neural networks proposed by [200] include the residual of a PDE at a set of points into the loss function of a neural network. The resulting neural network-based collocation method can be used to learn input-output maps of parametric or inverse problems involving PDEs without having access to a dedicated forward operator. It thus forms the bridge to another class of methods that use neural networks to directly learn the solution operators of various PDE-related problems based on training data provided by classical solvers [137, 153]. In order to improve the computational efficiency and generalization performance, a large number of architectures inspired by existing fast solvers or physical intuition have been proposed [77–79, 152].

1.2.2 Contribution in the first part of this thesis

The first part of this thesis is based on probabilistic interpretations of the well-known Cholesky factorization that uses Gaussian elimination to express a positive definite matrix Θ as the square $LL^{\top}$ of a lower triangular matrix L.

Cholesky factorization can be formulated as recursive application of the identity

$$\begin{pmatrix}\Theta_{1,1} & \Theta_{1,2}\\ \Theta_{2,1} & \Theta_{2,2}\end{pmatrix} = \begin{pmatrix}\mathrm{Id} & 0\\ \Theta_{2,1}(\Theta_{1,1})^{-1} & \mathrm{Id}\end{pmatrix}\begin{pmatrix}\Theta_{1,1} & 0\\ 0 & \Theta_{2,2}-\Theta_{2,1}(\Theta_{1,1})^{-1}\Theta_{1,2}\end{pmatrix}\begin{pmatrix}\mathrm{Id} & (\Theta_{1,1})^{-1}\Theta_{1,2}\\ 0 & \mathrm{Id}\end{pmatrix} \tag{1.1}$$

to the Schur complement $\Theta_{2,2} - \Theta_{2,1}\left(\Theta_{1,1}\right)^{-1}\Theta_{1,2}$ obtained at the previous step. In particular, the k-th column of the final Cholesky factor L is a multiple of the first column of the Schur complement when setting $\Theta_{1,1} = \Theta_{1:(k-1),1:(k-1)}$. If Θ is the covariance matrix of the Gaussian vector $X = (X_1, X_2) \sim \mathcal{N}(0, \Theta)$, the well-known identities

$$\mathbb{E}[X_2 \mid X_1 = a] = \Theta_{2,1}(\Theta_{1,1})^{-1}a, \tag{1.2}$$

$$\mathbb{C}\mathrm{ov}[X_2 \mid X_1] = \Theta_{2,2} - \Theta_{2,1}(\Theta_{1,1})^{-1}\Theta_{1,2}\,, \tag{1.3}$$

imply that Cholesky factorization of Θ is equivalent to the iterative conditioning of the Gaussian vector X. In particular, conditional (near-)independence of X implies (near-)sparsity of the Cholesky factors of Θ!

In many Gaussian process models, $\Theta_{ij} = \mathcal{G}\left(x_i, x_j\right)$ is obtained from evaluations of a covariance function in pairs of points $\{x_i\}_{1\leq i\leq N} \subset \mathbb{R}^d$. The *screening effect* [228] predicts that the conditional correlation length of a smooth Gaussian process after conditioning on its value in a few locations is inversely proportional to their density.

We can maximize the conditional independence by choosing conditioning points that are spread out as far as possible and use the screening effect to predict the sparsity set, leading to the following near-linear complexity algorithm for the approximate Cholesky factorization of Θ (See Figure 1.2).

1. Reorder the rows and columns of Θ such that the associated points $x_1 \ldots x_k$ are well spread out for all $k \leq N$.
2. Select a sparsity set according to the predictions of the screening effect.
3. Apply Gaussian elimination restricted to entries of the sparsity set.

As we discuss in Chapter 2, Green's functions of elliptic partial differential equations are a natural choice of covariance functions for smooth Gaussian processes. The screening effect exhibited by these processes is dual to the localization of coarse-grained partial differential operators. In Chapter 4, we use and extend results from [163, 190], to obtain the first rigorous proof of an exponential screening effect for finitely smooth Gaussian processes.

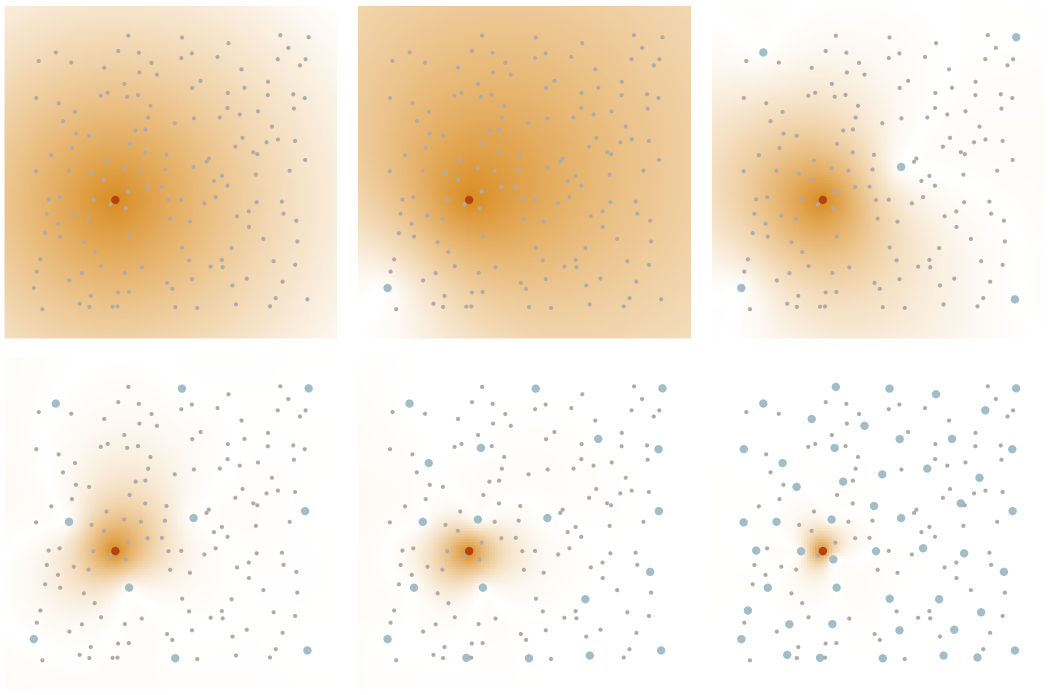


Figure 1.1: **The screening effect.** The length of the conditional correlation of the point in red decreases with the density of the conditioning set.

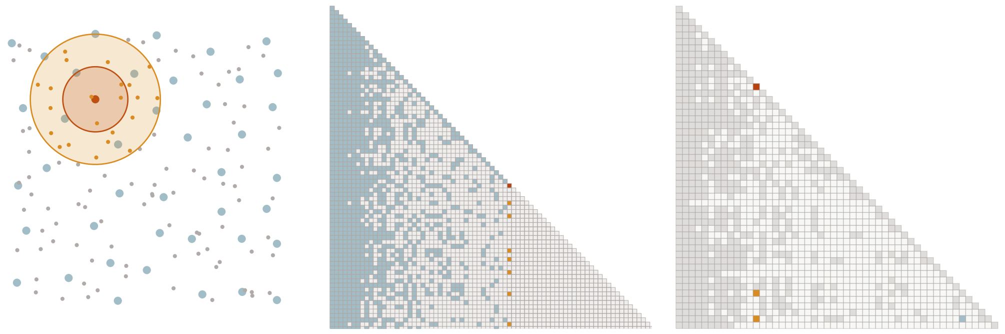


Figure 1.2: **The maximin ordering and sparsity pattern.**We successively select the point x_i that has maximal distance ℓ_i from the points that were selected so far (left). We add entries corresponding to interactions of x_i with points within radius $\rho\ell_i$ to the sparsity set S (middle). When computing Cholesky factors of Θ, we skip updates $\Theta_{kj} \leftarrow \Theta_{kj} - \Theta_{ki}\Theta_{ji}/\Theta_{ii}$ that use entries outside of S (right).

We use these results to prove rigorous accuracy-vs-complexity estimates of algorithms as presented in Figure 1.2, as well as a sparse Cholesky factorization of the precision matrix Θ^{-1}. Thus we obtain algorithms that provably compute ϵ accurate Cholesky factors of Green's matrices of elliptic PDEs and their inverses in complexity $\mathcal{O}\left(N \log^2(N) \log^{2d}(N/\epsilon)\right)$ and $\mathcal{O}\left(N \log^{2d}(N/\epsilon)\right)$, improving the state-of-the-art for fast solvers for general elliptic PDE.

We next show that for a given sparsity pattern S, we can compute the optimal (in Kullback-Leibler divergence) S-sparse approximate Cholesky factor of Θ^{-1} in closed form, from entries of Θ.

$$L = \underset{\hat{L} \text{ that are } S\text{-sparse}}{\arg\min} \mathbb{D}_{\mathrm{KL}}\Big(\mathcal{N}(0, \Theta) \parallel \mathcal{N}(0, (\hat{L}\hat{L}^\top)^{-1})\Big) \Leftrightarrow L_{s_i,i} = \frac{\Theta_{s_i,s_i}^{-1}\mathbf{e}_1}{\sqrt{\mathbf{e}_1^\top \Theta_{s_i,s_i}^{-1}\mathbf{e}_1}}. \tag{1.4}$$

Recovering "Vecchia approximation" [241] and factorized sparse approximate inverses [141], this algorithm is highly stable, almost embarassingly parallel, and further improves the complexity of inverting Θ to $O\left(N \log^{2d}(N/\epsilon)\right)$, matching that of inverting Θ^{-1}. It furthermore provides us with a way to extend screening-based methods to independent sums of Gaussian processes, thus resolving a longstanding and pressing computational problem in spatial statistics.

Finally, we provide efficient implementation to show that our methods are also fast in practice to the point of being competitive with established libraries based on existing methods. For instance, we show that a preconditioner for the solution of elliptic PDEs based on the above work outperforms the established algebraic multigrid implementations of Trilinos and AMGCL on challenging problems arising in two- and three-dimensional linearized elasticity (see Figure 1.3).

1.3 Game theory as a paradigm for algorithm design

The second part of this thesis, comprising Chapters 7 through 9, is concerned with the use of game theory as a guiding principle for the design of new algorithms

1.3.1 From optimization to competitive optimization

Optimization is a powerful paradigm for the design of algorithms. In order to solve a novel problem, we cast it as the minimization of an appropriate cost function and use one of the many existing optimization algorithms. In many branches of continuous optimization, the workhorses of this approach are local iterative methods such as gradient or Newton descent that minimize a series of regularized local approximations. This can be interpreted as an agent that, based on local information, tries to greedily decrease a loss that encodes the original problem.

Instead of a single agent, *Competitive optimization* features a multitude of agents trying to minimize their respective loss, each of which may depend on the actions of all agents.[1] The expressiveness of competitive optimization begs the question, how

[1]Some of the agents may also collaborate, but this thesis focuses on competitive games.

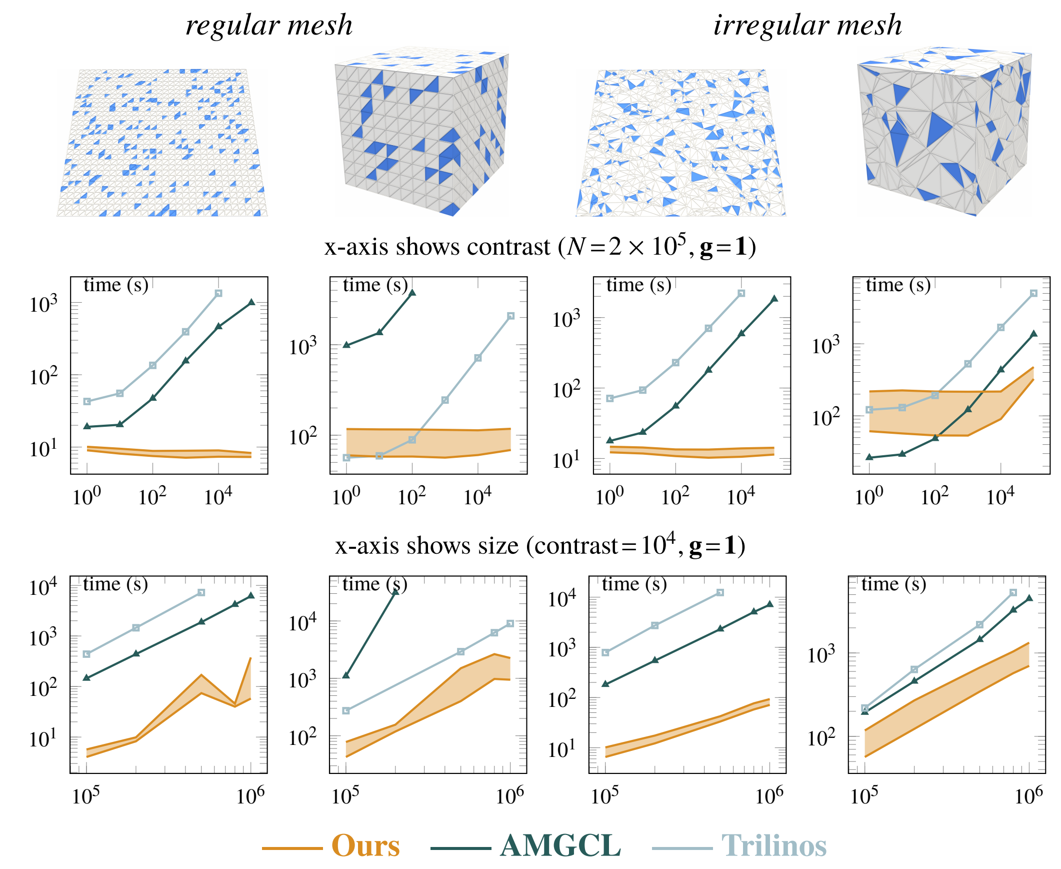


Figure 1.3: **Comparison against AMG.** We compare the solver described in Section 5.6 against state of the art implementations of algebraic multigrid methods on a problem arising in linear elasticity. The shaded region illustrates the performance of our method over different parameter choices.

to cast a given computational problem into a competitive game and for which types of problems competitive optimization offers benefits over classical optimization. A general answer to these questions is presently out of reach. Instead we present four important applications of competitive optimization.

1. Convex duality enables us to write a given convex lower-semicontinuous function $f(x)$ as $f(x) = \max_y \langle x, y \rangle - f^*(y)$, where f^* is the so-called *convex dual* of f. If f^* is better behaved than f it can be beneficial to cast the minimization of f as the competitive game $\min_x \max_y \langle x, y \rangle - f^*(y)$. For instance, the convex function f_0 with $f_0(0) = 0$ and $f_0(x) = \infty$ for $x \neq 0$, has the convex dual $f_0^* \equiv 0$ leading to primal-dual methods for equality constrained optimization.

2. In optimization and statistics, we might aim to be robust to a perturbation w applied to the solution, leading to problem formulations of the form $\min_x \max_w f(x+w)$ [37, 124]. Similar formulations are employed by adversarial training approaches that aim to harden neural network classifiers agains adversarial examples [161].

3. A number of applications of reinforcement learning are competitive games [219, 243], making it paradigm for training policies for these tasks, as well.

4. Recently, multiple self-supervision learning techniques have been implemented as games [157, 194, 196, 242, 245]. In particular, *generative adversarial networks* (GANs) [96] have revolutionized the field of generative modeling in many domains by introducing a competitive game between a generator network that produces new data, and a discriminator network that tries to tell apart real and artificial data.

1.3.2 Contribution in the second part of this thesis

In the second part of this thesis, which comprises Chapters 7 through 9, we propose a natural generalization of gradient descent to constrained and competitive optimization involving two agents. We furthermore provide a novel perspective on the mechanisms that allow GANs to drastically outperform conventional methods for automatic image generation.

Variants of gradient descent (GD) serve as workhorses for numerous applications, including virtually all of deep learning. A seemingly natural generalization of GD to competitive optimization is *simultaneous gradient descent* (SimGD), where each agent performs a step according to the present gradient of its loss function. However, even on a simple bilinear minimax game $\min_x \max_y xy$, this algorithm features oscillatory and divergent behavior as opposed to converging to the Nash equilibrium in $(0, 0)$. This phenomenon, illustrated in Figure 1.4, is the analogue of *"Rock! Paper! Scissors! Rock! Paper! Scissors! Rock! Paper!..."* in the eponymous hand game.

This leads us to question whether SimGD is really the natural generalization of GD to multiple agents and to search for alternatives. Our point of departure is the observation that gradient descent with stepsize η applied to the function $f : \mathbb{R}^m \longrightarrow \mathbb{R}$ can be written as

$$x_{k+1} = \operatorname*{arg\,min}_{x \in \mathbb{R}^m} (x^\top - x_k^\top) \nabla_x f(x_k) + \frac{1}{2\eta} \|x - x_k\|^2. \tag{1.5}$$

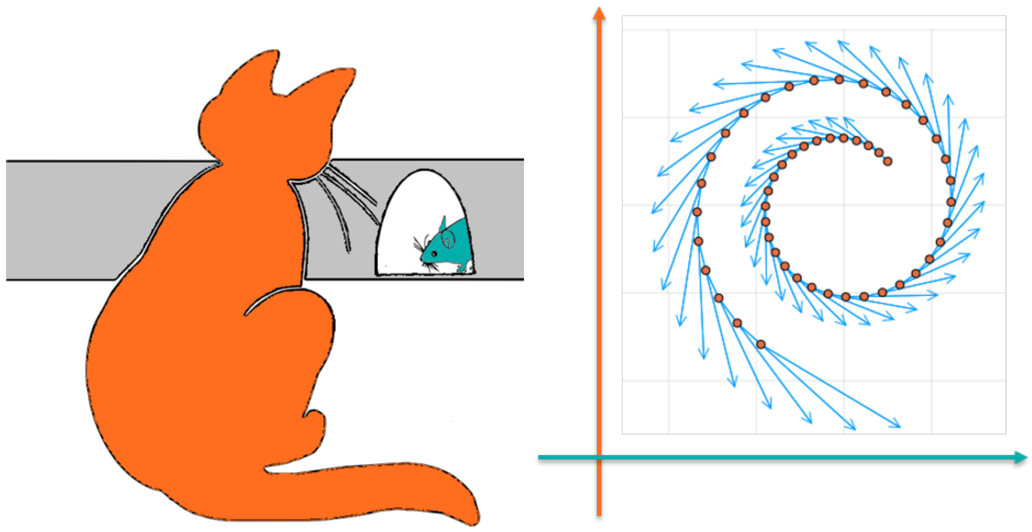


Figure 1.4: **The cycling problem.** The cycling problem of SimGD arises, because each chooses the optimal action according the *last* action of the other agent.

This can be seen as a (single) player solving a local linear approximation of the (minimization) game, subject to a quadratic penalty that expresses her limited confidence in the global accuracy of the model. The natural generalization of this idea to the competitive optimization problem involving two agents x and y minimizing f and g is to have both players solve a local approximation of the true game, each subject to a quadratic penalty that expresses their limited confidence in the accuracy of the local approximation.

If we choose a linear approximation of the loss function, we recover SimGD. Indeed, the cycling behavior of SimGD can be understood as arising from the fact that the resulting local game does not incorporate any interaction between the two agents.

An equally valid generalization of the linear approximation to the setting of two agents is to use a *bilinear* approximation. The bilinear approximation captures some interaction between the two players, hence we argue that the natural generalization of gradient descent to competitive optimization is not SimGD, but rather the update rule $(x_{k+1}, y_{k+1}) = (x_k, y_k) + (x, y)$, where (x, y) is a Nash equilibrium of the game

$$\begin{aligned} &\min_{x \in \mathbb{R}^m} x^\top \nabla_x f + x^\top D_{xy}^2 f y + y^\top \nabla_y f + \frac{1}{2\eta} x^\top x \\ &\min_{y \in \mathbb{R}^n} y^\top \nabla_y g + y^\top D_{yx}^2 g x + x^\top \nabla_x g + \frac{1}{2\eta} y^\top y, \end{aligned} \tag{1.6}$$

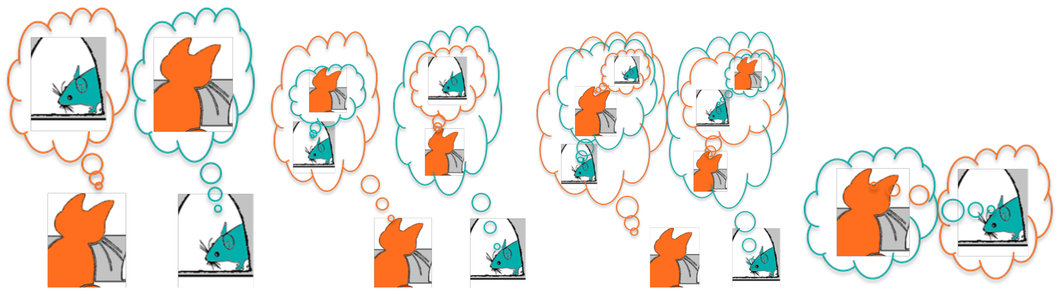


Figure 1.5: **What I think that they think that I think...** The partial sums of a Neumann-series representation of Equation (1.7) represent different orders of opponent-awareness, recovering the Nash-equilibrium in the limit.

that has the closed form solution

$$\begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = - \begin{pmatrix} \mathrm{Id} & \eta D_{xy}^2 f \\ \eta D_{yx}^2 g & \mathrm{Id} \end{pmatrix}^{-1} \begin{pmatrix} \nabla_x f \\ \nabla_y g \end{pmatrix}. \tag{1.7}$$

By using Δx and Δy as update rule, we obtain CGD. We can approximate the matrix inverse in this update rule with a Neumann series to reveal another game-theoretic interpretation. Each partial sum of the Neumann series represents another level of opponent awareness, letting agents choose the optimal action if their opponent stays still, the optimal action if their opponent thinks that they stay still, the optimal ... until the Nash equilibrium is recovered in the limit.

In Chapter 8, we extend CGD to a large class of constrained optimization problems by using the geometric information provided by a Bregman divergence, obtaining competitive mirror descent. Where a naive implementation would lose the computational efficiency of CGD, we propose to play the local game in Equation (1.6) on the *tangent space* of the manifold structure induced by the Bregman divergence. We then perform the update according to its *dual exponential map* that describes an alternative notion of a straight path that incorporates the global geometry of the constraint set. The resulting *competitive mirror descent* (CMD) separates the local linear computation of the Nash equilibrium from the global nonlinear update, thus preserving the computational efficiency of CGD.

In Chapter 9, we study the mechanisms behind the impressive performance of GANs. We begin by casting doubt on the interpretation of GANs as trying to minimize a divergence measure arising in the limit of a "perfect" discriminator. To this end, we point out the *GAN-dilemma* that divides GAN formulations into two classes: those that rely on a notion of a predefined distance between data points and those that do not. In the former case, the discriminator can always become arbitrarily accurate,

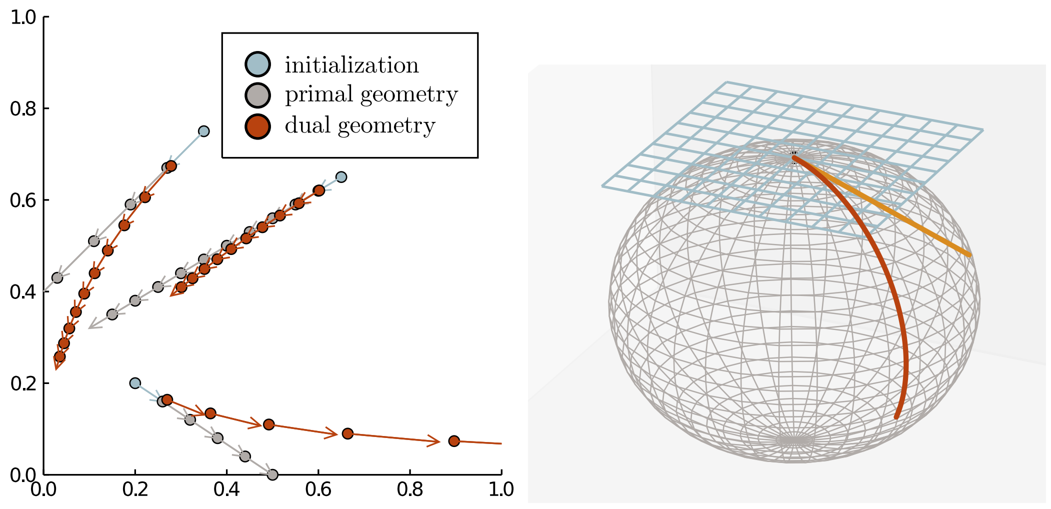


Figure 1.6: **Dual geometry.** (left:) The dual notion of straight line induced by the Shannon entropy guarantees feasible iterates on $\mathbb{R}^m_+$. (right:) A manifold $\mathcal{M}$, tangent space $\mathcal{T}_p\mathcal{M}$, tangent vector $x \in \mathcal{T}_p\mathcal{M}$, and the path described by the exponential map $\{q : q = \operatorname{Exp}_p(tx), \text{for } t \in [0,1]\}$.

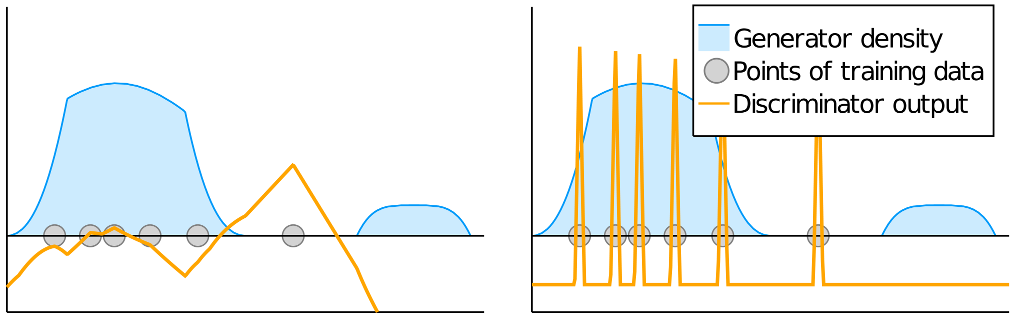


Figure 1.7: **The discriminator can always improve.** We want the discriminator confidence to reflect the relative abundance of true and fake data (left). But by picking out individual data points, the discriminator can almost always achieve arbitrarily low loss on any finite data set (right). Even in the limit of infinite data, the slightest misalignment of the supports of generated and real data can be exploited in a similar way.

rendering the divergence minimization point of view meaningless (See Figure 1.7). In the second case, the results are strongly determined by the choice of distance function. Most present methods implicitly use the pixel-wise mean-square distance on images, which is a *terrible* notion of image similarity (see Figure 1.8).

We attempt to resolve the GAN-dilemma by arguing that GAN performance arises from the *implicit competitive regularization* (ICR) due to the dynamics of simultaneous training. We support our claims with a range of experiments on toy problems

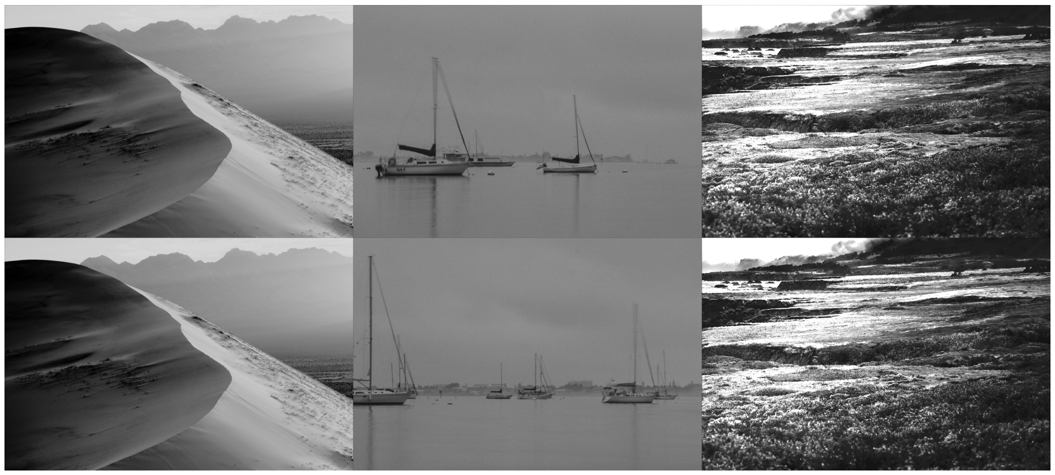


Figure 1.8: **The Euclidean distance is not perceptual.** We would like to challenge the reader to order the above three pairs of images according to the Euclidean distance of their representation as vectors of pixel-intensities.[2]

and real GANs. In particular, we observe that a CGD-trained GAN achieves higher image quality on CIFAR10 (measure by inception score and Frechet inception distance) than explicitly regularized GANs trained with Adam. Since CGD improves the convergence properties around points that are stabilized by ICR, we interpret this as additional evidence that ICR is instrumental for GAN performance.

[2]The pairs of images are ordered from left to right and in increasing order of distance. The first pair is identical, while the third pair differs by a tiny warping.

Chapter 2

ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS AND SMOOTH GAUSSIAN PROCESSES

2.1 Linear Elliptic Partial Differential Equations

2.1.1 Local laws for global phenomena

Our quantitative understanding of the physical world leans heavily on the observation that even the most complicated macroscopic phenomena can be characterized by simple microscopic laws. The macroscopic evolution of the temperature distribution jet engine might be complicated, yet the change of temperature is proportional to the difference between its present temperature and the mean temperature in its neighborhood. Similarly, the colonization of a petri dish by bacteria might seem hard to describe globally, but microscopically the growth rate at each point can be related to the concentration of nutrients.

The expression of complicated global phenomena in terms of simpler local laws is not restricted to physical sciences, either. While finding the dist (P, Q) between two points P and Q involves a global optimization problem, it is easy to derive a relation between $\operatorname{dist}(P, Q)$ and $\operatorname{dist}(\tilde{P}, Q)$ for any point $\tilde{P}$ neighboring P. Similarly, properties of a random walk starting in P can be expressed in terms of random walks starting in points neighboring P.

The discovery of simple local laws governing complicated global phenomena is a key component in most quantitative fields of science. The theory of partial differential equations provides a general framework to formalize these laws and study the properties of the objects they describe.

2.1.2 Partial differential equations (PDEs)

Instead of looking for laws defined in terms of small "neighborhoods" of finite size, one can pass to "infinitely small neighborhoods" and obtain laws given in terms of derivatives. For instance, the difference between the value $u(x)$ of a function in x and its mean in a small ball of radius h around x is proportional to the sum of its second derivatives:

$$\frac{\int_{B_h(x)} u(y) \, \mathrm{d}y}{\int_{B_h(x)} 1 \, \mathrm{d}y} - u(x) \approx \sum_i \partial_i \partial_i u(x) = \Delta u(x). \tag{2.1}$$

The observation that the rate of change in the temperature in a point is proportional to the difference of its present temperature, and the mean temperature in nearby points then leads us to the *heat* or *diffusion* equation

$$\partial_t u(x,t) = \Delta u(x,t). \tag{2.2}$$

More generally, a functional equation given by point-wise equalities involving any combination of partial derivatives of a function is called a partial differential equation (PDE). In light of the vast range of phenomena described by PDEs, it is not surprising that a "general theory of PDEs" is not available. Instead, a variety of classifications exist that identify families of PDEs for which systematic theories can be developed.

2.1.3 The Poisson problem

This thesis is concerned with the class of linear elliptic partial differential equations that has the Poisson equation as its most well-known representative

$$-\Delta u(x) = f(x). \tag{2.3}$$

Following Equation 2.1, this means that we are looking for a function that, at each point x, differs from its local average by $f(x)$. This can be viewed as an equilibration phenomenon where a physical quantity tends to be distributed homogeneously in space, bar the effect of external perturbations or forces acting on the system. Thus, the right-hand side of Equation 2.3 is often called the *forcing term.*

For instance, Equation 2.3 governs the electrostatic potential u in a medium in a charge density given by f. In the absence of electric charges, the electric potential at each point is equal to its local average in a small neighborhood. In the presence of a positive (negative) charge, it is be larger (smaller) than its local average.

In the case of the incompressible Navier-Stokes equations, if an outside force f acts on the fluid with velocity function u, the pressure p is given by the Poisson equation

$$-\Delta p(x) = -\operatorname{div}(f(x)) + [Du(x)] : [Du(x)]^\top, \tag{2.4}$$

where " : " denotes the scalar product between the Jacobian of the velocity field $Du(x)$ and its transpose. Thus, the difference between the pressure at a given point in the fluid and the average in its local neighborhood is equal to the net inbound forces exerted by f and the movement of the surrounding fluid. In the absence of net inbound or outbound forces, the pressure equilibrates to equal its local average.

The Poisson problem is also intimately related to the heat equation

$$\frac{\partial u}{\partial t} = \Delta u \tag{2.5}$$

and the wave equation

$$\frac{\partial^2 u}{\partial t^2} = \Delta u. \tag{2.6}$$

Thus, the updates of implicit time-stepping schemes for these equations and their steady-state solutions can be obtained as the solution of a Poisson problem.

2.1.4 The Dirichlet energy

For a domain Ω, we define the Dirichlet energy associated to the forcing term f as

$$\mathcal{D}(u, f) := \int_{\Omega} \frac{1}{2}\|Du(x)\|^2 - f(x)u(x)\,\mathrm{d}x. \tag{2.7}$$

By formally computing the derivative with respect to u in the direction given by v that is equal to zero on $\partial\Omega$ and using the divergence theorem, we obtain

$$\frac{\partial \mathcal{D}(u+v, f)}{\partial v}\mathcal{D}(u, f) \tag{2.8}$$

$$= \int_{\Omega} \langle Du(x), Dv(x)\rangle - f(x)v(x)\,\mathrm{d}x \tag{2.9}$$

$$= \int (-\Delta u(x) - f(x))\, v(x)\,\mathrm{d}x. \tag{2.10}$$

Thus, for a function g specifying values on the boundary $\partial\Omega$ of Ω, critical points of the minimization problem

$$\min_{u : u = g \text{ on } \partial\Omega} \mathcal{D}(u, f) \tag{2.11}$$

are characterized as solutions of the Poisson problem in Equation 2.3, with boundary values given by g. This provides another perspective on the Poisson problem: In the absence of forcing ($f = 0$), it amounts to finding functions with prescribed boundary values that are varying as slowly as possible in space. Otherwise, the Poisson problem characterizes an optimal trade-off between the contradictory objectives of spatial regularity, a large L^2 inner product with f, and boundary values equal to g.

2.1.5 General linear elliptic PDEs

In many ways, the Poisson problem introduced above is the archetypical elliptic partial differential equation. However, the results in this thesis hold for a much larger class of elliptic partial differential equations, which we will now introduce. We consider a connected open subset $\Omega \in \mathbb{R}^d$ with Lipschitz boundary and denote by $C_c^\infty(\Omega)$ the space of infinitely smooth functions with support compactly contained in Ω. For an integer s, and $u \in C_c^\infty(\Omega)$, we define the *Sobolev norm*

$$\|\cdot\|_{H^s(\Omega)}^2 \coloneqq \sum_{0 \leq r \leq s} \|D^s\|_{L^2(\Omega)}^2, \tag{2.12}$$

where $D^r u(x)$ is the order r tensor containing the derivatives of order r of u in x. We denote as H_0^s the closure of $C_c^\infty(\Omega)$. For $u \in C_c^\infty(\Omega)$, we define the dual Sobolev norm $\|\cdot\|_{H^{-s}(\Omega)}$ as

$$\|u\|_{H^{-s}(\Omega)} \coloneqq \sup_{0 \neq v \in H_0^s(\Omega)} \frac{\int_\Omega u(x)v(x)\,\mathrm{d}x}{\|v\|_{H_0^s(\Omega)}} \tag{2.13}$$

and the dual Sobolev space $H^{-s}(\Omega)$ as closure of $C_x^\infty(\Omega)$ with respect to $\|\cdot\|_{H^{-s}(\Omega)}$. For $u \in H^{-s}(\Omega)$ and $v \in H_0^s(\Omega)$ that are obtained as limits of $u_k, v_k \subset C_0^\infty(\Omega)$, we define

$$[u, v] \coloneqq \lim_{k \to \infty} [u_k, v_k] \coloneqq \lim_{k \to \infty} \int_\Omega u_k(x) v_k(x)\,\mathrm{d}x. \tag{2.14}$$

For the purposes of this thesis, we define an elliptic partial differential operator as follows.

Definition 1 (Linear elliptic partial differential operator (elliptic operator))**.** *For a domain $\Omega \subset \mathbb{R}^d$ with Liptschitz and a positive integer s, we call an operator $\mathcal{L} : H_0^s(\Omega) \longrightarrow H^{-s}(\Omega)$ a linear elliptic partial differential operator if it is linear, bounded, invertible, local in the sense that for $u, v \in C_c^\infty(\Omega)$*

$$\forall u, v \in C_c^\infty(\Omega) : \operatorname{supp} u \cap \operatorname{supp} v = \emptyset \Rightarrow [\mathcal{L}u, v] = 0, \tag{2.15}$$

symmetric in the sense that

$$\forall u, v \in C_c^\infty(\Omega) : [\mathcal{L}u, v] = [\mathcal{L}v, u], \tag{2.16}$$

and positive in the sense that

$$\forall u \in H_0^s(\Omega) :\Rightarrow [\mathcal{L}u, u] \geq 0. \tag{2.17}$$

The linear elliptic PDE associated to the elliptic operator $\mathcal{L}$ with forcing term or *right hand side* $f \in H^{-s}(\Omega)$ is then given by

$$\mathcal{L}u = f, \tag{2.18}$$

with the Poisson problem being the special case of $\mathcal{L} = -\Delta$. We frequently refer to the inverse of an elliptic operator $\mathcal{L}$ as the *Green's operator* denoted by $\mathcal{G} := \mathcal{L}^{-1}$.

For a given elliptic operator, we write $\|u\|_{\mathcal{L}}^2 := [\mathcal{L}u, u]$ for its associated *operator norm* on $H_0^s(\Omega)$ and $\|u\|_{\mathcal{L}^{-1}}^2 := [u, \mathcal{L}^{-1}u]$ for its associated *dual operator norm* on $H^{-s}(\Omega)$. Where this does not lead to ambiguities, we instead write $\|\cdot\|$ and $\|\cdot\|_*$ for the operator norm and its dual.

Generalizing the Dirichlet energy introduced in the last section, we obtain a general variational principle for elliptic PDEs, given by

$$\mathcal{L}u = f \Leftrightarrow u = \operatorname*{arg\,min}_{v \in H_0^s(\Omega)} \frac{1}{2}\|v\|_{\mathcal{L}}^2 - [f, u]. \tag{2.19}$$

2.1.6 Discretization and solution of elliptic PDEs

Elliptic operators, acting on infinite-dimensional function spaces, are infinite-dimensional objects. In order to solve the associated equations numerically, we have to find finite-dimensional representations that we can manipulate using numerical linear algebra.

The popular *Galerkin* approach to discretization is based on the observation that

$$\mathcal{L}u = f \Leftrightarrow \forall v \in H^s(\Omega) : [\mathcal{L}u, v] = [f, v]. \tag{2.20}$$

This problem can be approximated by choosing a finite-dimensional subspace $\mathbb{V}^N = \operatorname{span}\{v_1, \dots v_N\} \subset H_0^s(\Omega)$ and searching for $u^N \subset \mathbb{V}^N$ such that

$$[\mathcal{L}u^N, v] = [f, v], \forall v \in \mathbb{V}^N. \tag{2.21}$$

This problem can be reexpressed as

$$u^N = \sum_{1 \leq k \leq N} (A^{-1}b)_k v_k, \tag{2.22}$$

where $b_i := [f, v_i]$ and $A_{ij} := [\mathcal{L}v_i, v_j]$ is called the *stiffness matrix* of $\mathcal{L}$. One can show that if $\lim_{N\to\infty} V^N = H_0^s(\Omega)$, u^N converges to the true solution of the elliptic PDE. If instead of the differential operator $\mathcal{L}$ we have access to the Green's operator $\mathcal{G} = \mathcal{L}^{-1}$, we similarly obtain a Galerkin discretization of $\mathcal{G}$ by using a finite dimensional subspace $\mathbb{W}^N = \operatorname{span}\{w_1, \dots w_N\} \subset H^{-s}(\Omega)$ to form the Green's matrix $\Theta_{ij} := [w_i, \mathcal{G}w_j]$.

2.2 Smooth Gaussian processes

2.2.1 Gaussian vectors

A Gaussian random vector $X \sim \mathcal{N}(\mu, \Theta)$ with mean $\mu \in \mathbb{R}^N$ and $\Theta \in \mathbb{R}^{N \times N}$ symmetric positive definite is a random element of $\mathbb{R}^N$ that is distributed according to the probability density (with respect to the Lebesgue measure)

$$p_{\mathcal{N}(\mu,\Theta)}(X) := \frac{1}{\sqrt{(2\pi)^N \det(\Theta)}} \exp\left(-\frac{1}{2}(X-\mu)^\top \Theta^{-1}(X-\mu)\right). \tag{2.23}$$

Gaussian vectors are an immensely popular modeling tool for multivariate data. They can be motivated by a variety of ways including rotational invariance (with respect to the inner product induced by Θ^{-1}) [132, Chapter 13], the central limit theorem [132, Chapter 5], and even game theory [103, 190]. Beyond these theoretical considerations, they have the computational benefit that most probabilistic operations on Gaussian vectors can be characterized in terms of linear algebraic operations on Θ and μ.

1. The mean and covariance of $X \sim \mathcal{N}(\mu, \Theta)$ are given by μ and Θ. We thus refer to Θ as the covariance matrix of $\mathcal{N}(\mu, \Theta)$.

2. The marginal log-likelihood of the Gaussian process model given data y is given as

$$-\frac{1}{2}(y-\mu)^\top \Theta^{-1}(y-\mu) - \frac{1}{2}\operatorname{logdet}(\Theta) - \frac{N}{2}\log(2\pi). \tag{2.24}$$

3. Writing $X = \binom{X_1}{X_2} \in \mathbb{R}^{N_1+N_2}$, and blocking μ and Θ accordingly, the distribution of X_2 conditioned on X_1 is given as

$$X_2 \mid X_1 \sim \mathcal{N}\left(\mu_2 + \Theta_{2,1}\left(\Theta_{1,1}\right)^{-1}(X_1-\mu_1), \Theta_{2,2} - \Theta_{2,1}\left(\Theta_{1,1}\right)^{-1}\Theta_{2,1}\right). \tag{2.25}$$

4. The conditional correllations of X are encoded in the *precision* $A := \Theta^{-1}$, in that

$$\frac{A_{ij}}{\sqrt{A_{ii}A_{jj}}} = (-1)^{i \neq j} \frac{\mathbb{C}\mathrm{ov}\left[X_i, X_j \mid X_{\notin\{i,j\}}\right]}{\sqrt{\mathbb{V}\mathrm{ar}\left[X_i \mid X_{\notin\{i,j\}}\right]\mathbb{V}\mathrm{ar}\left[X_j \mid X_{\notin\{i,j\}}\right]}}, \tag{2.26}$$

where $\notin \{i, j\}$ denotes the set $\{1, \dots N\} \setminus \{i, j\}$.

2.2.2 Gaussian processes

A common setting in Gaussian process statistics is such that we observe data $y_{\mathrm{Tr}} \in \mathbb{R}^{N_{\mathrm{Tr}}}$ at N_{Tr} training locations in $\mathbb{R}^d$ and choose a covariance matrix $\Theta_{\mathrm{Tr},\mathrm{Tr}}$ that explains y_{Tr}, for instance by maximizing the marginal likelihood 2 of $\mathcal{N}\left(0, \Theta_{\mathrm{Tr},\mathrm{Tr}}\right)$ given y_{Tr}. If we want to use this data to predict data at a different set of N_{Pr} prediction locations, we need not only $\Theta_{\mathrm{Tr},\mathrm{Tr}}$ but also the training-prediction covariance $\Theta_{\mathrm{Tr},\mathrm{Pr}}$, (and $\Theta_{\mathrm{Pr},\mathrm{Pr}}$ if we want to perform uncertainty quantification). Without observing data from the prediction locations, there seems to be no way for us to decide which $\Theta_{\mathrm{Tr},\mathrm{Pr}}, \Theta_{\mathrm{Pr},\mathrm{Pr}}$ to choose.

In order to define covariances between arbitrary locations, we can model our data y_{Tr} as measurements of an infinite-dimensional Gaussian vector assigning a value to each point in $\mathbb{R}^d$. This idea is formalized by the notion of a *Gaussian field*.

Definition 2. *Given a separable Banach space $\mathcal{B}$ and its dual $\mathcal{B}^*$, let $\mathcal{L} : \mathcal{B} \longrightarrow \mathcal{B}^*$ and $\mathcal{G} := \mathcal{L}^{-1} : \mathcal{B}^* \longrightarrow \mathcal{B}$ be symmetric and bounded linear operators. Let furthermore $\boldsymbol{H}$ be a Hilbert space of univariate Gaussian random variables, equipped with the L^2 inner product. We call a linear map $\xi : \mathcal{B}^* \longrightarrow \boldsymbol{H}$ a Gaussian field with covariance operator $\mathcal{G}$, precision operators $\mathcal{L}$, and mean $\mu \in \mathcal{B}$, if it is an affine isometry, meaning that for all $\phi \in \mathcal{B}^*$ we have*

$$\xi\left(\phi\right) \sim \mathcal{N}([\phi, \mu], [\phi, \mathcal{G}\phi]). \tag{2.27}$$

Following the notation for Gaussian vectors, we then write $\xi \sim N(\mu, \mathcal{G})$. Here, $[\cdot, \cdot]$ is the duality product of $\mathcal{B}^*$ and $\mathcal{B}$. Abusing notation, we write $[\phi, \xi] := \xi(\phi)$.

For finite-dimensional $\mathcal{B}$, ξ can be obtained from a Gaussian vector $X \sim \mathcal{N}\left(\mu, \mathcal{G}\right)$ as $\xi(\phi) := [\phi, X]$. For infinite-dimensional $\mathcal{B}$, a random element $X \in \mathcal{B}$ that realizes the mapping $\xi : \mathcal{B}^* \longrightarrow \mathbf{H}$ usually does not exist. Instead, ξ can be realized by a probability measure on a space larger than $\mathcal{B}$, or by a cylinder measure on $\mathcal{B}$ itself (see [190][Chapter 17] for additional details). Either way, ξ provides us with a way to assign to any finite collection of measurements $\{\phi_i\}_{1 \leq i \leq N} \in \mathcal{B}^*$ a joint distribution given by $\mathcal{N}(([\phi_i, \mu])_{1 \leq i \leq N}, \Theta)$, where $\Theta_{ij} := [\phi_i, \mathcal{G}\phi_j]$. If $\mathcal{B}$ is a subset of the continuous functions on $\mathbb{R}^d$, by choosing the $\{\phi_i\}_{1 \leq i \leq N} \in \mathcal{B}^*$ as pointwise evaluations in a set of points $\{x_i\}_{1 \leq i \leq N}$, we can obtain covariance matrices of the joint distributions of arbitrary combinations of data points. Given training data y_{Tr} in a set of training locations, we can use the maximum likelihood criterion in order to choose a Gaussian field $\xi \sim \mathcal{N}(0, \mathcal{G})$. Once found, the Gaussian field ξ allows us to perform inference at arbitrary sets of prediction points.

2.2.3 Smooth Gaussian processes and elliptic PDE

Even in the setting of Gaussian vectors, we have somewhat brushed over the question of *how* to choose a covariance model. For Gaussian fields, the space of possible choices is vastly bigger, making it even less obvious how to single out a particular covariance operator for a given task.

The choice of covariance operator $\mathcal{G}$ is a modeling choice whereby we assume structure in our data that we can later use to perform inference. One of the most fundamental assumptions on data is *smoothness*, meaning that the spatial derivatives of the unknown function u are not too large and therefore u does not vary too rapidly as a function from $\mathbb{R}^d$ to $\mathbb{R}$. Restricting our attention to centered Gaussian processes, we can **formally** extend the formula Equation 2.23 to the Gaussian field setting by writing

$$p_{\mathcal{N}(0,\mathcal{G})}(u) \propto \exp\left(-\frac{1}{2}[\mathcal{L}u, u]\right). \tag{2.28}$$

The log-likelihood of a realization u decreases, as the quadratic form $[\mathcal{L}u, u]$ increases. This suggests defining Gaussian fields by choosing an $\mathcal{L}$ for which $[\mathcal{L}u, u]$ is a measure of the *roughness* of the function. The elliptic operators of Section 2.1.5 where chosen as bounded invertible linear operators from $H_0^s(\Omega)$ to $H^{-s}(\Omega)$. Therefore, their associated quadratic norm is equivalent to the squared Sobolev norm (see for instance [190, Lemma 2.4])

$$\|\mathcal{L}^{-1}\|^{-1}\|u\|^2_{H_0^s(\Omega)} \leq [\mathcal{L}u, u] \leq \|\mathcal{L}\|\|u\|^2_{H_0^s(\Omega)}. \tag{2.29}$$

The Sobolev norms, being the sum of the L^2 norms of the first s derivatives, provide a natural measure of the roughness of a realization $\mathcal{L}$. This makes elliptic operators a natural choice for the precision operators of finitely smooth Gaussian fields. They can either be constructed by discretizing the precision operator using a finite element basis (see [155, 207] for examples), based on a closed-form of the Green's operator $\mathcal{G}$ (the most well-known example being the Matérn family of kernels [167, 248, 249]). We point out that the popular *Gaussian* kernel is not a Green's function of an elliptic PDE, but of a parabolic PDE corresponding to infinite order smoothness or $s \to \infty$. We also note that many finitely smooth Gaussian process models from the literature, such as fractional order Matérn covariances or the "Cauchy class" of covariance functions, do not strictly fit into the framework of Section 2.1.5, yet show the same behavior in practice.

2.3 The cubic bottleneck

In Sections 2.1 and 2.2, we have introduced the closely related problems computing with linear elliptic operators and smooth Gaussian processes. Here and in the following, we denote as $\mathcal{L} : H_0^s(\Omega) \longrightarrow H^{-s}(\Omega)$ the differential precision operator, and as $\mathcal{G} : H^{-s}(\Omega) \longrightarrow H_0^s(\Omega)$ its inverse, the Green's or covariance operator. We cannot compute with the infinite operators $\mathcal{L}$ and $\mathcal{G}$ on a finite machine and therefore approximate $\mathcal{L}$ and $\mathcal{G}$ by the stiffness or precision matrix $A \in \mathbb{R}^{N \times N}$ and the Green's or covariance matrix $\Theta \in \mathbb{R}^{N \times N}$ that describe the actions of $\mathcal{L}$ and $\mathcal{G}$ on N-dimensional subspaces of $H_0^s(\Omega)$ and $H^{-s}(\Omega)$.

In most applications, we are interested in performing some of the following operations

1. Compute $f \rightarrow A^{-1} f$ to solve an elliptic PDE with right hand side f using a Galerkin discretization.
2. Compute $b \rightarrow \Theta_{\Omega,\partial\Omega}\left(\Theta_{\partial\Omega,\partial\Omega}\right)^{-1} b$ to solve an elliptic boundary value problem with boundary data b.
3. Compute $y_{\mathrm{Tr}} \rightarrow \Theta_{\mathrm{Pr},\mathrm{Tr}}\left(\Theta_{\mathrm{Tr},\mathrm{Tr}}\right)^{-1} y_{\mathrm{Tr}}$ or $y_{\mathrm{Tr}} \rightarrow -\left(A_{\mathrm{Pr},\mathrm{Pr}}\right)^{-1} A_{\mathrm{Pr},\mathrm{Tr}} y_{\mathrm{Tr}}$ to compute the conditional mean given training data y_{Tr}.
4. Compute $\Theta_{\mathrm{Pr},\mathrm{Pr}} - \Theta_{\mathrm{Pr},\mathrm{Tr}}\left(\Theta_{\mathrm{Tr},\mathrm{Tr}}\right)^{-1} \Theta_{\mathrm{Tr},\mathrm{Pr}}$ or $\left(A_{\mathrm{Pr},\mathrm{Pr}}\right)^{-1}$ to compute the conditional covariance matrix.
5. Compute the marginal log-likelihood
$$-\frac{1}{2}(y-\mu)^{\top} \Theta^{-1}(y-\mu) - \frac{1}{2}\operatorname{logdet}(\Theta) - \frac{N}{2}\log(2\pi), \tag{2.30}$$
or
$$-\frac{1}{2}(y-\mu)^{\top} A (y-\mu) + \frac{1}{2}\operatorname{logdet}(A) - \frac{N}{2}\log(2\pi), \tag{2.31}$$
and its derivatives with respect to Θ or A, in order to perform model selection using the likelihood criterion.
6. Sample from the Gaussian process $\mathcal{N}(\mu, \Theta)$ or $\mathcal{N}\left(\mu, A^{-1}\right)$.

The problem in all these applications is that while A and Θ^{-1} are (almost) sparse when using localized basis functions or measurements to discretize $\mathcal{L}$ and Θ, their inverses A^{-1} and Θ are invariably *dense*. This can be illustrated by plotting the

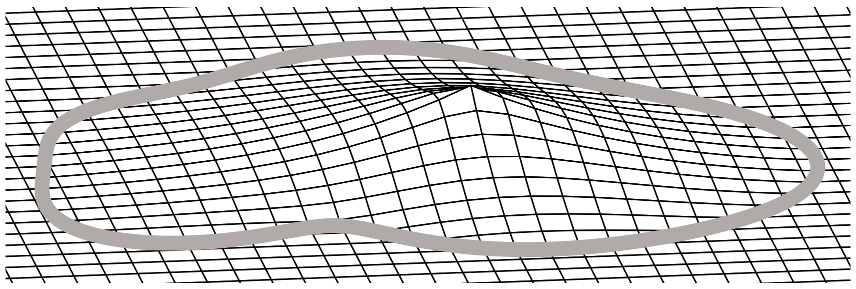


Figure 2.1: **An elliptic Green's function.** We plot the graph of the Green's function $y \mapsto [x_0, \mathcal{G}y]$ of an elliptic PDE on the domain Ω. Most of its values are significantly larger than zero, leading to a dense matrix Θ.

Green's function $(x, y) \mapsto [\boldsymbol{\delta}_x, \mathcal{G}\boldsymbol{\delta}_y]$ for $\boldsymbol{\delta}$ denoting the Dirac delta function, as shown in Figure 2.1.

Performing the above operations using dense linear algebra leads to computational complexity $O\left(N^2\right)$ in space and $O\left(N^3\right)$ in space, making them excessively expensive for $N \gtrapprox 10^5$ and presenting a major computational bottleneck for the computation with large amounts of data or high-resolution physical models.

Existing approaches

Given the immense importance of elliptic operators throughout computational mathematics, it is not surprising that a vast range of methods has been developed to facilitate their computational treatment. Each of these methods is based on finding *simple* representations of the seemingly complicated solution operators A^{-1} and Θ.

Efficient representation of dense covariance matrices

Most approximations of Θ present in the literature rely on sparsity, low-rank structure, their combinations, and multiscale variants. Low-rank techniques such as the Nyström approximation [87, 221, 250] or rank-revealing Cholesky factorization [21, 85] seek to approximate Θ by low-rank matrices whereas sparsity-based methods like *covariance tapering* [88] seek to approximate Θ with a sparse matrix by setting entries corresponding to long-range interactions to zero. These two approximations can also be combined to obtain *sparse low-rank* approximations [25, 199, 212, 215, 223], which can be interpreted as imposing a particular graphical structure on the Gaussian process. When Θ is neither sufficiently sparse nor of sufficiently low rank, these approaches can be implemented in a hierarchical manner. For low-rank methods, this leads to *hierarchical* ($\mathcal{H}$- and $\mathcal{H}^2$-) matrices

[112, 114, 115], *hierarchical off-diagonal low rank* (HODLR) matrices [10, 11], and hierarchically semiseparable (HSS) matrices [150, 251] that rely on computing low-rank approximations of sub-blocks of Θ corresponding to far-field interactions on different scales. The interpolative factorization developed by [118] combines hierarchical low-rank structure with the sparsity obtained from an elimination ordering of nested-dissection type. Hierarchical low-rank structures were originally developed as an algebraic abstraction of the fast multipole method of [100]. In order to construct hierarchical low-rank approximations from entries of the kernel matrix efficiently, both deterministic and randomized algorithms have been proposed [28, 164]. For many popular covariance functions, including Green's functions of elliptic PDEs [29], hierarchical matrices allow for (near-)linear-in-N complexity algorithms for the inversion and approximation of Θ, at exponential accuracy. Wavelet-based methods [38, 94], using the separation and truncation of interactions on different scales, can be seen as a hierarchical application of sparse approximation approaches. The resulting algorithms have near-linear computational complexity and rigorous error bounds for asymptotically smooth covariance functions. [82] use operator-adapted wavelets to compress the expected solution operators of random elliptic PDEs. In [134], although no rigorous accuracy estimates are provided, the authors establish the near-linear computational complexity of algorithms resulting from the multiscale generalization of probabilistically motivated sparse and low-rank approximations [25, 199, 212, 215, 223]. Finally, we note that Veccha approximations [136, 241] and factorized sparse approximate inverses [33, 141] are able to utilize sparsity of the Cholesky factors of A or Θ^{-1} to compute efficient approximations of Θ and A^{-1}. We will encounter these algorithms again in Chapter 6.

Obtaining approximations of A^{-1} from A

In applications where we are given access to the sparse stiffness or precision matrix A, we can use slightly different techniques.

Firstly, we can use iterative methods such as conjugate gradient [218] to approximate the matrix-vector products $v \mapsto A^{-1}$ in terms of matrix-vector products with A. These matrix-vector products can usually be computed in complexity $O(N)$ providing us with a candidate for a much more efficient algorithm. However, the sparsity pattern of A will usually be geometric in the sense that it captures interactions of basis functions within a certain distance h. Therefore, in order for the approximation to A^{-1} to account for *any* interaction between two degrees of freedom of distance δ, we need to perform at least δ/h iterations of the conjugate gradient method. This means that the required number of iterations grows at least with $N^{1/d}$, where d is the spatial dimension. In practice, in particular for higher-order PDEs, it can be much higher than that. This problem can be mitigated by the use of multigrid solvers [41, 81, 111, 113] that work on grids of different length scales simultaneously. In classical multigrid methods, these grids are constructed based on only geometric information, without accounting for the values of the entries of A. This approach achieves high performance on spatially homogeneous problems, but it can perform arbitrarily badly on general PDEs with *rough* coefficients [20]. In order to overcome this problem, algebraic multigrid methods [7, 41, 42, 253, 255] or the operator adapted methods of [121, 189, 190] construct the multiresolution hierarchy using the entries of A.

The sparsity of A further enables us to apply techniques from sparse linear algebra such as sparse Cholesky factorization in nested dissection ordering [90–92, 156]. The computational complexity of these methods is superlinear, but they can be highly efficient for moderate size. Therefore, some authors [155, 205–207] in the spatial statistics literature suggest replacing the kernel matrices obtained from Matérn covariance functions with the inverses of sparse stiffness matrices obtained from finite element discretization of the associated PDE.

Chapter 3

SPARSE CHOLESKY FACTORS BY SCREENING

3.1 Gaussian elimination and Cholesky factorization

Gaussian elimination might be the oldest method of numerical linear algebra, being known to Chinese mathematicians for more than two millennia [99]. For a more modern treatment, we refer the reader to [236]. When applied to symmetric and positive definite matrices, it is also known as Cholesky factorization and amounts to representing the input matrix $A = LL^{\top}$ as the product of a lower triangular matrix with its transpose, using Algorithm 1.

Algorithm 1 Cholesky factorization.

```
Input: Positive definite matrix $A \in \mathbb{R}^{N \times N}$
Output: Lower triangular matrix $L \in \mathbb{R}^{N \times N}$
1: for $i = 1 : N$ do
2:     $L_{:i} \leftarrow A_{:i} / \sqrt{A_{ii}}$
3:     for $j = i + 1 : N$ do
4:         for $k = i + 1 : N$ do
5:             $A_{kj} \leftarrow A_{kj} - \frac{A_{ki} A_{ji}}{A_{ii}}$
6:         end for
7:     end for
8: end for
```

Once the lower triangular factor L is computed, the determinant of A can be obtained as the squared product of the diagonal entries of L and linear systems in A can be solved by substitution (Algorithm 2)

Algorithm 2 Solving a linear system by forward and back substitution.

```
Input: Nonsingular lower triangular matrix $L \in \mathbb{R}^{N \times N}$ and vector $b \in \mathbb{R}^{N}$.
Output: Vector $x = (LL^{\top})^{-1} \in \mathbb{R}^{N}$ in place of $b$
1: {Forward substitution computing $L^{-1}b$ in place:}
2: for $i = 1 : N$ do
3:     $b_i \leftarrow \left(b_i - L_{i,1:(i-1)} \cdot b_{1:(i-1)}\right) / L_{ii}$
4: end for
5: {Backward substitution computing $L^{-\top}b$ in place:}
6: for $i = N : -1 : 1$ do
7:     $b_i \leftarrow \left(b_i - L_{(i+1):N,i} \cdot b_{(i+1):N}\right) / L_{ii}$
8: end for
```

Furthermore, by multiplying a standard Gaussian vector with either $L^\top$ (or L^{-1}), we can obtain samples of a Gaussian vector with covariance (or precision matrix) given by A. Finally, by using only the first k columns of L, we can obtain a rank-k approximation or A.

3.2 Sparse Cholesky factorization

A serious limitation of Cholesky factorization is that its computational cost scales cubically with the dimension N of the matrix. If A is sparse, as is the case in many applications, this cost seems excessive. A vast body of work is concerned with using sparsity properties of A to reduce the computational cost of its Cholesky factorization. The main difficulty that these methods need to overcome is the phenomenon of *fill-in* by which the Cholesky factor L of A can contain considerably more nonzero entries than the original matrix A.

Fill-in is usually analyzed using graph theory, by defining the undirected *sparsity graph* of a symmetric matrix $A \in \mathbb{R}^{N \times N}$ as the undirected graph with N vertices that has an edge between nodes i and j if and only if $A_{ij} \neq 0$. After each iteration of the outer loop in Line 1, the sparsity graph of A is modified by adding an edge between each pair of neighbors of i that are still to be eliminated.

As illustrated in Figure 3.1, the fill-in produced by the factorization strongly depends on the *elimination ordering*: the ordering of the rows and columns of the matrix. This observation initiated a substantial body of research that uses graph-theoretical approaches to find *fill-in reducing* elimination orderings. Popular approaches are approximate minimal degree [12], (reverse) Cuthill–McKee [56], and nested dissection [90, 156] orderings. In Figure 3.1, we show how these approaches affect the amount of fill-in when factorizing a finite-difference Laplacian in two dimensions. For a general sparsity pattern, the cost of the resulting algorithms is hard to characterize, but in the important special case of lattice graphs in two and three dimensions, it is well understood. In two dimensions, the factors have $O(N \log(N))$ nonzero entries and can be computed in $O\left(N^{3/2}\right)$ time with relatively small constants, making sparse Cholesky factorization an attractive choice for two-dimensional problems of moderate size. In three dimensions, the factors have $O\left(N^{4/3}\right)$ nonzeros and can be computed in $O\left(N^2\right)$ time [63]. The quadratic scaling of the computational cost, and in particular the superlinear memory requirements, greatly limit the utility of sparse Cholesky factorization for three-dimensional problems.

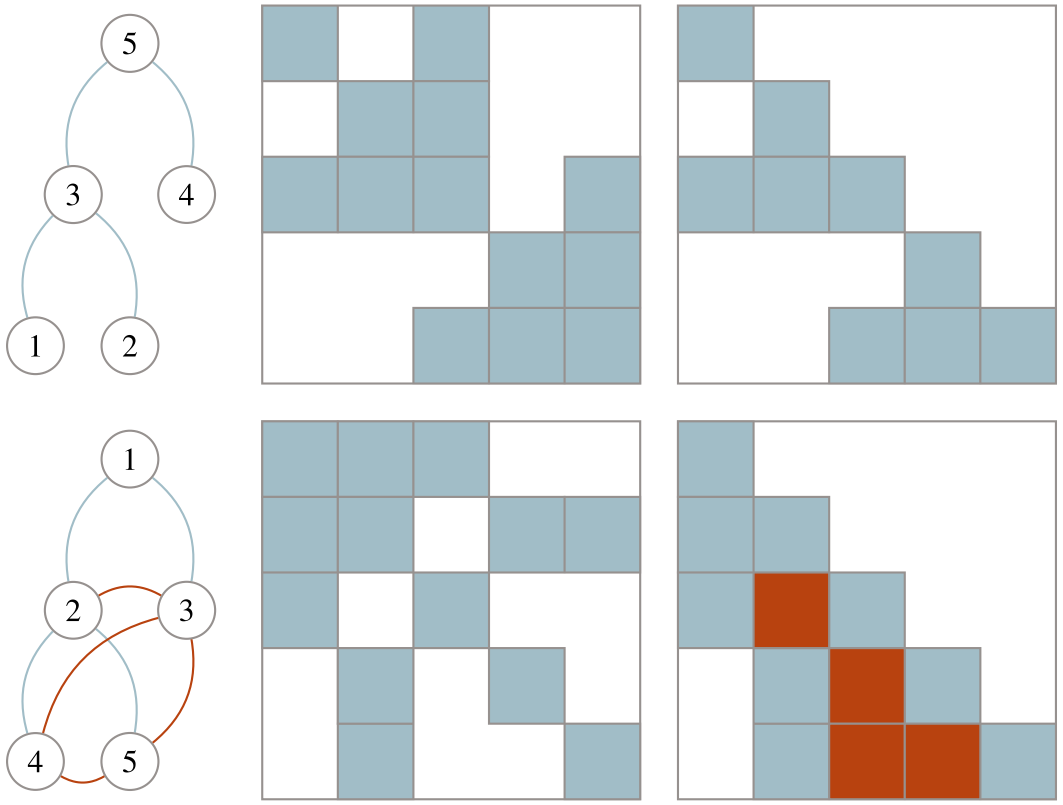
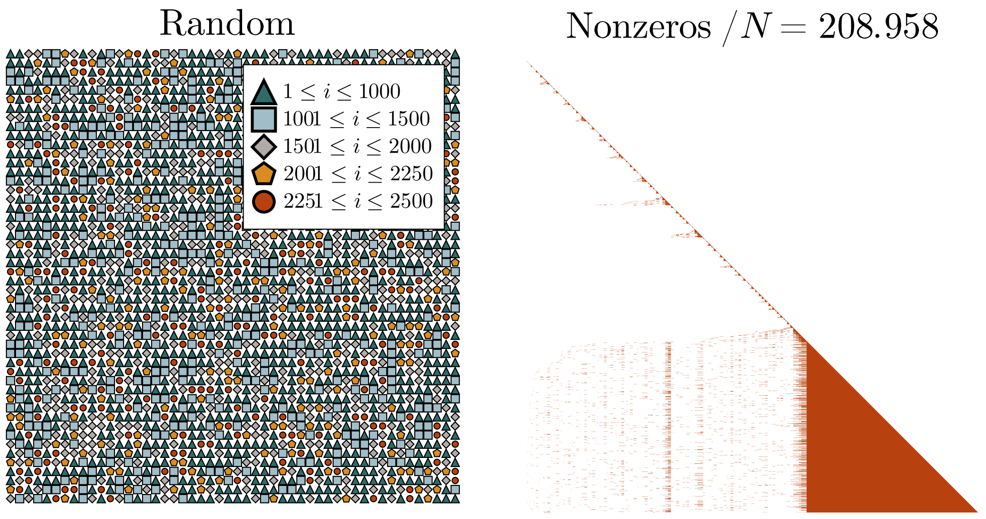


Figure 3.1: **Fill-in.** For a given matrix with nonzero entries denoted in blue, the amount of fill-in (in red) depends strongly of the ordering of the rows and columns of the matrix (note that we do not plot self-edges of the sparsity graph).



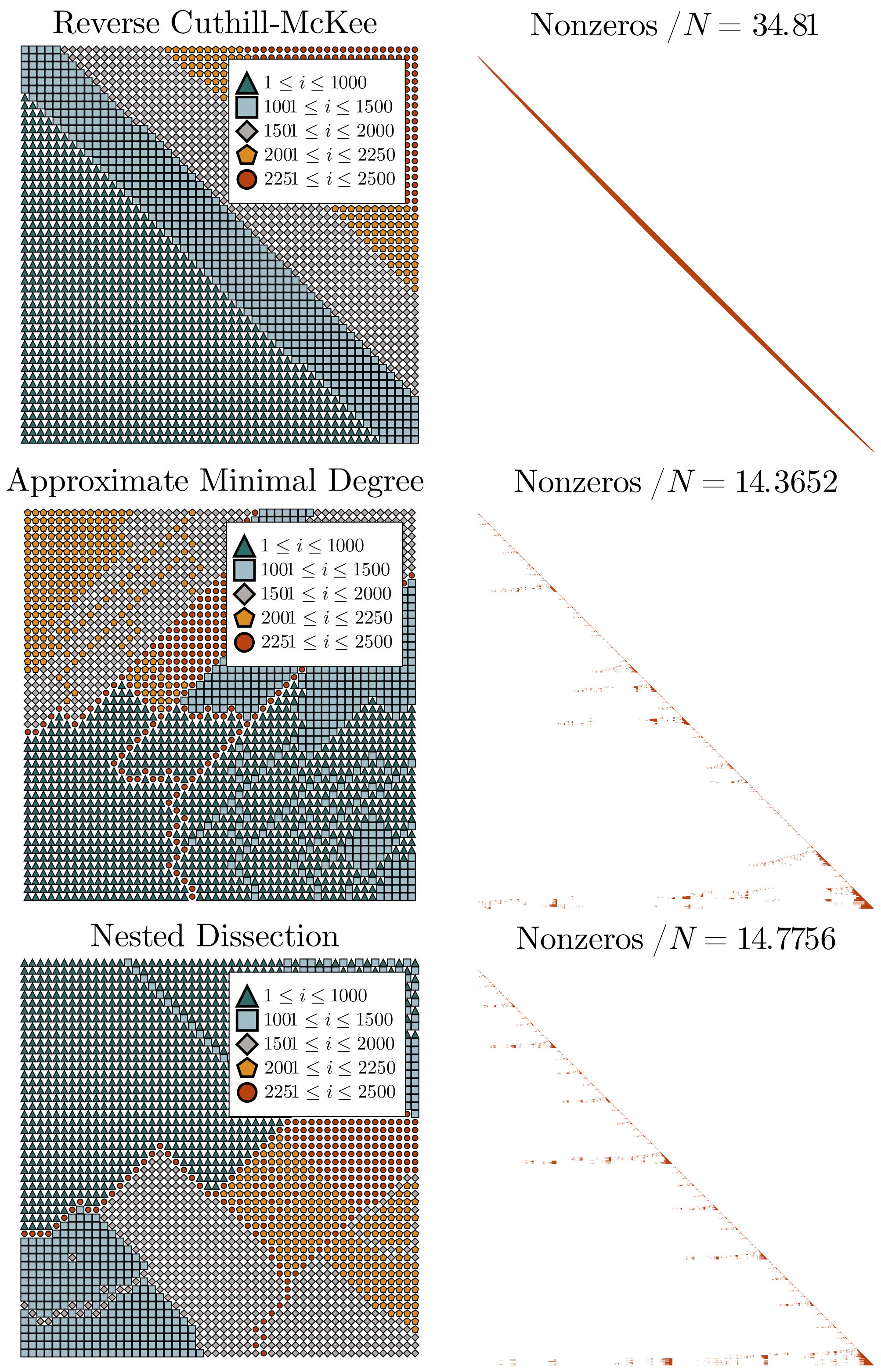


Figure 3.1: **Fill-reducing ordering.** We show common reordering heuristics and their effects on the sparsity of the Cholesky factor of a finite difference Laplacian.

3.3 Gaussian elimination and Gaussian conditioning

As described in the last section, the amount of fill-in incurred when computing the Cholesky factorization of sparse matrices can be studied by analyzing the sparsity graph of the matrix. This line of work leads to the development software libraries such as CHOLMOD [46] that are now widely applied for solving sparse linear systems in practice. But the graph-theoretic way of thinking about Cholesky factorization has two important limitations. It is not helpful when computing the Cholesky factorization of dense matrices, and the resulting algorithms usually have superlinear running time.

In this section, we present an alternative, probabilistic heuristic for reasoning about the sparsity of Cholesky factors.

The dense (block-)Cholesky factorization of a matrix Θ can be seen as the recursive application of the matrix identity

$$\begin{pmatrix} \Theta_{1,1} & \Theta_{1,2} \\ \Theta_{2,1} & \Theta_{2,2} \end{pmatrix} = \begin{pmatrix} \mathrm{Id} & 0 \\ \Theta_{2,1}(\Theta_{1,1})^{-1} & \mathrm{Id} \end{pmatrix} \begin{pmatrix} \Theta_{1,1} & 0 \\ 0 & \Theta_{2,2} - \Theta_{2,1}(\Theta_{1,1})^{-1}\Theta_{1,2} \end{pmatrix} \begin{pmatrix} \mathrm{Id} & (\Theta_{1,1})^{-1}\Theta_{1,2} \\ 0 & \mathrm{Id} \end{pmatrix}, \tag{3.1}$$

where, at each step of the outermost loop, the above identity is applied to the Schur complement $\Theta_{2,2} - \Theta_{2,1}\left(\Theta_{1,1}\right)^{-1}\Theta_{1,2}$ obtained at the previous step. As a result, the k-th column of the final Cholesky factor L is a multiple of the first column of the Schur complement when setting $\Theta_{1,1} = \Theta_{1:(k-1),1:(k-1)}$.

Interpreting the positive definite matrix Θ as the covariance matrix of the Gaussian vector $X = (X_1, X_2) \sim \mathcal{N}(0, \Theta)$, the well-known identities

$$\mathbb{E}[X_2 \mid X_1 = a] = \Theta_{2,1}(\Theta_{1,1})^{-1}a, \tag{3.2}$$

$$\mathbb{C}\mathrm{ov}[X_2 \mid X_1] = \Theta_{2,2} - \Theta_{2,1}(\Theta_{1,1})^{-1}\Theta_{1,2} \tag{3.3}$$

allow us to relate Equation 3.1 to the conditional expectation and variance of X.

Observation 1. *For Θ positive definite with lower triangular Cholesky factor L and $X \sim \mathcal{N}(0, \Theta)$, we have*

$$L_{ij} = \frac{\mathbb{C}\mathrm{ov}\left[X_i, X_j \middle| X_{1:(j-1)}\right]}{\sqrt{\mathbb{V}\mathrm{ar}\left[X_j \middle| X_{1:(j-1)}\right]}} \tag{3.4}$$

In particular, the (i, j)-th entry of L is (almost) zero if and only if the X_i and X_j are (almost) independent, conditional on $X_{1:(j-1)}$.

This observation follows directly from well-known results, yet we are not aware that it has made before. Since conditional independence is a core concept of probability

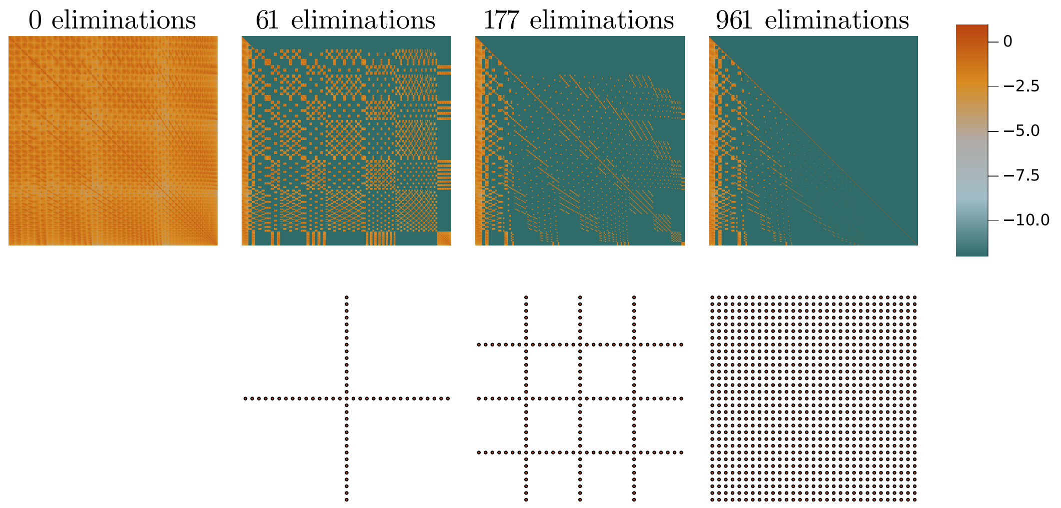


Figure 3.2: **Sparsification by elimination.** As we eliminate more columns (left to right), the Cholesky factor and Schur complement become increasingly sparse (top row, magnitude on $\log_{10}$-scale). The bottom row shows the geometric locations corresponding to the eliminated columns, and how they dissect the graph.

theory, there should be interesting *dense* matrices with *sparse* Cholesky factors, contrary to what the classical view described in Section 3.2 suggests.

A first example of this phenomenon are Gaussian processes with a *Markov property*.

Definition 3. *A random vector $X \in \mathbb{R}^N$ has the Markov property according to the graph G with vertices given by $V = \{1, \ldots N\}$ if for all $I, J \subset V$ that are not connected by an edge of G, X_I and X_J are independent, conditional on $\{X_k\}_{k \in V \setminus (I \cup J)}$.*

For $X \sim \mathcal{N}(0, \Theta)$, this is equivalent to assuming that the precision matrix $A = \Theta^{-1}$ has G as its sparsity graph. When computing the Cholesky factorization of the dense inverse of a sparse matrix, this suggests using an elimination ordering that recursively divides the sparsity graph into conditionally independent components, as illustrated in Figure 3.2. As we progress through the factorization, the initially dense matrix becomes more and more sparse. Instead of fill-in, we observe a novel *fade-out* phenomenon!

The elimination ordering in Figure 3.2 is the *reverse* of a nested dissection-type ordering (see Figure 3.1 and [90]). This is no coincidence since, as mentioned in Section 2.2.1, the inverse covariance or *precision* matrix $A := \Theta^{-1}$ of a Gaussian vector X is related to its conditional correlation by

$$\frac{A_{ij}}{\sqrt{A_{ii}A_{jj}}} = (-1)^{i \neq j} \frac{\mathbb{C}\mathrm{ov}\left[X_i, X_j \mid X_{\notin\{i,j\}}\right]}{\sqrt{\mathbb{V}\mathrm{ar}\left[X_i \mid X_{\notin\{i,j\}}\right] \mathbb{V}\mathrm{ar}\left[X_j \mid X_{\notin\{i,j\}}\right]}}, \tag{3.5}$$

where $\notin \{i, j\}$ denotes the set $\{1, \dots N\} \setminus \{i, j\}$.

Observation 2. *For $A \in \mathbb{R}^{N \times N}$ positive definite with lower triangular Cholesky factor L and $X \sim \mathcal{N}(0, A^{-1})$, we have*

$$\frac{L_{ij}}{L_{jj}} = (-1)^{i \neq j} \frac{\mathbb{C}\mathrm{ov}\left[X_i, X_j \middle| X_{(j+1):N\setminus\{i\}}\right]}{\mathbb{V}\mathrm{ar}\left[X_j \middle| X_{(j+1):N\setminus\{i\}}\right]} \tag{3.6}$$

In particular, the (i, j)-th entry of L is (almost) zero if and only if the X_i and X_j are (almost) independent, conditional on $X_{(j+1):N\setminus\{i\}}$.

Observation 2 on Cholesky factorization of precision matrices is more well-known in the statistics community than Observation 1 and has been used, for instance, in the context of Vecchia approximation [135]. While sparsity of the j-th column of the Cholesky factor of Θ is determined by conditional independence conditional on the variables appearing *before* j in the ordering, the sparsity of A is determined by the conditional independence, conditional on the variables appearing *after* j in the ordering. In elimination orderings for Θ, we want the *earlier* variables to induce as much conditional independence as possible, while in elimination orderings for A, we want the *later* variables to induce as much conditional independence as possible. Therefore, *reversed* sparsity inducing elimination orderings for Θ tend to be sparsity inducing elimination orderings for A, and vice versa.

3.4 The screening effect

Observation 1 together with the Markov property in Definition 3 allows us to compute *exact* sparse Cholesky factors of dense matrices with sparse inverses. Unfortunately, as mentioned in Section 3.2, the sparsity pattern of $A = \Theta^{-1}$ only allows for superlinear time algorithms in most cases. This raises the question whether there are other mechanisms for conditional (near-)independence that lead to near-linear complexity algorithms.

Spatial statisticians have long observed that many smooth Gaussian processes are subject to the "screening effect" [49, 131], described by [228] as: "*The screening effect is the geostatistical term for the phenomenon of nearby observations tending to reduce the influence of more distant observations when using kriging (optimal linear* *prediction) for spatial interpolation*." The intuitive explanation behind the screening effect is that the values at any given site are most strongly dependent on those at nearby sites. Thus, after conditioning on values at nearby sites, the information gain from knowing the values at distant sites is marginal. They are conditionally near-independent from the value at the point in question. In Figure 3.3, we illustrated the screening effect in a Gaussian process with Matérn covariance.

As described in Section 2.2.3, many smooth Gaussian processes arise naturally from (local) elliptic PDEs and therefore naturally satisfy a form of Markov property. In this case, the maximin ordering can be seen as a relaxation of the nested dissection approach of Figure 3.2 that achieves *approximate* independence with much fewer conditioning points. However, the Markov property is only based on the nonzero pattern of the entries of the precision matrix. In contrast, the screening effect crucially depends on *what* its nonzero values are. One can construct matrices that satisfy a Markov property without exhibiting a screening effect. For instance, precision matrices of the form $(A - \lambda I)^2$ often do not admit a screening effect for large λ, even if the precision matrix A does.[1] Conversely, important covariance models such as fractional order Matérn kernels seem to admit a screening effect, even though they are not associated to a known local precision operator (see Section 5.5.4, in particular Tables 5.5 and 5.6).

3.5 The maximin ordering and sparsity pattern

When computing the Cholesky factorization of a covariance matrix Θ, Observation 1 suggests beginning the elimination with degrees of freedom that induce as much conditional independence as possible.

If Θ is subject to a screening effect as illustrated in Figure 3.3, this can be done by ensuring that for any k, the leading k columns correspond to points that are spread out as far and evenly as possible. This motivates the *maximin ordering* [8, 105] that successively picks points that are furthest from those points picked before.

Definition 4 (Maximin ordering)**.** *A maximin ordering of a point set $\{x_i\}_{i \in I} \subset \mathbb{R}^d$ with starting set C is obtained by picking as k-th point x_{i_k} a point that has the furthest possible distance to $\left\{x_{i_l}\right\}_{l<k} \cup C$. We call $\ell_k \coloneqq \operatorname{dist}\left(x_{i_k}, \left\{x_{i_l}\right\}_{l<k} \cup C\right)$ the length-scale of the point x_{i_k} in the maximin ordering.*

[1] This is closely related to the difficulty of solving high-frequency Helmholtz equations.

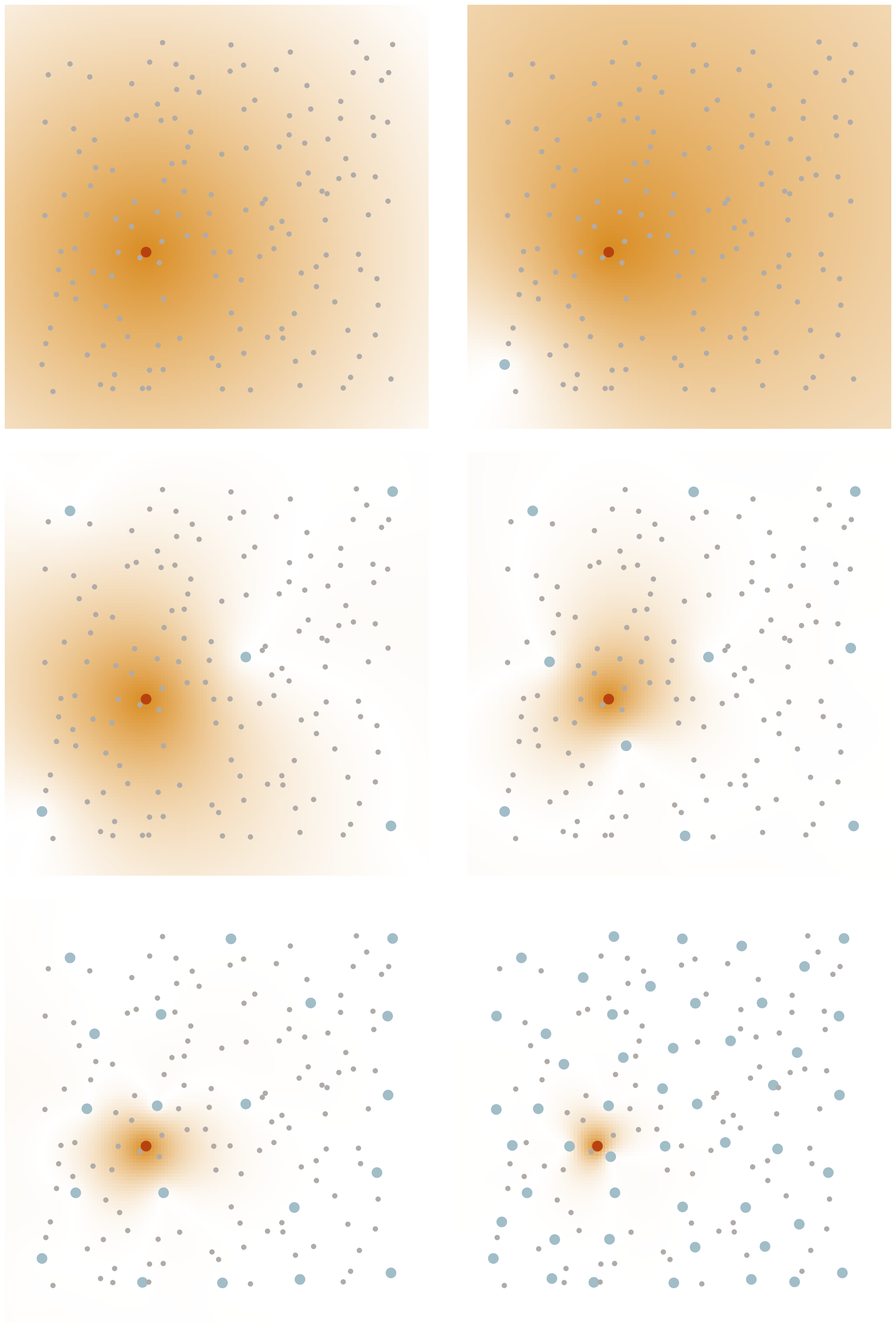


Figure 3.3: **The screening effect.** We condition a Gaussian process with Matérn covariance on increasing conditioning sets and plot the conditional correlation of the point in red as a heat map.

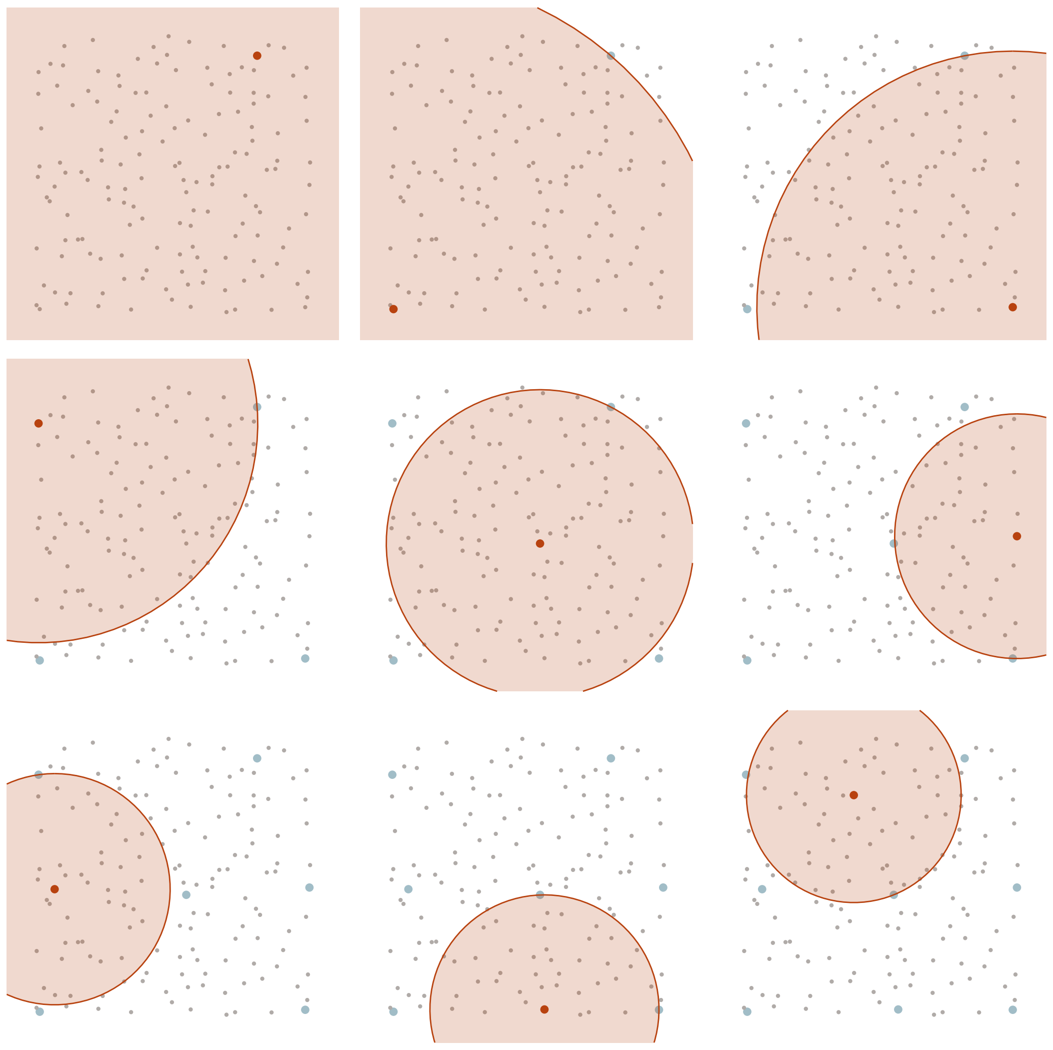


Figure 3.4: **Maximin ordering.** The first nine elements of the maximin ordering on a pointset in $\mathbb{R}^2$, with the asocciated length scale ℓ_i visualized as a shaded radius.

Based on our observation in Figure 3.3, we expect the conditional correlations to decay on a scale ℓ_k after conditioning on the first k points in the maximin ordering. This suggests to use the maximin sparsity pattern illustrated in Figure 3.5 where we include only interactions of x_{i_k} with points within a distance $\approx \ell_k$.

Definition 5 (Maximin sparsity pattern)**.** *Given a maximin ordering of the point set* $\{x_i\}_{i \in I} \subset \mathbb{R}^d$ *with length scales* $\{\ell_k\}_{1 \leq k \leq N}$ *and a sparsity parameter* $\rho \in \mathbb{R}_+$*, we define its sparsity set* $S \subset I \times I$ *as* $S \coloneqq \left\{(i_k, i_l) \text{ s. t. } k > l \text{ and } \operatorname{dist}\left(x_{i_k}, x_{i_l}\right) \leq \rho \ell_k\right\}$.

The entries outside of the maximin sparsity pattern will be of small magnitude but not zero. Therefore a trade-off between accuracy and computational cost needs to be chosen by varying ρ. As we increasing ρ, under mild assumptions, the size of the

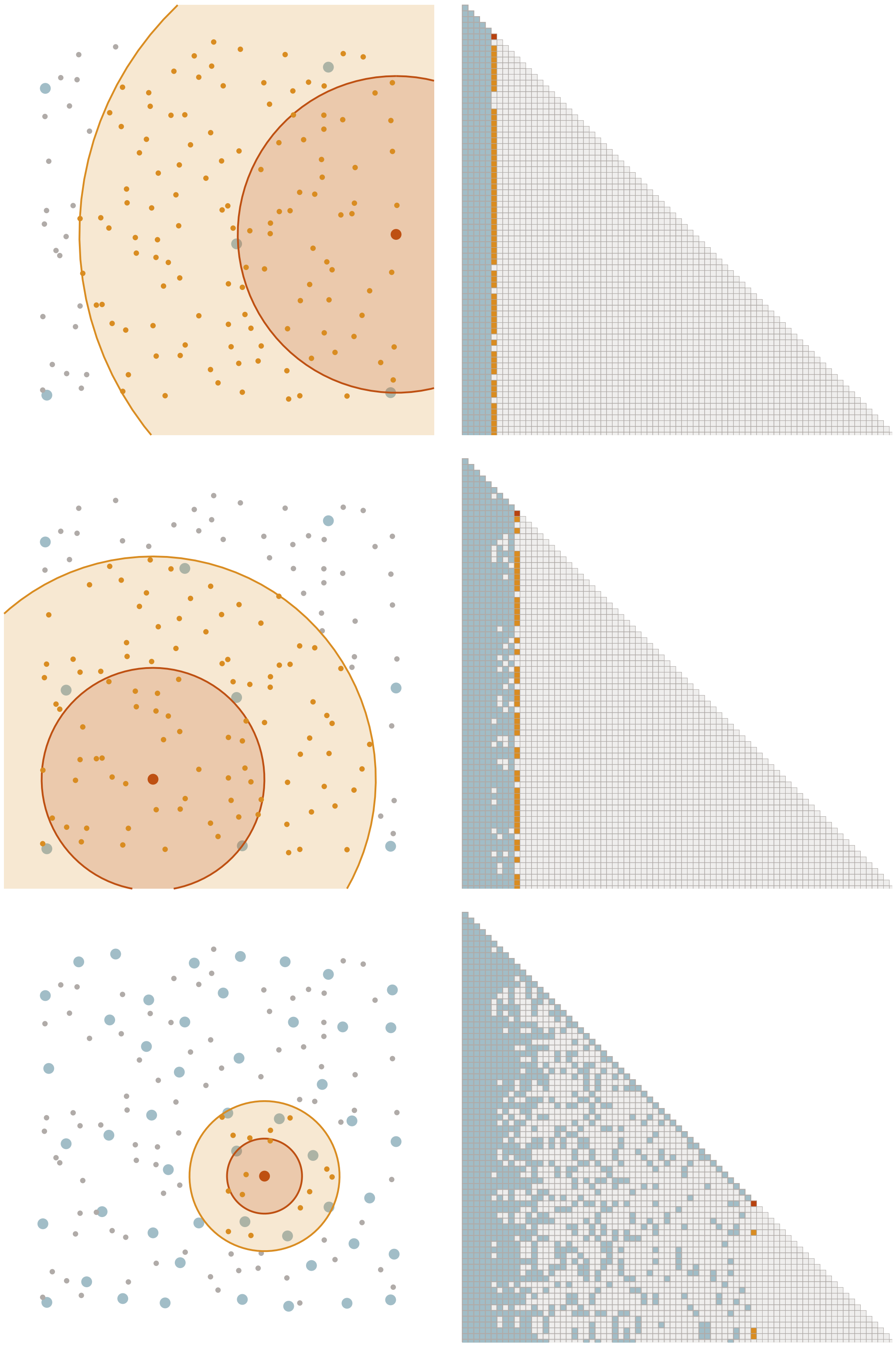


Figure 3.5: **Maximin sparsity pattern.** Each column of the maximin sparsity pattern includes interactions with points that are within a factor ρ of the corresponding length scale ℓ_k.

sparsity pattern will grow as $\mathcal{O}\left(N\log(N)\rho^d\right)$, allowing us to compute the factors in time $\mathcal{O}\left(N\log^2(N)\rho^{2d}\right)$, as detailed in Chapter 5. As we will discuss in Chapter 4, the approximation error ϵ will usually decay exponentially as $\log(\epsilon) \lessapprox \log(N) - \rho$, allowing us to compute an ϵ approximation in complexity $\mathcal{O}\left(N\log^2(N)\log(N/\epsilon)\right)$.

We have seen in Section 3.3 that when factorizing the precision matrix $A = \Theta^{-1}$, sparsity of a given column is equivalent to independence of the Gaussian process conditional on degrees of freedom *after* k in the elimination ordering. Thus we factorize the *precision matrices* of smooth Gaussian processes by using the *reverse* maximin ordering as elimination ordering.

Definition 6 (Reverse maximin ordering)**.** *A reverse maximin ordering of a point set $\{x_i\}_{i\in I} \subset \mathbb{R}^d$ with starting set C is obtained by reverting a maximin ordering of $\{x_i\}_{i\in I} \subset \mathbb{R}^d$ with starting set C. The associated ℓ_k is then obtained as the $(\#I + 1 - k)$-th length scale of the maximin ordering.*

As before, the associated sparsity pattern only contains interactions between points within distance ρ times the associated length scale.

Definition 7 (Reverse maximin sparsity pattern)**.** *Given a reverse maximin ordering of the point set $\{x_i\}_{i\in I} \subset \mathbb{R}^d$ with length scales $\{\ell_k\}_{1\leq k\leq N}$ and a sparsity parameter $\rho \in \mathbb{R}_+$, its sparsity set $S \subset I\times I$ is $S := \left\{(i_k, i_l) \text{ s. t. } k > l \text{ and } \operatorname{dist}\left(x_{i_k}, x_{i_l}\right) \leq \rho\ell_k\right\}$.*

However, as shown in Figure 3.6, there is an important difference. Since each degree of freedom only interacts with others later in the ordering and the largest length scales now appear *last* in the ordering, the reverse maximin sparsity pattern has only $\mathcal{O}\left(N\rho^d\right)$ entries. As we will see in Chapters 5 and 6, this allows to compute the Cholesky factors of A in only $\mathcal{O}\left(N\rho^{2d}\right)$ time.

3.6 Cholesky factorization, numerical homogenization, and *gamblets*

3.6.1 The maximin ordering as a multiresolution method

In Section 3.5, we have introduced the maximin ordering in terms of a sequential maximum distance selection. Instead, we now interprete it as a multiresolution basis transform. To this end, we choose a scale factor $h \in (0,1)$ and define the index sets $I^{(k)} := \left\{i : h^k \leq \ell_i/\ell_1\right\}$. As illustrated in Figure 3.8, the subspaces $V^{(k)} := \operatorname{span}\{\mathbf{1}_i\}_{i\in I^{(k)}}$, with $\mathbf{1}_i$ being the i-th standard basis vector, then form a multiresolution sequence of subspaces of $\mathbb{R}^{I^{(q)}} \cong \mathbb{R}^N$ in the sense of [162]:

$$0 = V^{(0)} \subset V^{(1)} \subset \ldots \subset V^{(q-1)} \subset V^{(q)} = \mathbb{R}^I. \tag{3.7}$$

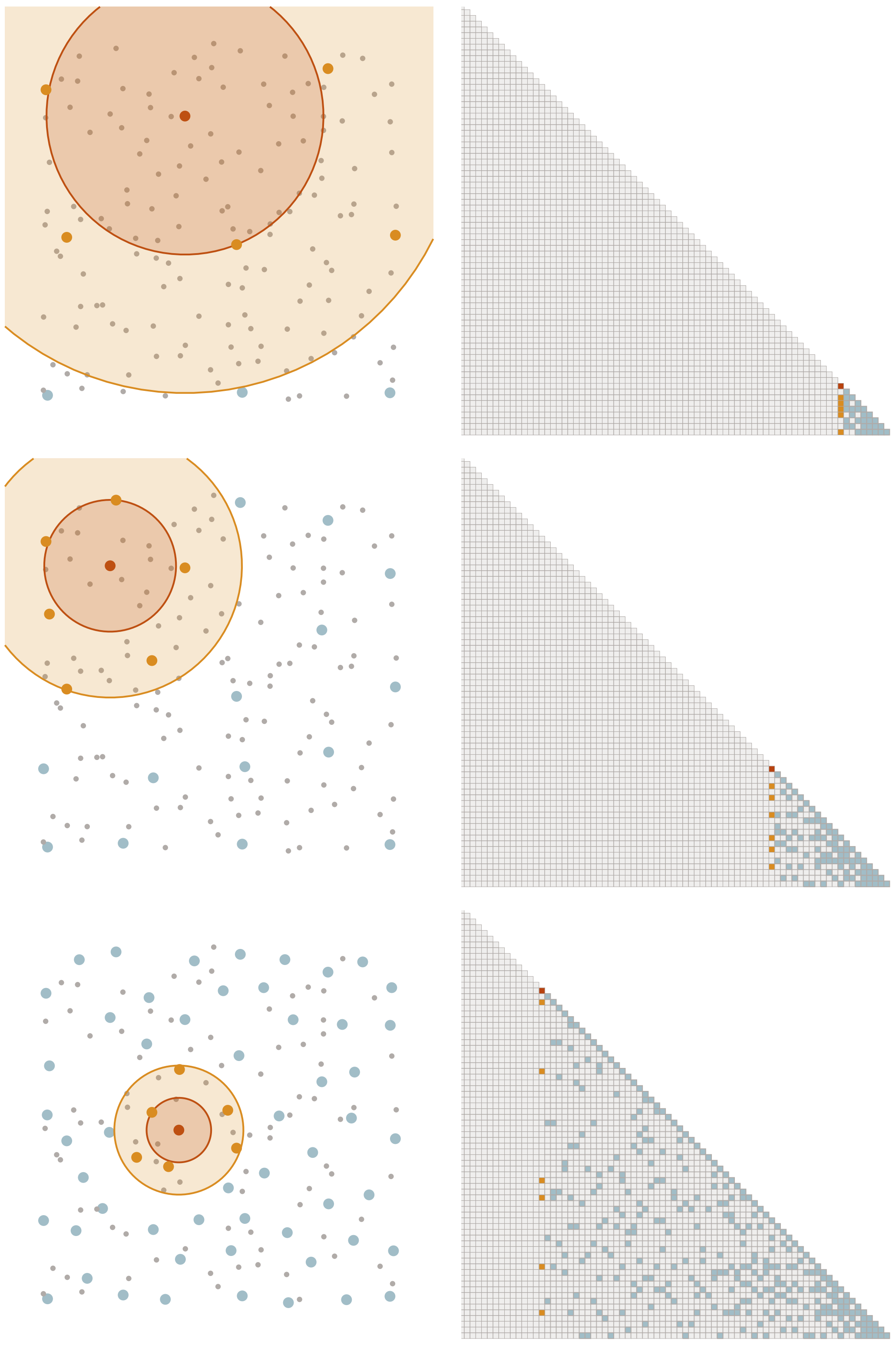


Figure 3.6: **Reverse maximin sparsity pattern.** Each column of the reverse maximin sparsity pattern includes interactions within a factor ρ of the corresponding length scale ℓ_k. The number of nonzeros per column is approximately constant.

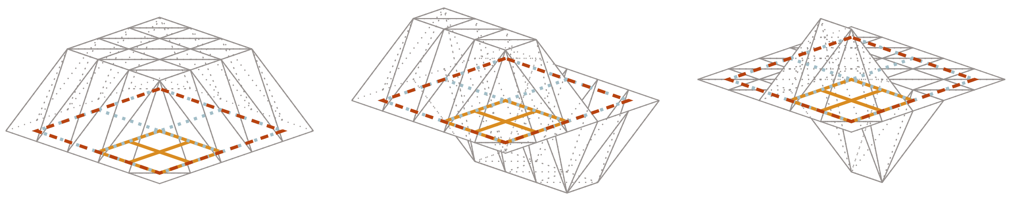


Figure 3.7: **A Haar-type multiresolution basis.** We begin the construction by forming averages on different scales. On all but the coarsest scale, we obtain the basis function functions as linear combination of nearby averages on a given scale that are chosen to be orthogonal to all averages on the coarser scale. In the Figure, we show basis functions on the three coarsest scales.

By computing the orthogonal complement of $V^{(k-1)}$ in $V^{(k)}$ for $1 \leq k \leq q$, we obtain the orthogonal subspaces $W^{(k)}$ that form an orthogonal multiresolution splitting of $V^{(q)} = \mathbb{R}^N$ in the sense that for all $1 \leq k \leq q$ we have $I^{(k)} = \bigoplus_{1 \leq l \leq k} J^{(k)}$.

Based on the above, the maximin ordering can be thought of as a rudimentary multiresolution basis, a generalization of the so-called "lazy wavelets" to multivariate, irregularly sampled data. Instead of using subsampling with density $\approx h^{-dk}$, we could have obtained $V^{(k)}$ by forming averages over regions of size $\approx h^k$ and obtained $W^{(k)}$ as orthogonal complement of $V^{(k-1)}$ in $V^{(k)}$, resulting in a Haar-type multiresolution basis as illustrated in Figure 3.7.

Analogs of the screening effect illustrated in Figure 3.3 hold true for a wide range of multiresolution systems in the sense that after measuring a smooth Gaussian process against the elements of $V^{(k)}$, its conditional correlations decay rapidly on the scale h^k of the measurements. In particular, the Cholesky factors of Green's and stiffness matrices of elliptic PDEs have almost sparse Cholesky factors, when represented in a multiresolution basis and using a coarse to fine (Green's matrix) or fine to coarse (stiffness matrix) elimination ordering. The connection to multiresolution analysis will also allow us to use tools from numerical homogenization to provide rigorous proofs of the screening effect.

3.6.2 Cholesky, Nystrom, and numerical homogenization

Let us now represent the Green's matrix Θ and its inverse A in a two-scale basis with the first block $\Theta_{1,1}, A_{1,1}$ representing the coarse scale and the second block $\Theta_{2,2}, A_{2,2}$ representing the fine scale, and $\Theta_{12}, \Theta_{21}, A_{12}, A_{21}$ representing the interactions between the two scales.

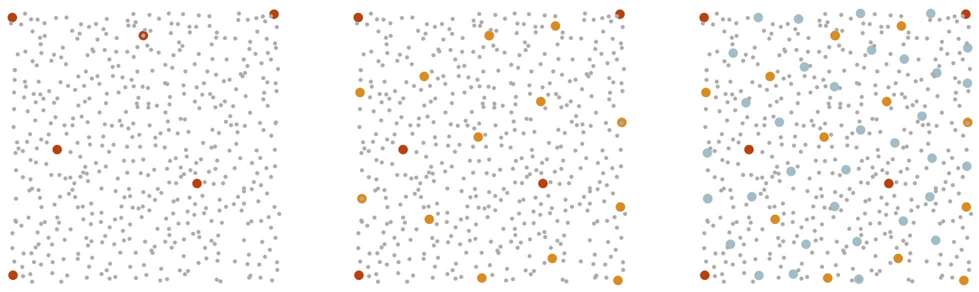


Figure 3.8: **The hidden multiscale structure of the maximin ordering.** The index sets $I^{(k)} := \left\{i : h^k \leq \ell_i/\ell_1\right\}$ of the maximin ordering can be interpreted as the scale spaces $V^{(k)}$ of a multiresolution basis. The index sets $J^{(k)}) := I^{(k)} \setminus I^{(k-1)}$ can then be viewed as the resulting orthogonal multiresolution decomposition. In this figure, from left to right, we display $I^{(1)}$, $I^{(2)}$, and $I^{(3)}$, plotting $J^{(1)}$ in red, $J^{(2)}$ in orange, and $J^{(3)}$ in blue.

An important problem in numerical analysis is to compute a low-rank approximation of Θ that correctly captures the coarse-scale behavior. Given access to Θ, this problem can be solved by using the "Nystrom" approximation

$$\begin{pmatrix} \Theta_{1,1} & \Theta_{1,2} \\ \Theta_{2,1} & \Theta_{2,2} \end{pmatrix} \approx \begin{pmatrix} \mathrm{Id} \\ \Theta_{2,1}\left(\Theta_{1,1}\right)^{-1} \end{pmatrix} \Theta_{1,1} \left(\mathrm{Id}, \quad \left(\Theta_{1,1}\right)^{-1} \Theta_{12}\right). \tag{3.8}$$

A classical method in numerical analysis [23, Chapter 3], the Nystrom approximation has recently been used to compress kernel matrices arising in machine learning [87, 222, 250]. Following the discussion in Section 3.3, this approximation amounts to assuming that the fine-scale behavior is deterministic, conditional on the coarse-scale behavior. As described in [199], many popular sparse Gaussian process models arise from refinements of this assumption.

In physical problems described by elliptic PDEs, we typically do not have access to Θ, but rather to its inverse A, the stiffness matrix arising from a discretization of the differential operator. The field of *numerical homogenization* is concerned with computing low-rank approximations of Θ that capture its coarse-scale behavior, from access only to A. At the same time, it tries to preserve (some of) the sparsity of the stiffness matrix A that arises from the locality of the partial differential operator.

The seminal work of [163] solves this problem by constructing an operator-adapted set of basis functions that achieve (up to a constant factor) the optimal discretization error while being (up to an exponentially small error) localized in space.

To understand the relationship of numerical homogenization to sparse Cholesky factorization, we remind ourselves of the linear algebraic identity (see Lemma 4.15, we give indexing $[\,\cdot\,]_{i,j}$ precedence over inversion $[\,\cdot\,]^{-1}$)

$$\begin{pmatrix}\Theta_{1,1} & \Theta_{1,2}\\ \Theta_{2,1} & \Theta_{2,2}\end{pmatrix} = \left(\begin{pmatrix}\mathrm{Id} & A_{1,2}A_{2,2}^{-1}\\ 0 & \mathrm{Id}\end{pmatrix}\begin{pmatrix}\left(A_{1,1} - A_{1,2}A_{2,2}^{-1}A_{2,1}\right) & 0\\ 0 & A_{2,2}\end{pmatrix}\begin{pmatrix}\mathrm{Id} & 0\\ A_{2,2}^{-1}A_{2,1} & \mathrm{Id}\end{pmatrix}\right)^{-1} \tag{3.9}$$

$$= \begin{pmatrix}\mathrm{Id} & 0\\ A_{2,2}^{-1}A_{2,1} & \mathrm{Id}\end{pmatrix}^{-1}\begin{pmatrix}\left(A_{1,1} - A_{1,2}A_{2,2}^{-1}A_{2,1}\right)^{-1} & 0\\ 0 & A_{2,2}^{-1}\end{pmatrix}\begin{pmatrix}\mathrm{Id} & A_{1,2}A_{2,2}^{-1}\\ 0 & \mathrm{Id}\end{pmatrix}^{-1} \tag{3.10}$$

$$= \begin{pmatrix}\mathrm{Id} & 0\\ -A_{2,2}^{-1}A_{2,1} & \mathrm{Id}\end{pmatrix}\begin{pmatrix}\left(A_{1,1} - A_{1,2}A_{2,2}^{-1}A_{2,1}\right)^{-1} & 0\\ 0 & A_{2,2}^{-1}\end{pmatrix}\begin{pmatrix}\mathrm{Id} & -A_{1,2}A_{2,2}^{-1}\\ 0 & \mathrm{Id}\end{pmatrix}. \tag{3.11}$$

We can now recover the Nystrom approximation as

$$\begin{pmatrix}\Theta_{1,1} & \Theta_{1,2}\\ \Theta_{2,1} & \Theta_{2,2}\end{pmatrix} \approx \begin{pmatrix}\mathrm{Id}\\ \Theta_{2,1}\left(\Theta_{1,1}\right)^{-1}\end{pmatrix}\Theta_{1,1}\left(\mathrm{Id}, \quad \left(\Theta_{1,1}\right)^{-1}\Theta_{12}\right) \tag{3.12}$$

$$= \begin{pmatrix}\mathrm{Id}\\ -A_{2,2}^{-1}A_{2,1}\end{pmatrix}\left(A_{1,1} - A_{1,2}A_{2,2}^{-1}A_{2,1}\right)^{-1}\left(\mathrm{Id}, \quad -A_{1,2}A_{22}^{-1}\right) \tag{3.13}$$

$$= \Psi\left(\Psi^{\top}A\Psi\right)^{-1}\Psi^{\top}, \quad \text{for } \Psi := \begin{pmatrix}\mathrm{Id}\\ -A_{2,2}^{-1}A_{2,1}\end{pmatrix}. \tag{3.14}$$

If we interpret the columns of Ψ as a new set of *operator adapted finite element functions*, obtain the Nystrom approximation by projecting the problem onto the space of these adaptive finite element functions, inverting the resulting stiffness matrix, and expressing the resulting solution again in terms the original set of finite element functions. The authors of [163] show that for a two-scale splitting similar to the one in Figure 3.7 and for A being the stiffness matrix of a second-order elliptic PDE with L^{∞}-coefficients, the following holds:

1. The matrices Ψ and $\Psi^{\top}A\Psi$ are sparse up to exponentially small terms (*localization*).

2. As we decrease the scale of the small-scale, and hence increase the dimension of Ψ, the resulting Nystrom approximation of Θ improves with the optimal rate (*homogenization*).

3.6.3 Gamblets as Block-Cholesky factorization

The work of [163] provides an efficient, near-optimal coarse-graining of A, but it does not provide us with a fast solver. In fact, computing Ψ requires inverting the matrix $A_{2,2}$, which could be almost as difficult as inverting A.

Motivated by ideas from game and decision theory, as well as Gaussian process regression, [188] extend the construction of Ψ in the last section to multiple scales. The resulting family of operator-adapted wavelets, called *gamblets*, can be constructed in near-linear time. Once constructed, it can be used to invert A in near-linear time.

In order to relate gamblets to Cholesky factorization, we assume that we are given an orthogonal multiresolution decomposition $\left\{W^{(k)}\right\}_{1\leq k\leq q}$ and that for $1 \leq k, l \leq q$, the matrix blocks $\Theta_{k,l}, A_{k,l}$ represent the restriction of Θ, A onto $W^{(k)} \times W^{(l)}$. A multiscale extension of Equation 3.9 can then be obtained (see Lemma 1) by writing

$$\Theta = \Psi B^{-1} \Psi^{\top} = \Psi \left(\Psi^{\top} A \Psi\right)^{(-1)} \Psi^{\top} \tag{3.15}$$

with

$$B := \begin{pmatrix} B^{(1)} & 0 & \dots & 0 \\ 0 & B^{(2)} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & B^{(q)} \end{pmatrix}, \Psi := \begin{pmatrix} \mathrm{Id} & \dots & \dots & 0 \\ B^{(2),-1} A^{(2)}_{2,1} & \ddots & 0 & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ B^{(q),-1} A^{(q)}_{q,1} & \dots & B^{(q),-1} A^{(q)}_{q,q-1} & \mathrm{Id} \end{pmatrix}^{-1}, \tag{3.16}$$

where $A^{(k)} := \left(\Theta_{1:k,1:k}\right)$ and $B^{(k)} := A^{(k)}_{k,k}$.

The columns of Ψ represent the gamblet basis functions, and B is the (block-diagonal) stiffness matrix obtained when representing A in the gamblet basis. In the same setting as [163], [188] prove that

1. The matrices Ψ and $\Psi^{\top} A \Psi$ are sparse up to exponentially small terms (*localization*).
2. The approximation of A obtained by using only the leading k block-columns of Ψ improves at the optimal rate when increasing k (*homogenization*).
3. The matrices $B^{(k)}$ have uniformly bounded condition numbers.

Using and extending proof techniques of [142], [189, 190] extend these results to higher order operators and more general classes of multiresolution splittings $\left\{W^{(k)}\right\}_{1\leq k\leq q}$. We build upon these results in order to prove the sparsity of the

Cholesky factors of Θ. To this end, the main analytic contribution of our work is an extension of the homogenization estimates of gamblets to simplistic multiresolution splittings such as the one implied by the maximin ordering.

Chapter 4

PROVING EXPONENTIAL DECAY OF CHOLESKY FACTORS

4.1 Overview

In Chapter 3, we have used the intuition provided by the screening effect to obtain elimination orderings and sparsity patterns that lead to *near-linear* complexity solvers based on sparse Cholesky factorization. But the resulting sparse Cholesky factor merely *approximates* the original Θ or A, with the approximation accuracy depending on ρ.

This raises the question at what rate the error decays when increasing ρ? How strong is the screening effect, and when does it hold? Despite numerous attempts [26, 227, 228], a rigorous understanding has proven largely elusive. Existing results (see [228] for an overview) are asymptotic and do not provide explicit rates.

A central contribution of this thesis is to provide proof of an exponential screening effect for Gaussian processes with covariance functions given as Green's function of elliptic partial differential equations. This allows us to prove that up to exponentially small errors, the Cholesky factors of Θ and its inverse A can be approximated by sparse matrices with only $O\left(N\log(N)\rho^d\right)$ and $O\left(N\rho^d\right)$ nonzero entries for an approximation error ϵ satisfying $\log(\epsilon) \lessapprox \log(N) - \rho$. These results, which are summarized in Section 4.6, can be specialized to:

Theorem 1. *Let $\Omega \in \mathbb{R}^d$ be a Lipschitz-bounded domain and let $\mathcal{L}$ be a linear elliptic partial differential operator of order $2s > d$. Let $\{x_i\}_{1\leq i \leq N}$ be roughly uniformly distributed in Ω and ordered in the maximin [reverse maximin] ordering. For $\mathcal{G} = \mathcal{L}^{-1}$ the Dirichlet Green's function define, $\Theta_{ij} := \mathcal{G}\left(x_i, x_j\right)$ and let L be the Cholesky factor of Θ [$A := \Theta^{-1}$]. Let $S \subset \{1, \dots, N\}^2$ be the maximin [reverse maximin] sparsity pattern with starting set $\partial\Omega$ and sparsity parameter ρ. Defining $\left(L^S\right)_{ij} := \mathbb{1}_{(i,j)\in S} L_{ij}$, we then have*

$$\log\left(\left\|L^S L^{S,\top} - LL^\top\right\|_{\mathrm{Fro}}\right) \lessapprox \log(N) - \rho. \tag{4.1}$$

A key step for proving this result is the derivation of a novel low-rank approximation result that is interesting in its own right.

Theorem 2. *In the setting of Theorem 1, let $L_{:,1:k}$ be the rank k matrix defined by the first k columns of the Cholesky factor L of Θ in the maximin ordering. Denoting as $\|\cdot\|$ the operator norm and as ℓ_k the length scales of the ordering, we have*

$$\left\|\Theta - L_{(:,1:k)}\left(L_{:,1:k}\right)^{\top}\right\| \lessapprox \|\Theta\| k^{-2s/d}. \tag{4.2}$$

Following the discussion in Section 3.6.2, this low-rank approximation also results in a numerical homogenization results for $\mathcal{L}$.

Theorem 3. *In the setting of Theorem 1, let $L_{:,1:k}$ be the rank k matrix defined by the first k columns of the Cholesky factor L of $A \coloneqq \Theta$ in the reverse maximin ordering. We then have*

$$\left\|\Theta - \Psi(L_{1:k,1:k}L_{1:k,1:k}^{\top})^{-1}\Psi^{\top}\right\| \lessapprox \|\Theta\| k^{-2s/d}, \quad \textit{for} \quad \Psi = \begin{pmatrix} \mathrm{Id} \\ -L_{(k+1):N,1:k}L_{1:k,1:k}^{-1} \end{pmatrix}. \tag{4.3}$$

Here, it is important to note that the resulting approximation of Θ can be efficiently applied to a vector using matrix-vector multiplication of L and substitution (see Algorithm 2) in $L_{1:k,1:k}$, without forming another matrix.

Rigorous forms of the above theorems are stated and proved as Theorems 11, 12 and 13 in Section 4.6.

For $s \leq d/2$, Theorems 2 and 3 are false, and while Theorem 1 seems to hold empirically, a proof of it remains elusive. However, following the discussion in Section 3.6.1, we will prove analogues of Theorems 1, 2, and 3 where the maximin ordering is replaced by a simple averaging-based multiresolution scheme.

While we present our results in terms of the exact Green's matrix, analog results for the approximate Green's matrix Θ_{discrete} obtained as the inverse of a Galerkin discretization A_{discrete} of $\mathcal{L}$ using locally supported basis functions could be obtained by repeating the proof in this setting.

4.2 Setting and notation

4.2.1 The class of elliptic operators

For our rigorous, a priori, complexity-vs.-accuracy estimates, we assume that $\mathcal{G}$ is the Green's function of an elliptic operator $\mathcal{L}$ of order $2s$ ($s, d \in \mathbb{N}$), defined on a bounded Lipschitz domain $\Omega \subset \mathbb{R}^d$, and acting on $H_0^s(\Omega)$, the Sobolev space of (zero boundary value) functions having derivatives of order s in $L^2(\Omega)$. More

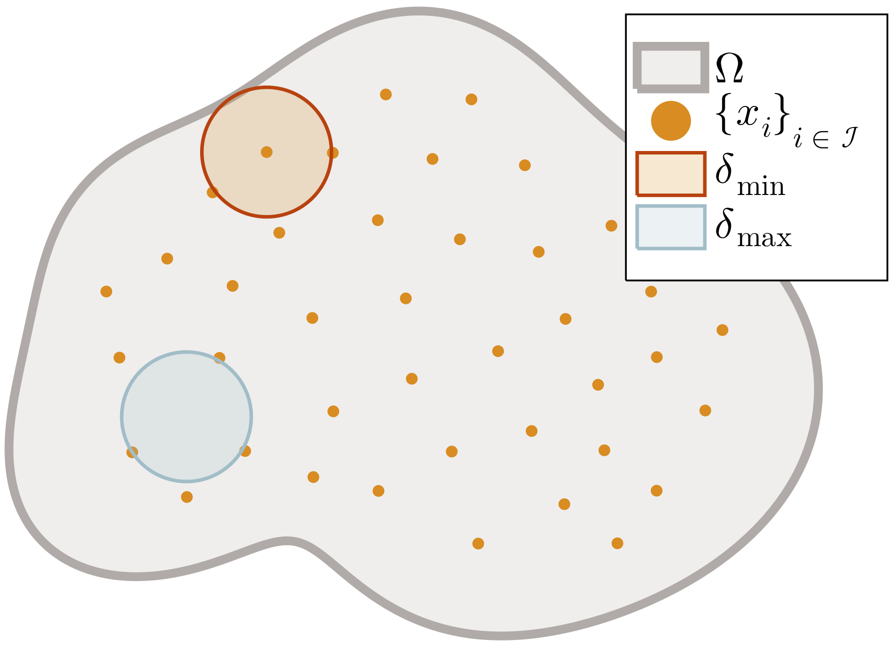


Figure 4.1: **A regularity criterion.** We measure the regularity of the distributions of measurement points as the ration of $\delta_{\min}$, the smallest distance between neighboring points or points and the boundary, and $\delta_{\max}$, the radius of the largest ball that does not contain any points.

precisely, writing $H^{-s}(\Omega)$ for the dual space of $H^s_0(\Omega)$ with respect to the $L^2(\Omega)$ scalar product, our rigorous estimates will be stated for an arbitrary linear bijection

$$\mathcal{L} \colon H^s_0(\Omega) \to H^{-s}(\Omega) \tag{4.4}$$

that is *symmetric* (i.e. $\int_\Omega u \mathcal{L} v \, \mathrm{d}x = \int_\Omega v \mathcal{L} u \, \mathrm{d}x$), *positive* (i.e. $\int_\Omega u \mathcal{L} u \, \mathrm{d}x \geq 0$), and *local* in the sense that

$$\int_\Omega u \mathcal{L} v \, \mathrm{d}x = 0 \text{ for all } u, v \in H^s_0(\Omega) \text{ such that } \operatorname{supp} u \cap \operatorname{supp} v = \emptyset. \tag{4.5}$$

Let $\|\mathcal{L}\| := \sup_{u \in H^s_0} \|\mathcal{L} u\|_{H^{-s}} / \|u\|_{H^s_0}$ and $\|\mathcal{L}^{-1}\| := \sup_{f \in H^{-s}} \|\mathcal{L}^{-1} f\|_{H^s_0} / \|f\|_{H^{-s}}$ denote the operator norms of $\mathcal{L}$ and $\mathcal{L}^{-1}$. The complexity and accuracy estimates for our algorithm will depend on (and only on) d, s, Ω, $\|\mathcal{L}\|$, $\|\mathcal{L}^{-1}\|$, and the parameter

$$\delta := \frac{\delta_{\min}}{\delta_{\max}} := \frac{\min_{i \neq j \in I} \operatorname{dist}\left(x_i, \{x_j\} \cup \partial\Omega\right)}{\max_{x \in \Omega} \operatorname{dist}\left(x, \{x_i\}_{i \in I} \cup \partial\Omega\right)}, \tag{4.6}$$

the geometric meaning of which is illustrated in Figure 4.1.

4.2.2 Discretization in the abstract

Before talking about computation, we need to *discretize* the infinite-dimensional spaces $H_0^s(\Omega)$ and $H^{-s}(\Omega)$ by approximating them with finite vector spaces. We first introduce this procedure in the abstract.

For $\mathcal{B}$ a separable Banach space with dual space $\mathcal{B}^*$ (such as $H_0^s(\Omega)$ and $H^{-s}(\Omega)$), we write $[\cdot,\cdot]$ for the duality product between $\mathcal{B}^*$ and $\mathcal{B}$. Let $\mathcal{L}\colon \mathcal{B} \to \mathcal{B}^*$ be a linear bijection and let $\mathcal{G} := \mathcal{L}^{-1}$. Assume $\mathcal{L}$ to be symmetric and positive (i.e. $[\mathcal{L}u, v] = [\mathcal{L}v, u]$ and $[\mathcal{L}u, u] \geq 0$ for $u, v \in \mathcal{B}$). Let $\|\cdot\|$ be the quadratic (energy) norm defined by $\|u\|^2 := [\mathcal{L}u, u]$ for $u \in \mathcal{B}$ and let $\|\cdot\|_*$ be its dual norm defined by

$$\|\phi\|_* := \sup_{0 \neq u \in \mathcal{B}} \frac{[\phi, u]}{\|u\|} = [\phi, \mathcal{G}\phi] \text{ for } \phi \in \mathcal{B}^*. \tag{4.7}$$

Let $\{\phi_i\}_{i\in I}$ be linearly independent elements of $\mathcal{B}^*$ (known as *measurement functions*) and let $\Theta \in \mathbb{R}^{I\times I}$ be the symmetric positive-definite matrix defined by

$$\Theta_{ij} := [\phi_i, \mathcal{G}\phi_j] \quad \text{for } i, j \in I. \tag{4.8}$$

We assume that we are given $q \in \mathbb{N}$ and a partition $I = \bigcup_{1\leq k\leq q} J^{(k)}$ of I. We represent $I \times I$ matrices as $q \times q$ block matrices according to this partition. Given an $I \times I$ matrix M, we write $M_{k,l}$ for the $(k,l)^{\text{th}}$ block of M, and $M_{k_1:k_2,l_1:l_2}$ for the sub-matrix of M defined by blocks ranging from k_1 to k_2 and l_1 to l_2. Unless specified otherwise, we write L for the lower-triangular Cholesky factor of Θ and define

$$\Theta^{(k)} := \Theta_{1:k,1:k}, \qquad A^{(k)} := \Theta^{(k),-1}, \qquad B^{(k)} := A^{(k)}_{k,k} \qquad \text{for } 1 \leq k \leq q. \tag{4.9}$$

We interpret the $\{J^{(k)}\}_{1\leq k\leq q}$ as labelling a hierarchy of scales with $J^{(1)}$ representing the coarsest and $J^{(q)}$ the finest. We write $I^{(k)}$ for $\bigcup_{1\leq k'\leq k} J^{(k')}$.

Throughout this section, we assume that the ordering of the set I of indices is compatible with the partition $I = \bigcup_{k=1^q} J^{(k)}$, i.e. $k < l$, $i \in J^{(k)}$ and $j \in J^{(l)}$ together imply $i \prec j$. We will write L or $\operatorname{chol}(\Theta)$ for the Cholesky factor of Θ in that ordering.

4.2.3 Discretization of $H_0^s(\Omega)$ and $H^{-s}(\Omega)$

While similar results are true for a wide range of measurements $\{\phi_i\} \in \mathcal{B} = H^{-s}(\Omega)$ we will restrict our attention to two archetypical examples given by pointwise evaluation and nested averages.

We will assume (without loss of generality after rescaling) that $\operatorname{diam}(\Omega) \leq 1$. As described in Figure 3.8, successive points of the maximin ordering can be gathered into levels so that after appropriate rescaling of the measurements, the Cholesky factorization in the maximin ordering falls in the setting of Example 1.

Example 1. *Let $s > d/2$. For $h, \delta \in (0,1)$ let $\{x_i\}_{i \in I^{(1)}} \subset \{x_i\}_{i \in I^{(2)}} \subset \cdots \subset \{x_i\}_{i \in I^{(q)}}$ be a nested hierarchy of points in Ω that are homogeneously distributed at each scale in the sense of the following three inequalities:*

(1) $\sup_{x \in \Omega} \min_{i \in I^{(k)}} |x - x_i| \leq h^k$,

(2) $\min_{i \in I^{(k)}} \inf_{x \in \partial\Omega} |x - x_i| \geq \delta h^k$, *and*

(3) $\min_{i,j \in I^{(k)}: i \neq j} |x_i - x_j| \geq \delta h^k$.

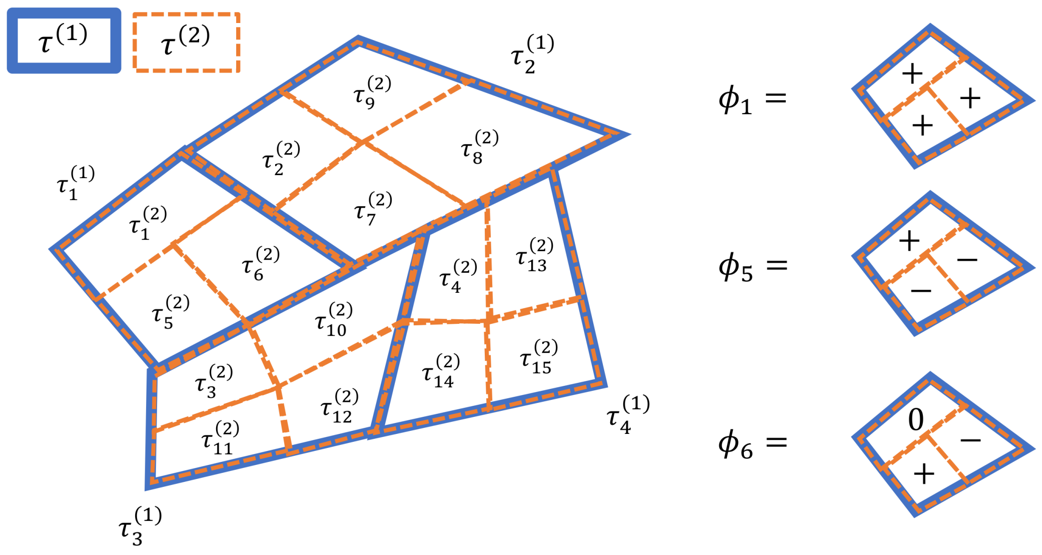
Let $J^{(1)} := I^{(1)}$ and $J^{(k)} := I^{(k)} \setminus I^{(k-1)}$ for $k \in \{2, \dots, q\}$. Let $\boldsymbol{\delta}$ denote the unit Dirac delta function and choose

$$\phi_i := h^{\frac{kd}{2}} \boldsymbol{\delta}(x - x_i) \text{ for } i \in J^{(k)} \text{ and } k \in \{1, \dots, q\}. \tag{4.10}$$

The discretization chosen in Example 1 is not applicable for $s < d/2$ since in this case, functions in $H_0^s(\Omega)$ are not defined point-wise and thus $\boldsymbol{\delta} \notin \mathcal{B}^*$. A possible alternative is to replace $\phi_i := h^{\frac{kd}{2}} \boldsymbol{\delta}(x - x_i)$ with $\phi_i := h^{-\frac{kd}{2}} \mathbf{1}_{B_h(0)}(x - x_i)$. However, while the exponential decay result in Theorem 4 seems to be true empirically for this choice of measurements, we are unable to prove it for $s < d/2$. Furthermore, the numerical homogenization result of Theorem 7 is *false* for this choice of measurements and $s < d/2$. However, our results can still be recovered by choosing measurements obtained as a hierarchy of local averages.

Given subsets $\tilde{I}, \tilde{J} \subset I$, we extend a matrix $M \in \mathbb{R}^{\tilde{I} \times \tilde{J}}$ to an element of $\mathbb{R}^{I \times J}$ by padding it with zeros.

Example 2. *(See Figure 4.2.) For $h, \delta \in (0,1)$, let $(\tau_i^{(k)})_{i \in I^{(k)}}$ be uniformly Lipschitz convex sets forming a regular nested partition of Ω in the following sense. For $k \in \{1, \dots, q\}$, $\Omega = \bigcup_{i \in I^{(k)}} \tau_i^{(k)}$ is a disjoint union except for the boundaries. $I^{(k)}$ is a nested set of indices, i.e. $I^{(k)} \subset I^{(k+1)}$ for $k \in \{1, \dots, q-1\}$. For $k \in \{2, \dots, q\}$ and $i \in I^{(k-1)}$, there exists a subset $c_i \subset I^{(k)}$ such that $i \in c_i$ and $\tau_i^{(k-1)} = \bigcup_{j \in c_i} \tau_j^{(k)}$. Assume that each $\tau_i^{(k)}$ contains a ball $B_{\delta h^k}(x_i^{(k)})$ of center $x_i^{(k)}$ and radius δh^k, and is contained in the ball $B_{h^k}(x_i^{(k)})$. For $k \in \{2, \dots, q\}$ and $i \in$*



$I^{(1)} = \{1, \dots, 4\}$ $\quad I^{(2)} = \{1, \dots, 15\}$ $\quad J^{(1)} = \{1, \dots, 4\}$ $\quad J^{(2)} = \{5, \dots, 15\}$

Figure 4.2: **Hierarchical averaging.** We illustrate the construction described in Example 2 in the case $q = 2$. On the left we see the nested partition of the domain, and on the right we see (the signs of) a possible choice for ϕ_1, ϕ_5, and ϕ_6.

$I^{(k-1)}$, let the submatrices $\mathfrak{w}^{(k),i} \in \mathbb{R}^{(c_i \setminus \{i\}) \times c_i}$ satisfy $\sum_{j \in c_i} \mathfrak{w}^{(k),i}_{m,j} \mathfrak{w}^{(k),i}_{n,j} |\tau_j^{(k)}| = \delta_{mn}$ and $\sum_{j \in c_i} \mathfrak{w}^{(k),i}_{l,j} |\tau_j^{(k)}| = 0$ for each $l \in c_i \setminus \{i\}$, where $|\tau_i^{(k)}|$ denotes the volume of $\tau_i^{(k)}$. Let $J^{(1)} := I^{(1)}$ and $J^{(k)} := I^{(k)} \setminus I^{(k-1)}$ for $k \in \{2, \dots, q\}$. Let $W^{(1)}$ be the $J^{(1)} \times I^{(1)}$ matrix defined by $W^{(1)}_{ij} := \delta_{ij}$. Let $W^{(k)}$ be the $J^{(k)} \times I^{(k)}$ matrix defined by $W^{(k)} := \sum_{i \in I^{(k-1)}} \mathfrak{w}^{(k),i}$ for $k > 2$, where we set

$$\phi_i := h^{-kd/2} \sum_{j \in I^{(k)}} W^{(k)}_{i,j} \mathbf{1}_{\tau_j^{(k)}} \quad \textit{for each } i \in J^{(k)} \tag{4.11}$$

and define $[\phi_i, u] := \int_\Omega \phi_i u \,\mathrm{d}x$. In order to keep track of the distance between the different ϕ_i of Example 2, we choose an arbitrary set of points $\{x_i\}_{i \in I} \subset \Omega$ with the property that $x_i \in \operatorname{supp}(\phi_i)$ for each $i \in I$.

In the above, we have discretized the Green's functions of the elliptic operators resulting in the Green's matrix Θ as the fundamental discrete object. The inverse A of Θ can be interpreted as the stiffness matrix obtained from the Galerkin discretization of $\mathcal{L}$ using the basis given by

$$\psi_i := \sum_j A_{ij} \mathcal{G}(\phi_j) \in \mathcal{B}. \tag{4.12}$$

These types of basis functions are referred to as *gamblets* in the prior works of [188–190] that form the basis for the proofs in this chapter. While exponentially decaying, these basis functions are nonlocal and unknown apriori, hence they cannot be used to discretize a partial differential operator with unknown Green's function. For Θ the inverse of a Galerkin discretization of $\mathcal{L}$ in a local basis, analog results can be obtained by repeating the proofs of Theorems 4 and 7 in the discrete setting.

In the setting of Examples 1 and 2, denoting as L the lower triangular Cholesky factor of Θ or Θ^{-1}, we will show that

$$|L_{ij}| \leq \operatorname{poly}(N) \exp(-\gamma d(i,j)), \tag{4.13}$$

for a constant $\gamma > 0$ and a suitable distance measure $d(\cdot,\cdot)\colon I \times I \longrightarrow \mathbb{R}$.

4.3 Algebraic identities and roadmap

We will use the following block-Cholesky decomposition of Θ to obtain (4.13).

Lemma 1. *We have $\Theta = \bar{L} D \bar{L}^T$, with $\bar{L}$ and D defined by*

$$D := \begin{pmatrix} B^{(1),-1} & 0 & \dots & 0 \\ 0 & B^{(2),-1} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & B^{(q),-1} \end{pmatrix}, \bar{L} := \begin{pmatrix} \mathrm{Id} & \dots & \dots & 0 \\ B^{(2),-1} A^{(2)}_{2,1} & \ddots & 0 & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ B^{(q),-1} A^{(q)}_{q,1} & \dots & B^{(q),-1} A^{(q)}_{q,q-1} & \mathrm{Id} \end{pmatrix}^{-1}. \tag{4.14}$$

In particular, if $\tilde{L}$ is the lower-triangular Cholesky factor of D, then the lower-triangular Cholesky factor L of Θ is given by $L = \bar{L}\tilde{L}$.

Proof. To obtain Lemma 1, we successively apply Lemma 2 to Θ (see Section .1 for details). Lemma 2 summarizes classical identities satisfied by Schur complements. □

Lemma 2 ([258, Chapter 1.1])**.** *Let $\Theta = \left(\begin{smallmatrix} \Theta_{1,1} & \Theta_{1,2} \\ \Theta_{2,1} & \Theta_{2,2} \end{smallmatrix}\right)$ be symmetric positive definite and $A = \left(\begin{smallmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{smallmatrix}\right)$ its inverse. Then*

$$\Theta = \begin{pmatrix} \mathrm{Id} & 0 \\ L_{2,1} & \mathrm{Id} \end{pmatrix} \begin{pmatrix} D_{1,1} & 0 \\ 0 & D_{2,2} \end{pmatrix} \begin{pmatrix} \mathrm{Id} & L_{2,1}^{\top} \\ 0 & \mathrm{Id} \end{pmatrix} \tag{4.15}$$

$$A = \begin{pmatrix} \mathrm{Id} & -L_{2,1}^{\top} \\ 0 & \mathrm{Id} \end{pmatrix} \begin{pmatrix} D_{1,1}^{-1} & 0 \\ 0 & D_{2,2}^{-1} \end{pmatrix} \begin{pmatrix} \mathrm{Id} & 0 \\ -L_{2,1} & \mathrm{Id} \end{pmatrix} \tag{4.16}$$

where

$$L_{2,1} = \Theta_{2,1}\Theta_{1,1}^{-1} = -A_{2,2}^{-1} A_{2,1} \tag{4.17}$$

$$D_{1,1} = \Theta_{1,1} = \left(A_{1,1} - A_{1,2} A_{2,2}^{-1} A_{2,1}\right)^{-1} \tag{4.18}$$

$$D_{2,2} = \Theta_{2,2} - \Theta_{2,1}\Theta_{1,1}^{-1}\Theta_{1,2} = A_{2,2}^{-1}. \tag{4.19}$$

Based on Lemma 1, (4.13) can be established by ensuring that:

(1) the matrices $A^{(k)}$ (and hence also $B^{(k)}$) decay exponentially according to $d(\cdot,\cdot)$;

(2) the matrices $B^{(k)}$ have uniformly bounded condition numbers;

(3) the products of exponentially decaying matrices decay exponentially;

(4) the inverses of well-conditioned exponentially decaying matrices decay exponentially;

(5) the Cholesky factors of the inverses of well-conditioned exponentially decaying matrices decay exponentially; and

(6) if a $q \times q$ block lower-triangular matrix $\bar{L}$ with unit block-diagonal decays exponentially, then so does its inverse.

We will carry out this program in the setting of Examples 1 and 2 and prove that (4.13) holds with

$$d(i,j) := h^{-\min(k,l)} \operatorname{dist}(x_i, x_j), \quad \text{for each } i \in J^{(k)},\, j \in J^{(l)}. \tag{4.20}$$

To prove (1), the matrices $\Theta^{(k)}$, $A^{(k)}$ (interpreted as coarse-grained versions of $\mathcal{G}$ and $\mathcal{L}$), and $B^{(k)}$ will be identified as stiffness matrices of the $\mathcal{L}$-adapted wavelets described in Section 3.6.3. This identification is established on the general identities $\Theta^{(k)}_{i,j} = [\phi_i, \mathcal{G}\phi_j]$ for $i,j \in I^{(k)}$, $A^{(k)} = (\Theta^{(k)})^{-1}$, $A^{(k)}_{i,j} = [\mathcal{L}\psi^{(k)}_i, \psi^{(k)}_j]$ and $B^{(k)}_{i,j} = [\mathcal{L}\chi^{(k)}_i, \chi^{(k)}_j]$ where $\psi^{(k)}_i$ and $\chi^{(k)}_i$ are the gamblets introduced in [190].

4.4 Exponential decay of $A^{(k)}$

Our proof of the exponential decay of L will be based on that of $A^{(k)}$ as expressed in the following condition:

Condition 1. *Let $\gamma, C_\gamma \in \mathbb{R}_+$ be constants such that for $1 \leq k \leq q$ and $i,j \in I^{(k)}$,*

$$\left|A^{(k)}_{ij}\right| \leq C_\gamma \sqrt{A^{(k)}_{ii} A^{(k)}_{jj}} \exp(-\gamma d(i,j)). \tag{4.21}$$

The matrices $A^{(k)}$ are coarse-grained versions of the local operator $\mathcal{L}$ and thus inherit some of its locality in the form of exponential decay. Such exponential localization results were first obtained by [163] for the coarse-grained operators obtained from local orthogonal decomposition (LOD) applied to second-order elliptic PDEs with rough coefficients. [188] gives similar results for measurement functions chosen as in Example 2. [120] extend the results on exponential decay to higher-order operators satisfying a strong ellipticity condition. These results were obtained using similar *mass chasing* techniques that are difficult to extend to general higher-order operators. [142] present a simpler proof of the exponential decay of the LOD basis functions of [163] based on the exponential convergence of subspace iteration methods. [189] extend this technique (by presenting necessary and sufficient conditions expressed as frame inequalities in dual spaces) to elliptic PDEs of arbitrary (integer) order and new classes of (possibly non-conforming) measurements, including those of Examples 1 and 2. More recently, [43] show localization results for the fractional partial differential operators by using the Caffarelli–Silvestre extension. The results of [189] are sufficient to show that Condition 1 holds true in the setting of Examples 1 and 2.

Theorem 4 ([189])**.** *In the setting of Examples 1 and 2, the matrices $A^{(k)}$ satisfy*

$$\left|A_{ij}^{(k)}\right| \leq C_\gamma \sqrt{A_{ii}^{(k)} A_{jj}^{(k)}} \exp\left(-\frac{\gamma}{h^k} \operatorname{dist}\left(\operatorname{supp}\left(\phi_i\right), \operatorname{supp}\left(\phi_j\right)\right)\right). \tag{4.22}$$

This implies that in Example 1, they satisfy

$$\left|A_{ij}^{(k)}\right| \leq C_\gamma \sqrt{A_{ii}^{(k)} A_{jj}^{(k)}} \exp(-\gamma d(i,j)) \tag{4.23}$$

and in Example 2, they satisfy

$$\left|A_{ij}^{(k)}\right| \leq C_\gamma \exp\left(\frac{\gamma}{h}\right) \sqrt{A_{ii}^{(k)} A_{jj}^{(k)}} \exp(-\gamma d(i,j)), \tag{4.24}$$

with the constants C_γ and γ depending only on $\|\mathcal{L}\|$, $\|\mathcal{L}^{-1}\|$, s, d, Ω, and δ. In particular, they satisfy Condition 1 with the constants described above.

Proof. Our Example 1 is equivalent to Example 2.29 of [189]. In [189, Theorem 2.25 and Theorem 2.26], it is shown that in the gamblets $\{\psi_i^{(k)}\}_{i \in I^{(k)}}$ computed in this setting decay exponentially on the length-scale h^k, with respect to the energy norm. By [189, Theorem 3.8], we have $A_{ij}^{(k)} = [\psi_i^{(k)}, \mathcal{L}\psi_j^{(k)}]$ and, therefore, the exponential decay of gamblets implies the exponential decay of the $A^{(k)}$.

We further note that Example 2 is equivalent to Example 2.27 in [189]. Therefore, by the same theorems, as above, the results of [189] imply exponential decay of the $A^{(k)}$ in this setting[1].

[1]The block $A_{m,l}^{(k)}$ in our notation is $W^{(m)}\pi^{(m,k)}A^{(k)}\pi^{(k,l)}W^{(l),\top}$ in the notation of [189].

See also [190, Theorem 15.45] for a detailed proof and [190, Theorem 15.43] for required sufficient lower bounds on $A^{(k)}_{ii}$. □

4.5 Bounded condition numbers

In this section, we will bound the condition numbers of $B^{(k)}$ based on the following condition, which we will show to be satisfied for Examples 1 and 2.

Condition 2. *Let $H \in (0,1), C_\Phi \geq 1$ be constants such that for $1 \leq k < l \leq q$,*

$$\lambda_{\min}\left(\Theta^{(k)}\right) \geq \frac{1}{C_\Phi} H^{2k}, \tag{4.25}$$

$$\lambda_{\max}\left(\Theta^{(q)}_{l,l} - \Theta^{(q)}_{l,1:k}\Theta^{(q),-1}_{1:k,1:k}\Theta^{(q)}_{1:k,l}\right) \leq C_\Phi H^{2k}\,. \tag{4.26}$$

Theorem 5. *Condition 2 implies that, for all $1 \leq k \leq q$,*

$$C_\Phi^{-1} H^{-2(k-1)} \mathrm{Id} \prec B^{(k)} \prec C_\Phi H^{-2k} \mathrm{Id}, \tag{4.27}$$

and, for $\kappa := H^{-2} C_\Phi^2$,

$$\operatorname{cond}\left(B^{(k)}\right) \leq \kappa\,. \tag{4.28}$$

Proof. The lower bound in (4.27) follows from (4.26) and

$$B^{(k)} = \left(\Theta^{(q)}_{k,k} - \Theta^{(q)}_{k,1:(k-1)}\Theta^{(q),-1}_{1:k,1:k}\Theta^{(q)}_{1:(k-1),k}\right)^{-1}. \tag{4.29}$$

The upper bound in (4.27) follows from (4.25) and $B^{(k)} = \left(\left(\Theta^{(k)}\right)^{-1}\right)_{k,k}$. □

The following theorem shows that (4.26) is a Poincaré inequality closely related to the accuracy of numerical homogenization basis functions [120, 163, 191] and (4.25) is an inverse Sobolev inequality related to the regularity of the discretization of $\mathcal{L}$:

Theorem 6. *Condition 2 holds true if the constants $C_\Phi \geq 1$ and $H \in (0,1)$ satisfy*

(1) $\frac{1}{C_\Phi} H^{2k} \leq \frac{\|\phi\|_*^2}{|\alpha|^2}$, for $\alpha \in \mathbb{R}^{I^{(k)}}$ and $\phi = \sum_{i \in I^{(k)}} \alpha_i \phi_i$; and

(2) $\min_{\varphi \in \operatorname{span}(\phi_i)_{i \in I^{(k-1)}}} \frac{\|\phi - \varphi\|_*^2}{|\alpha|^2} \leq C_\Phi H^{2(k-1)}$, for $\alpha \in \mathbb{R}^{J^{(l)}}$, $k < l \leq q$, and $\phi = \sum_{i \in J^{(l)}} \alpha_i \phi_i$.

Proof. Inequality (4.25) is a direct consequence of the first assumption of the theorem, whereas (4.26) follows from the variational property [258, Theorem 5.1] of the Schur complement:

$$\alpha^\top \left(\Theta_{l,l} - \Theta^{(q)}_{l,1:k}\Theta^{(q),-1}_{1:k,1:k}\Theta^{(q)}_{1:k,l}\right)\alpha = \inf_{\beta\in\mathbb{R}^{I^{(k)}}} (\alpha-\beta)^\top \Theta^{(q)}(\alpha-\beta) \tag{4.30}$$

$$= \min_{\varphi\in\operatorname{span}\{\phi_i \mid i\in I^{(k)}\}} \|\phi-\varphi\|_*^2 \leq C_\Phi H^{2k}|\alpha|^2. \tag{4.31}$$

□

We will now show that Examples 1 and 2 satisfy the conditions of Theorem 6. For simplicity, for $\tilde{\Omega} \subset \Omega$ and $\phi \in H^{-s}(\Omega)$, we still write ϕ for the unique element $\tilde{\phi} \in H^{-s}(\tilde{\Omega})$ such that $[\tilde{\phi}, u] = [\phi, u]$ for $u \in H_0^s(\tilde{\Omega})$. The following Fenchel conjugate identity [40, Ex. 3.27, p. 93] will be useful throughout this section.

$$\|\phi\|^2_{H^{-s}(\Omega)} = \sup_{v\in H_0^s(\Omega)} 2[\phi, v] - \|v\|^2_{v\in H_0^s(\Omega)}. \tag{4.32}$$

The first condition can be verified in a similar way as is done in [189].

Lemma 3. *Let Θ be given as in Examples 1 and 2. Then there exists a constant C depending only on δ, s, and d, such that*

$$\frac{1}{C_\Phi} h^{2sk} \leq \frac{\|\phi\|_*^2}{|\alpha|^2}, \tag{4.33}$$

for $C_\Phi = \|\mathcal{L}\|C$, $\alpha \in \mathbb{R}^{I^{(k)}}$, and $\phi = \sum_i \alpha_i\phi_i$.

Proof. The proof can be found in Section .1. □

In order to verify the second condition in Theorem 6, we will construct a φ such that $\phi - \varphi$ integrates to zero against polynomials of order at most $s-1$ on domains of size h^k. Then an application of the Bramble–Hilbert lemma [65] will yield the desired factor h^{ks}. To avoid scaling issues, we define, for $1 \leq k \leq q$ and $i \in I^{(k)}$,

$$\phi_i^{(k)} := \begin{cases} \boldsymbol{\delta}_{x_i}, & \text{in Example 1,} \\ \mathbf{1}_{\tau_i^{(k)}}/|\tau_i^{(k)}|, & \text{in Example 2,} \end{cases} \tag{4.34}$$

noting that $\operatorname{span}\{\phi_i^{(k)} \mid i \in I^{(k)}\} = \operatorname{span}\{\phi_i \mid i \in I^{(k)}\}$. To obtain estimates independent of the regularity of Ω, for the simplicity of the proof and without loss of generality, we will partially work in the extended space $\mathbb{R}^d$ (rather than on Ω). We write v for the zero extension of $v \in H_0^s(\Omega)$ to $H^s(\mathbb{R}^d)$ and $\phi_i^{(k)}$ for the extension of $\phi_i^{(k)} \in H^{-s}(\Omega)$ to an element of the dual space of $H^s_{\text{loc}}(\mathbb{R}^d)$. We introduce new measurement functions in the complement of Ω as follows. For $1 \leq k \leq q$, we consider countably infinite index sets $\tilde{I}^{(k)} \supset I^{(k)}$. We choose points $(x_i)_{i \in \tilde{I}^{(q)} \setminus I^{(q)}}$ satisfying

$$\sup_{x \in \mathbb{R}^d \setminus \Omega} \min_{i \in \tilde{I}^{(k)}} \operatorname{dist}(x_i, x) \leq \delta^{-1} h^k, \qquad \min_{i \neq j \in \tilde{I}^{(k)} \setminus I^{(k)}} \operatorname{dist}(x_i, x_j \cup \partial\Omega) \geq \delta h^k. \tag{4.35}$$

We then define, for $1 \leq k \leq q$ and $i \in \tilde{I}^{(k)}$, $\phi_i^{(k)} := \delta_{x_i}$ for Example 1, and $\phi_i^{(k)} := \frac{\mathbf{1}_{B_{\delta h^k}(x_i)}}{|B_{\delta h^k}(x_i)|}$ for Example 2. Let $\mathcal{P}^{s-1}$ denote the linear space of polynomials of degree at most $s-1$ (on $\mathbb{R}^d$).

Lemma 4. *Let Θ be as in Example 1 or Example 2. Given $\rho \in (2, \infty)$ and $1 \leq k < l \leq q$, let $w \in \mathbb{R}^{J^{(l)} \times \tilde{I}^{(k)}}$ be such that*

$$\int_{B_{\rho h^k}(x_i)} \left(\phi_i - \sum_{j \in \tilde{I}^{(k)}} w_{ij} \phi_j^{(k)} \right)(x) p(x) \, \mathrm{d}x = 0, \quad \textit{for all } p \in \mathcal{P}^{s-1} \textit{ and } i \in J^{(l)} \tag{4.36}$$

and $w_{ij} \neq 0 \Rightarrow \operatorname{supp}\left(\phi_j^{(k)}\right) \subset B_{\rho h^k}(x_i)$. Then, for $\alpha \in \mathbb{R}^{J^{(l)}}$, $\phi := \sum_{i \in J^{(l)}} \alpha_i \phi_i$ and $\varphi := \sum_{i \in J^{(l)}, j \in I^{(k)}} \alpha_i w_{ij} \phi_j^{(k)}$ satisfy

$$\|\phi - \varphi\|_*^2 \leq \|\mathcal{L}^{-1}\| C(d,s) \frac{\rho^{d+2s}}{\delta^d} \left(1 + h^{-ld} \omega_{l,k}^2\right) h^{2sk} |\alpha|^2, \tag{4.37}$$

with $\omega_{l,k} := \sup_{i \in J^{(l)}} \sum_{j \in \tilde{I}^{(k)}} |w_{ij}|$ and $\|\phi\|_* := \sup_{u \in H_0^s(\Omega)} [\phi, u] / [\mathcal{L}u, u]^{\frac{1}{2}}$ as in (4.7).

We proceed by proving Lemma 4 in the setting of Example 1. The proof in the setting of Example 2 can be found in Section .1. For $u \in H^s(\Omega)$, write $\mathrm{D}^0 u := u$ and for $1 \leq k \leq s$, write $\mathrm{D}^k u$ for the vector of partial derivatives of u of order k, i.e. $\mathrm{D}^k u := \left(\frac{\partial^k u}{\partial_{i_1} \cdots \partial_{i_k}} \right)_{i_1, \ldots, i_k = 1, \ldots, d}$. The proof of Lemma 4 will use the following version of the Bramble–Hilbert lemma:

Lemma 5 ([65]). *Let $\Omega \subset \mathbb{R}^d$ be convex and let ϕ be a sublinear functional on $H^s(\Omega)$ for $s \in \mathbb{N}$ such that*

(1) there exists a constant $\tilde{C}$ such that, for all $u \in H^s(\Omega)$,

$$|\phi(u)| \leq \tilde{C} \sum_{k=0}^{s} \operatorname{diam}(\Omega)^k \|\mathrm{D}^k u\|_{L^2(\Omega)}; \tag{4.38}$$

(2) and $\phi(p) = 0$ for all $p \in \mathcal{P}^{s-1}$.

Then, for all $u \in H^s(\Omega)$,

$$|\phi(u)| \leq \tilde{C} C(d,s)\operatorname{diam}(\Omega)^s \|\mathrm{D}^s u\|_{L^2(\Omega)}. \tag{4.39}$$

The following lemma is obtained from Lemma 5:

Lemma 6. *For $1 \leq k < l \leq q$ and $i \in J^{(l)}$, let ϕ_i, w_{ij} be as in Lemma 4 and Example 2 and define $\varphi_i := \sum_{j \in I^{(k)}} w_{ij}\phi_j^{(k)}$. Then there exists a constant $C(d,s)$ such that, for all $v \in H_0^s(\Omega)$,*

$$\left|\int_{B_{\rho h^k}(x_i)} (\phi_i - \varphi_i)(x)v(x)\,\mathrm{d}x\right| \leq C(d,s)\rho^{s-d/2}h^{(s-d/2)k}\left(h^{ld/2} + \sum_{j \in \tilde{I}^{(k)}} |w_{ij}|\right)\|\mathrm{D}^s v\|_{L^2\left(B_{\rho h^k}(x_i)\right)}. \tag{4.40}$$

Proof. We apply Lemma 5 to the linear functional $u \mapsto \int_{B_{\rho h^k}} (\phi_i - \varphi_i)u$. Since the second requirement of Lemma 5 is fulfilled by definition, it remains to bound $\tilde{C}$. We only execute the proof for Example 1; the proof for Example 2 is analogous. We first note that while the sum in the definition of φ_i only ranges over $j \in I^{(k)}$, we can increase it to run over all of $j \in \tilde{I}^{(k)}$, since for $j \in \tilde{I}^{(k)} \setminus I^{(k)}$, the support of $\phi_j^{(k)}$ is disjoint from that of $v \in H_0^s(\Omega)$. Let $u \in H^s(\Omega)$. Writing $C(d,s)$ for the continuity constant of the embedding of $H^s(B_1(0))$ into $C_b(B_1(0))$, the inequalities

$$\max_{B_{\rho h^k}(x_i)} |u(\cdot)| = \max_{x \in B_1(0)} \left|u\left(\rho h^k(x - x_i)\right)\right| \leq C(d,s)\sum_{m=0}^{s}(\rho h^k)^m \left\|[\mathrm{D}^m u]\left(\rho h^k(\cdot - x_i)\right)\right\|_{L^2(B_1(0))}$$

and

$$\left\|[\mathrm{D}^m u]\left(\rho h^k(\cdot - x_i)\right)\right\|_{L^2(B_1(0))} = (\rho h^k)^{-d/2}\|\mathrm{D}^m u\|_{L^2(B_{\rho h^k}(x_i))}$$

imply that

$$|\phi_i(u) - \varphi_i(u)| \leq \left(h^{ld/2} + \sum_{j \in \tilde{I}^{(k)}} |w_{ij}|\right)\max_{x \in B_{\rho h^k}(x_i)} |u(x)| \tag{4.41}$$

$$\leq C(d,s)\rho^{-d/2}h^{-kd/2}\left(h^{ld/2} + \sum_{j \in \tilde{I}^{(k)}} |w_{ij}|\right)\sum_{m=0}^{s}(\rho h^k)^m\|\mathrm{D}^m u\|_{L^2(B_{\rho h^k}(x_i))}. \tag{4.42}$$

Therefore the first condition of Lemma 5 holds with

$$\tilde{C} = C(d,s)\rho^{-d/2}h^{-kd/2}\left(h^{ld/2} + \sum_{j \in \tilde{I}^{(k)}} |w_{ij}|\right), \tag{4.43}$$

and we conclude the proof by writing $C(d,s)$ for any constant depending only on d and s. □

We can now conclude the proof of Lemma 4.

Proof of Lemma 4. Write $\varphi := \sum_{i\in J^{(l)}} \alpha_i \varphi_i$ and $\varphi_i := \sum_{j \in I^{(k)}} w_{ij}\phi_j^{(k)}$. Equation (4.32) implies that

$$\|\phi-\varphi\|^2_{H^{-s}(\Omega)} = \sup_{v\in H_0^s(\Omega)} \left(\sum_{i\in J^{(l)}} 2\alpha_i \int_{B_{\rho h^k}(x_i)} (\phi_i - \varphi_i)(x) v(x)\,\mathrm{d}x \right) - \|v\|^2_{H_0^s(\Omega)}\,. \tag{4.44}$$

The packing inequality $\sum_{i\in J^{(l)}} \|\mathrm{D}^s v\|^2_{L^2\left(B_{\rho h^k}(x_i)\right)} \leq C(d)\left(h^{k-l}\rho/\delta\right)^d \|v\|^2_{H_0^s(\Omega)}$ together with Lemma 6 yields

$$\|\phi-\varphi\|^2_{H^{-s}(\Omega)} \leq \sup_{v\in H_0^s(\Omega)} \sum_{i\in J^{(l)}} \left[2|\alpha_i| C(d,s)\rho^{s-\frac{d}{2}} h^{(s-\frac{d}{2})k} \left(h^{\frac{ld}{2}} + \sum_{j\in I^{(k)}} |w_{ij}| \right) \|\mathrm{D}^s v\|_{L^2(B_{\rho h^k}(x_i))} \right. \tag{4.45}$$

$$\left. - (C(d))^{-1} \left(h^{k-l}\rho/\delta\right)^{-d} \|\mathrm{D}^s v\|^2_{L^2\left(B_{\rho h^k}(x_i)\right)} \right]. \tag{4.46}$$

Applying the inequality $2ax - bx^2 \leq a^2/b$ to each summand yields

$$\|\phi-\varphi\|^2_{H^{-s}(\Omega)} \leq C(d)\left(h^{k-l}\rho/\delta\right)^d \sum_{i\in J^{(l)}} \left(\alpha_j C(d,s) \rho^{s-\frac{d}{2}} h^{(s-\frac{d}{2})k} \left(h^{\frac{ld}{2}} + \sum_{j\in J^{(k)}} |w_{ij}| \right)\right)^2 \tag{4.47}$$

$$\leq C(d,s)\frac{\rho^{2s}}{\delta^d}\left(1 + h^{-ld}\omega_{l,k}^2\right) h^{2sk} |\alpha|^2\,. \tag{4.48}$$

Since, for all $f \in H^{-s}(\Omega)$,

$$\|f\|_*^2 = [f, \mathcal{L}^{-1} f] \leq \|f\|_{H^{-s}(\Omega)} \|\mathcal{L}^{-1} f\|_{H_0^s(\Omega)} \leq \|\mathcal{L}^{-1}\| \|f\|^2_{H^{-s}(\Omega)}, \tag{4.49}$$

we have $\|\phi-\varphi\|_* \leq \sqrt{\|\mathcal{L}^{-1}\|}\|\phi-\varphi\|_{H^{-s}(\Omega)}$, and this completes the proof. □

The following geometric lemma shows that the assumption (4.36) of Lemma 4 can be satisfied with a uniform bound on the value of ρ and the norm of weights $w_{i,j}$.

Lemma 7. *There exist constants $\rho(d,s)$ and $C(d,s,\delta)$ such that for all $1 \leq k < l \leq q$, there exist weights $w \in \mathbb{R}^{J^{(l)}\times \tilde{I}^{(k)}}$ satisfying (4.36) and (with $\omega_{l,k}$ defined as in Lemma 4)*

$$\omega_{l,k}^2 \leq h^{ld} C(d,s,\delta)\,. \tag{4.50}$$

Proof. For Example 1, (4.36) is equivalent to

$$h^{ld/2} p(x_i) = \sum_{j\in \tilde{I}_\rho^{(k)}} w_{ij} p(x_j), \forall p \in \mathcal{P}^{s-1}, \tag{4.51}$$

where $\tilde{I}^{(k)}_{\rho} := \{j \in \tilde{I}^{(k)} \mid x_j \in B(x_i, \rho h^k)\}$.

Fix $i \in J^{(l)}$, let $\lambda > 0$, and write $x^{\lambda}_j := \frac{x_j - x_i}{\lambda}$. Write $\mathbf{0} := (0, \dots, 0) \in \mathbb{R}^d$. Since the function $p(\cdot) \mapsto p(\frac{\cdot - x_i}{\lambda})$ is surjective on $\mathcal{P}^{s-1}$, (4.51) is satisfied if

$$h^{ld/2} p(\mathbf{0}) = \sum_{j \in \tilde{I}^{(k)}_{\rho}} w_{ij} p(x^{\lambda}_j), \forall p \in \mathcal{P}^{s-1}. \tag{4.52}$$

For a multiindex $n = (n_1, \dots, n_d) \in \mathbb{N}^d$ and a point $z = (z_1, \dots, z_d) \in \mathbb{R}^d$, write $z^n := \prod_{m=1}^{d} z_m^{n_m}$. Use the convention $\mathbf{0}^n = 0$ if $n \neq \mathbf{0}$ and $\mathbf{0}^{\mathbf{0}} = 1$. To satisfy (4.52), it is sufficient to identify a subset σ of $\tilde{I}^{(k)}_{\rho}$ and $w_{i,\cdot} \in \mathbb{R}^{\tilde{I}^{(k)}}$ such that $\#\sigma = s^d$, $w_{i,j} = 0$ for $j \notin \sigma$, and

$$h^{ld/2} \mathbf{0}^n = \sum_{j \in \sigma} w_{ij} (x^{\lambda}_j)^n, \forall n \in \{0, \dots, s-1\}^d. \tag{4.53}$$

Let $\mathbb{V}^{\lambda} \in \mathbb{R}^{\{0,1,\dots,s-1\}^d \times \sigma}$ be the $s^d \times s^d$ matrix defined by

$$\mathbb{V}^{\lambda}_{n,j} := \left(x^{\lambda}_j\right)^n. \tag{4.54}$$

For a multiindex $n \in \mathbb{N}^d$ and a point $x \in \mathbb{R}^d$, $x^n := \prod_{m=1}^{d} x^{n_m}$. Let $\mathbf{w} \in \mathbb{R}^{\sigma}$ be defined by $\mathbf{w}_j := w_{i,j}$ for $j \in \sigma$. Equation (4.53) is then equivalent to

$$h^{ld/2} \mathbf{e} = \mathbb{V}^{\lambda} \mathbf{w}, \tag{4.55}$$

where $\mathbf{e} \in \mathbb{R}^{\{0,1,\dots,s-1\}^d}$ is defined by $\mathbf{e}_n := \mathbf{0}^n$ for $n \in \{0, 1, \dots, s-1\}^d$. We will now identify $\mathbf{w}$ by inverting (4.55). To achieve this while keeping the norm of $\mathbf{w}$ under control, we will seek to identify the subset σ and $\lambda > 0$ such that $\sigma_{\min}(\mathbb{V}^{\lambda})$ (the minimal singular value of $\mathbb{V}^{\lambda}$) is bounded from below by a constant depending only on s and d.

For $\alpha \geq 0$, let $(\epsilon_j)_{j \in \{0,1,\dots,s-1\}^d}$ be elements of $\mathbb{R}^d$ satisfying $|\epsilon_j| \leq \alpha$ for all $j \in \{0, 1, \dots, s-1\}^d$. Let $\mathbf{1} := (1, \dots, 1) \in \mathbb{R}^d$ and, for $j \in \{0, 1, \dots, s-1\}^d$, let $z_j := \mathbf{1} + j + \epsilon_j$. Observe that for $\alpha = 0$, the points z_j are on a regular grid. Let $\bar{\mathbb{V}}^{\alpha} \in \mathbb{R}^{\{0,1,\dots,s-1\}^d \times \{0,1,\dots,s-1\}^d}$ be the $s^d \times s^d$ matrix defined by $\bar{\mathbb{V}}^{\alpha}_{n,j} := (z_j)^n$. Let V be the $s \times s$ Vandermonde matrix defined by $V_{i,j} = i^j$. Writing $\sigma_{\min}(V)$ for the minimal singular value of V, we have for $\alpha = 0$, by [119, Theorem 4.2.12],

$$\sigma_{\min}\left(\bar{\mathbb{V}}^0\right) = \left(\sigma_{\min}(V)\right)^d. \tag{4.56}$$

Since univariate polynomial interpolation on s points with polynomials of degree $s-1$ is uniquely solvable, we have $\sigma_{\min}(V) > 0$ and $\sigma_{\min}(\bar{\mathbb{V}}^0) > C(d, s) > 0$.

Therefore, the continuity of the minimal singular value with respect to the entries of $\bar{\mathbb{V}}^{\alpha}$ implies that there exists $\alpha^*, \sigma^* > 0$ depending only on s, d such that $\alpha \leq \alpha^*$ implies $\sigma_{\min}(\bar{\mathbb{V}}^{\alpha}) > \sigma^*$. Since (by construction) the $(x_i)_{i \in \tilde{I}^{(k)}}$ form a covering of $\mathbb{R}^d$ of radius h^k, the $(x_i^{\lambda})_{i \in \tilde{I}^{(k)}}$ form a covering of $\mathbb{R}^d$ of radius h^k/λ and for each $n \in \{0, 1, \ldots, s-1\}^d$, there exists an $x_{j_n}^{\lambda}$ that is at distance at most h^k/λ from n. Let $\sigma := \{j_n \mid n \in \{0, 1, \ldots, s-1\}^d\} \subset \tilde{I}^{(k)}$ be the collection of corresponding labels. It follows from $|x_{j_n}^{\lambda}| \leq \sqrt{d}s + h^k/\lambda$ that $|x_{j_n} - x_i| \leq \lambda\sqrt{d}s + h^k$, and $\sigma \subset \tilde{I}_{\rho}^{(k)}$ for $\rho > 1 + \lambda\sqrt{d}s/h^k$. Selecting $\lambda = h^k/\alpha^*$ implies that $\sigma_{\min}(\mathbb{V}^{\lambda}) > \sigma^*$ and $\sigma \subset \tilde{I}_{\rho}^{(k)}$ for $\rho > 1 + \sqrt{d}s/\alpha^*$. Defining

$$w_{ij} := \begin{cases} \left((\mathbb{V}^{\lambda})^{-1} h^{ld/2} \mathbf{e} \right)_n, & \text{if } j = j_n \in \sigma, \\ 0, & \text{otherwise,} \end{cases} \tag{4.57}$$

the weights w_{ij} satisfy $\omega_{kl} \leq C(s, d) h^{ld/2}$ and (4.36) with a ρ depending only on s and d. This concludes the proof for Example 1. The proof is similar for Example 2 with minor changes (the bound on ω also depends on δ). □

The following lemma concerns the satisfaction of the second condition of Theorem 6:

Lemma 8. *In the setting of Examples 1 and 2, there exists some constant $C(d, s, \delta) > 0$ such that, for $2 \leq k < l \leq q$, $\alpha \in \mathbb{R}^{J^{(l)}}$ and $\phi = \sum_i \alpha_i \phi_i$,*

$$\min_{\varphi \in \operatorname{span}(\phi_i)_{i \in I^{(k-1)}}} \frac{\|\phi - \varphi\|_*^2}{|\alpha|^2} \leq C(d, s, \delta) \|\mathcal{L}^{-1}\| h^{2s(k-1)} . \tag{4.58}$$

Proof. Apply Lemma 4 with the bounds on ρ and ω obtained in Lemma 7. □

The following theorem is a direct consequence of Theorems 6, Lemma 3 and Lemma 8.

Theorem 7. *In the setting of Examples 1 and 2, there exists a constant $C(d, s, \delta)$ such that Condition 2 is fulfilled with $C_{\Phi} := \max(\|\mathcal{L}\|, \|\mathcal{L}^{-1}\|) C(d, s, \delta)$ and $H := h^s$.*

Propagation of exponential decay

We will now derive the exponential decay of the Cholesky factors L by combining the algebraic identities of Lemma 1 with the bounds on the condition numbers of the $B^{(k)}$ (implied by Condition 2) and the exponential decay of the $A^{(k)}$ (specified in Condition 1). The core of our proof is based on a combination/extension of the results of [31, 32, 34, 68, 128, 144] on decay algebras. The pseudodistance $d(\cdot,\cdot)$ appearing in (4.13) is not a pseudometric because it does not satisfy the triangle inequality. However, to prove (4.13), we will only need the following weaker version of the triangle inequality:

Definition 8. *A function $d\colon I \times I \longrightarrow \mathbb{R}_+$ is called a* hierarchical pseudometric *if*

(1) $d(i,i) = 0$, for all $i \in I$;

(2) $d(i,j) = d(j,i)$, for all $i, j \in I$;

(3) for all $1 \leq k \leq q$, $d(\cdot,\cdot)$ restricted to $J^{(k)} \times J^{(k)}$ is a pseudometric;

(4) for all $1 \leq k \leq l \leq m \leq q$ and $i \in J^{(k)}, s \in J^{(l)}, j \in J^{(m)}$, we have $d(i,j) \leq d(i,s) + d(s,j)$.

Note that the $d(\cdot,\cdot)$ specified in (4.20) for Examples 1 and 2 is a hierarchical pseudometric. For a hierarchical pseudometric $d(\cdot,\cdot)$ and $\gamma \in \mathbb{R}_+$, let

$$c_d(\gamma) := \sup_{1\leq k\leq l\leq q} \sup_{j\in J^{(l)}} \sum_{i\in J^{(k)}} \exp(-\gamma d(i,j)). \tag{4.59}$$

The following theorem states the main result of this section:

Theorem 8 (Exponential decay of the Cholesky factors)**.** *Assume that Θ fulfills Conditions 1 and 2 with the constants $\gamma, C_\gamma, H, C_\Phi$ and the hierarchical pseudometric $d(\cdot,\cdot)$. Then*

$$\left|(\operatorname{chol}(\Theta))_{ij}\right| \leq \frac{2C_\Phi c_d\left(\tilde{\gamma}/8\right)^2}{(1-r)^2}\left(4c_d\left(\tilde{\gamma}/4\right)\frac{C_\Phi C_\gamma\left(c_d\left(\tilde{\gamma}/2\right)\right)^2}{(1-r)^2}\right)^q \exp\left(-\frac{\tilde{\gamma}}{8}d(i,j)\right), \tag{4.60}$$

where $C_R := \max\left\{1, \frac{2C_\gamma C_\Phi}{1+\kappa}\right\}$, $r := \frac{1-\kappa^{-1}}{1+\kappa^{-1}}$, $\tilde{\gamma} := \frac{-\log(r)}{1+\log(c_d(\gamma/2))+\log(C_R)-\log(r)}\frac{\gamma}{2}$, and $\kappa = H^{-2}C_\Phi^2$ is defined as in Theorem 5.

The remaining part of this section will present the proof of Theorem 8. We will use the following lemma on the stability of exponential decay under matrix multiplication, the proof of which is a minor modification of that of [128].

Lemma 9. *Let I be an index set that is partitioned as $I = J^{(1)} \cup \cdots J^{(q)}$ and let $d\colon I \times I \to \mathbb{R}_{\geq 0}$ satisfy*

$$d(i_1, i_{n+1}) \leq \sum_{k=1}^{n} d(i_k, i_{k+1}) \quad \textit{for all } 1 \leq n \leq q-1 \textit{ and } i_k \in J^{(k)}.$$

Let $M^{(k)} \in \mathbb{R}^{J^{(k)} \times J^{(k+1)}}$ be such that $|M^{(k)}_{i,j}| \leq C \exp(-\gamma d(i,j))$ for $1 \leq k \leq q-1$ and let

$$c_d(\gamma/2) := \sup_{1 \leq k \leq q-1} \sup_{j \in J^{(k+1)}} \sum_{i \in J^{(k)}} \exp\left(-\frac{\gamma}{2} d(i,j)\right) \textit{ for } \gamma \in \mathbb{R}_+. \tag{4.61}$$

Then, for $1 \leq n \leq q-1$,

$$\left| \left(\prod_{k=1}^{n} M^{(k)} \right)_{i,j} \right| \leq \left(c_d\left(\gamma/2\right) C\right)^n \exp\left(-\frac{\gamma}{2} d(i,j)\right).$$

Proof. Set $i_1 := i$, $i_{n+1} := j$. Then

$$\begin{aligned}
\left| \left(\prod_{k=1}^{n} M^{(k)} \right)_{i,j} \right| &\leq C^n \sum_{i_2, \ldots, i_n \in J^{(2)}, \ldots, J^{(n)}} \exp\left(-\gamma \sum_{k=1}^{n} d\left(i_k, i_{k+1}\right)\right) \\
&\leq C^n \exp\left(-\frac{\gamma}{2} d\left(i_1, i_{n+1}\right)\right) \sum_{i_2, \ldots i_n \in I} \exp\left(-\frac{\gamma}{2} \sum_{k=1}^{n} d\left(i_k, i_{k+1}\right)\right) \\
&\leq \left(c_d\left(\gamma/2\right) C\right)^n \exp\left(-\frac{\gamma}{2} d(i,j)\right).
\end{aligned}$$

□

The proof of the following lemma (on the stability of exponential decay under matrix inversion for well-conditioned matrices) is nearly identical to that of [128] (we only keep track of constants; see also [68] for a related result on the inverse of sparse matrices).

Lemma 10. *Let $A \in \mathbb{R}^{I \times I}$ be symmetric and positive definite such that for $C, \gamma > 0$ and a metric $d(\cdot,\cdot)$ on I we have $|A_{i,j}| \leq C \exp(-\gamma d(i,j))$. It holds true that*

$$\left|(A^{-1})_{i,j}\right| \leq \frac{4}{\left(\|A\| + \|A^{-1}\|^{-1}\right)(1-r)^2} \exp\left(-\frac{\log(\frac{1}{r})}{\left(1 + \log\left(c_d\left(\gamma/2\right)\right) + \log(C_R)\right) + \log(\frac{1}{r})} \frac{\gamma}{2} d(i,j)\right) \tag{4.62}$$

where $c_d(\gamma/2) := \sup_{j \in I} \sum_{i \in I} \exp\left(-\frac{\gamma}{2} d(i,j)\right)$, $C_R := \max\left\{1, \frac{2C}{\|A\| + \|A^{-1}\|^{-1}}\right\} = \max\left\{1, \frac{2C\|A^{-1}\|}{1+\kappa}\right\}$, $r := \frac{1 - \frac{1}{\|A\|\|A^{-1}\|}}{1 + \frac{1}{\|A\|\|A^{-1}\|}} = \frac{1-\kappa^{-1}}{1+\kappa^{-1}}$, and $\kappa := \|A\|\|A^{-1}\|$ is the condition number of A.

Proof. On a compact set not containing 0, the function $x \mapsto x^{-1}$ can be accurately approximated by low-order polynomials in x. Then, the spread of the exponential decay can be controlled by Lemma 9. See Section .1 for details. □

By representing Schur complements as matrix inverses, Lemma 10 can also be used to show that the Cholesky factors of well-conditioned exponentially-decaying matrices are exponentially decaying. The following lemma appears in a similar form in [34] for banded matrices and in [144] without explicit constants.

Lemma 11. *Let $B \in \mathbb{R}^{I \times I} \simeq \mathbb{R}^{N \times N}$ be symmetric and positive definite with condition number κ and such that $\left|B_{i,j}\right| \leq C \exp(-\gamma d(i,j))$ for some constant $C > 0$ and some metric d on I. Let L be the Cholesky factor (in an arbitrary order) of B^{-1} ($B^{-1} = LL^T$). Then*

$$\left|L_{i,j}\right| \leq \frac{4\sqrt{\|B\|}}{\left(\|B\| + \|B^{-1}\|^{-1}\right)(1-r)^2} \exp\left(\frac{\log(r)}{1 + \log\left(c_d\left(\gamma/2\right)\right) + \log(C_R) - \log(r)} \frac{\gamma}{2} d(i,j)\right) \tag{4.63}$$

where $c_d(\gamma/2) := \sup_{j \in I} \sum_{i \in I} \exp\left(-\frac{\gamma}{2} d(i,j)\right)$, $C_R := \max\left\{1, \frac{2C\|B^{-1}\|}{1+\kappa}\right\}$, and $r := \frac{1-\kappa^{-1}}{1+\kappa^{-1}}$.

Proof. Lemma 2 implies that the Schur complements of B^{-1} can be expressed as inverses of sub-matrices of B. The result then follows from Lemma 10 (see Proof.1 for details). □

The last ingredient needed to prove the exponential decay of the Cholesky factors of Θ is the following lemma showing the stability of exponential decay under inversion for block-lower-triangular matrices (this operation appears in the definition of $\bar{L}$ in (4.14)):

Lemma 12. *Let I be an index set that is partitioned as $I = J^{(1)} \cup \cdots J^{(q)}$ and assume that the matrix $L \in \mathbb{R}^{I \times I}$ is block-lower triangular with respect to this partition, with identity matrices as diagonal blocks. If $d(\cdot,\cdot)$ is a hierarchical pseudometric such that $|L_{ij}| \leq C \exp\left(-\gamma d(i,j)\right)$ (for some $C \geq 1$ and $\gamma > 0$), then it holds true that*

$$\left|(L^{-1})_{ij}\right| \leq 2^q \left(c_d\left(\gamma/2\right) C\right)^q \exp\left(-\frac{\gamma}{2} d(i,j)\right) \tag{4.64}$$

with $c_d(\gamma) := \sup_{1 \leq k \leq l \leq q} \sup_{j \in J^{(l)}} \sum_{i \in J^{(k)}} \exp\left(-\gamma d(i,j)\right)$.

Proof. The Neumann series of a $q \times q$ block-lower-triangular matrix with identity matrices on the (block) diagonal can be written as

$$L^{-1} = \sum_{k=0}^{q} \left(\mathrm{Id} - L\right)^k \ . \tag{4.65}$$

Since the sum terminates in q steps, the thickening of the exponential decay can be bounded using Lemma 9. See Proof .1 for details. $\square$

By applying the above results to the decomposition obtained in Lemma 1, we conclude the proof of Theorem 8. See Proof .1 for details.

4.6 Summary of results

The results of the previous sections allow us to prove the following theorem on the exponential decay of the Cholesky factors and the accuracy of their truncation:

Theorem 9. *In the setting of Examples 1 and 2, there exist constants $C, \gamma, \alpha > 0$ depending only on d, Ω, s, $\|\mathcal{L}\|$, $\|\mathcal{L}^{-1}\|$, h, and δ, such that the entries of the Cholesky factor L of Θ satisfy*

$$|L_{ij}| \leq CN^{\alpha} \exp(-\gamma d(i,j)), \tag{4.66}$$

where $\operatorname{dist}\colon I \times I \longrightarrow \mathbb{R}$ *is defined by*

$$\operatorname{dist}(i,j) := h^{-\min(k,l)} \operatorname{dist}\left(\operatorname{supp}(\phi_i), \operatorname{supp}(\phi_j)\right) \quad \textit{for all } i \in J^{(k)},\ j \in J^{(l)}. \tag{4.67}$$

As a consequence, writing

$$L^S_{ij} := \begin{cases} L_{ij}, & \textit{for } (i,j) \in S \\ 0, & \textit{else,} \end{cases} \tag{4.68}$$

with $S \supset S_{\operatorname{dist},\rho} := \{(i,j) \mid \operatorname{dist}(i,j) \leq \rho\}$, we have $\left\|\Theta - L^S L^{S,\top}\right\|_{\mathrm{Fro}} \leq \epsilon$ for $\rho \geq \tilde{C}(C,\gamma)\log(N/\epsilon)$.

Proof. Theorems 4 and 7 imply that Conditions 1 and 2 are fulfilled with constants depending only on d, s, $\|\mathcal{L}\|$, $\|\mathcal{L}^{-1}\|$, h, and δ. Theorem 8 concludes the exponential decay of L. The accuracy of the truncated factors follows directly from the exponential decay. We note that dist is not a hierarchical pseudometric, but there exists a constant $c_{\min}, c > 0$ only depending on h and δ such that for $\operatorname{dist}(i,j) \geq c_{\min}$, we have $c^{-1} d(i,j) \leq \operatorname{dist}(i,j) \leq c d(i,j)$ where $d(\cdot,\cdot)$ is the hierarchical pseudometric specified in (4.20). $\square$

We next show the exponential decay of the Cholesky factor of $A := A^{-1}$. We first observe the following consequence of Lemma 1:

Lemma 13. *When ordering the multiresolution basis from fine to coarse (q to 1), the Cholesky factors of A are given by the Block matrix*

$$L = \begin{pmatrix} L^{(q),-1} & \dots & \dots & 0 \\ A^{(q)}_{q-1,q} L^{(q),\top} & \ddots & 0 & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ A^{(q)}_{1,q} L^{(q),\top} & \dots & A^{(2)}_{1,2} L^{(2),\top} & L^{(1),-1} \end{pmatrix}, \tag{4.69}$$

where $L^{(k)}$ is the lower triangular Cholesky factor of $B^{(k),-1}$.

Proof. Lemma 1 implies $\Theta^{-1} = \bar{L}^{\top,-1} D^{-1} \bar{L}^{-1}$. By computing the Cholesky factorization of D and multiplying the resulting factors with $\bar{L}^{\top,-1}$ and $\bar{L}^{-1}$, we obtain a factorization $\Theta^{-1} = UU^\top$ with U block upper triangular with lower-triangular diagonal blocks. Letting P denote the permutation matrix that inverts the order of the blocks while keeping the order within each block the same, we obtain

$$P\Theta^{-1}P^\top = PUU^\top P^\top = PUP^\top PU^\top P^\top = \left(PUP^\top\right)\left(PUP^\top\right)^\top. \tag{4.70}$$

We can now verify that $PUP^\top$ is lower triangular and equal to L, which concludes the proof due to the uniqueness of the Cholesky factors. □

Theorem 10. *In the setting of Examples 1 and 2, there exist constants $C, \gamma, \alpha > 0$ depending only on d, Ω, s, $\|\mathcal{L}\|$, $\|\mathcal{L}^{-1}\|$, h, and δ, such that the entries of the Cholesky factor L of Θ satisfy*

$$|L_{ij}| \leq CN^\alpha \exp(-\gamma \operatorname{dist}(i,j)), \tag{4.71}$$

where $\operatorname{dist}\colon I \times I \longrightarrow \mathbb{R}$ *is defined by*

$$\operatorname{dist}(i,j) \coloneqq h^{-\max(k,l)} \operatorname{dist}\left(\operatorname{supp}(\phi_i), \operatorname{supp}(\phi_j)\right) \quad \textit{for all } i \in J^{(k)},\ j \in J^{(l)}. \tag{4.72}$$

As a consequence, writing

$$L^S_{ij} \coloneqq \begin{cases} L_{ij}, & \textit{for } (i,j) \in S \\ 0, & \textit{else,} \end{cases} \tag{4.73}$$

with $S \supset S_{d,\rho} \coloneqq \{(i,j) \mid d(i,j) \leq \rho\}$, we have $\left\|A - L^S L^{S,\top}\right\|_{\mathrm{Fro}} \leq \epsilon$ for $\rho \geq \tilde{C}(C,\gamma)\log(N/\epsilon)$.

Proof. The proof is similar to that of Theorem 9, but we will treat each block of L in isolation. There exist $c_{\min}, c$ depending only on h and δ such that for each $1 \leq k \leq q$ and $i, j \in J^{(k)}$ with $\operatorname{dist}(i, j) \geq c_{\min}$, we have $c^{-1} d\left(i, j\right) \leq \operatorname{dist}(i, j) \leq c d\left(i, j\right)$ for d defined as in (4.20). [2] Therefore, we can use Lemmas 10 and 11 to prove the exponential decay of $L^{(k)}$ according to dist. According to Theorem 4, this means that for any k, l and $i \in J^{(k)}, j, \tilde{j} \in J^{(l)}$, we have

$$\log\left(\left|\left(A^{(l)}\right)_{i\tilde{j}}\left(L^{(l),\top}\right)_{\tilde{j}j}\right|\right) \tag{4.74}$$

$$\lessapprox c + c_2 \log(N) - \frac{\left(\operatorname{dist}\left(\operatorname{supp}\left(\phi_i\right), \operatorname{supp}\left(\phi_{\tilde{j}}\right)\right) - \operatorname{dist}\left(\operatorname{supp}\left(\phi_{\tilde{j}}\right), \operatorname{supp}\left(\phi_j\right)\right)\right)}{h^l} \tag{4.75}$$

$$\lessapprox \tilde{c}_1 + \tilde{c}_2 \log(N) - \frac{\left(\operatorname{dist}\left(\operatorname{supp}\left(\phi_i\right), \operatorname{supp}\left(\phi_j\right)\right)\right)}{h^l}. \tag{4.76}$$

Here, the second inequality follows the fact that the size of $\operatorname{supp}\left(\phi_{\tilde{j}}\right)$ is approximately h^l and thus for $\operatorname{dist}(i, j)$ larger than a constant, it can lead to a violation of the triangle inequality by at most a constant factor. □

Theorems 9 and 10 now allow us to prove a rigorous version of Theorem 1:

Theorem 11 (Rigorous version of Theorem 1)**.** *Let $\Omega \in \mathbb{R}^d$ be a Lipschitz-bounded domain and let $\mathcal{L} : H_0^s\left(\Omega\right) \longrightarrow H^{-s}\left(\Omega\right)$ be a linear, symmetric, bounded, and local operator of order $2s > d$. Let $\{x_i\}_{1 \leq i \leq N} \subset \Omega$ and ordered in the maximin [reverse maximin] ordering such that δ as in (4.6) is positive. For $\mathcal{G} = \mathcal{L}^{-1]}$ the Dirichlet Green's function define $\Theta_{ij} := \mathcal{G}\left(x_i, x_j\right)$ and let L be the Cholesky factor of Θ [$A := \Theta^{-1}$]. Let $S \subset \{1, \ldots, N\}^2$ be the maximin [reverse maximin] sparsity pattern with starting set $\partial\Omega$ and sparsity parameter ρ. Defining $\left(L^S\right)_{ij} := \mathbb{1}_{(i,j) \in S} L_{ij}$, we then have*

$$\log\left(\left\|L^S L^{S,\top} - L L^\top\right\|_{\mathrm{Fro}}\right) \lessapprox \log(N) - \rho \tag{4.77}$$

where the constants depend only on d, Ω, s, $\|\mathcal{L}\|$, $\|\mathcal{L}^{-1}\|$, and δ.

Proof. We perform the proof for the Cholesky factorization of Θ in the maximin ordering. The Cholesky factor of A can be treated in an analog manner. As described in 3.6.1, the maximin ordering can be represented as a hierarchical ordering

[2]We note that this is *not* true for i and j on different scales!

satisfying the conditions of Example 1. The result follows from Theorem 9 by observing that the maximin sparsity pattern S_ρ satisfies

$$S_{d,(\delta h)^{-1}\rho} \supset S_\rho \supset S_{d,\delta h \rho}\,. \tag{4.78}$$

Scaling the weights of the measurement functions ϕ_i to 1 increases the error by a factor that is at most polynomial in N, which can be subsumed into the $\log(N)$-dependence of ρ by increasing the constants in the decay estimates. □

Similarly, Theorem 7 also implies a rigorous version of Theorem 2:

Theorem 12 (Rigorous version of Theorem 2)**.** *In the setting of Theorem 1, let $L_{:,1:k}$ be the rank k matrix defined by the first k columns of the Cholesky factor L of Θ in the maximin ordering. Denoting as $\|\cdot\|$ the operator norm, we have*

$$\left\|\Theta - L_{(:,1:k)}\left(L_{:,1:k}\right)^\top\right\| \leq C\|\Theta\|k^{-2s/d} \tag{4.79}$$

with constant C depending only on $\|\mathcal{L}\|$, $\|\mathcal{L}^{-1}\|$, d, δ, and s.

Proof. We begin by proving, in the setting of Example 1,

$$\left\|\Theta - L^{(k)}L^{(k),\top}\right\| \leq C l_{i_{k+1}}^{2s-d}. \tag{4.80}$$

Write $I = I_1 \cup I_2$ with $I_1 := \{i_1, \ldots, i_k\}$ and $I_2 := I \setminus I_1$ and ℓ_k, ℓ_{k+1} for the lenght-scales of the $k, k+1$-th points in the ordering, respectively. By Lemma 2, the approximation error made by keeping only the first k columns of the Cholesky factorization is equal to the Schur complement $\Theta_{2,2} - \Theta_{2,1}\Theta_{1,1}^{-1}\Theta_{1,2}$. Consider the implicit hierarchy of the maximin ordering as described in Section 3.6.1 with $h = 1/2$, and let $p \in \{1, \ldots, q\}$ be such that $2^{-p} \leq l[k]/l[1] \leq 2^{-p+1}$. Write $I = I_a \cup I_b$ with $I_a := I^{(p)}$ and $I_b := I \setminus I^{(p)}$. The variational property (4.30) implies that $\Theta_{2,2} - \Theta_{2,1}\Theta_{1,1}^{-1}\Theta_{1,2} \leq \Theta_{b,b} - \Theta_{b,a}\Theta_{a,a}^{-1}\Theta_{a,b}$. Theorem 7 (with $h = 1/2$ obtained from the implicit hierarchy) implies that $\Theta_{b,b} - \Theta_{b,a}\Theta_{a,a}^{-1}\Theta_{a,b} \leq C(\frac{1}{2})^{2s(p-1)-d}$, where the extra multiplicative $(\frac{1}{2})^{-d}$ term arises because the measurement functions are scaled by $h^{kd/2}$ in Example 1 with $h = \frac{1}{2}$. We conclude the proof of (4.80) using $2^{-p-1} \leq \ell[k+1]/\ell[1] \leq 2^{-p+1}$. We can now prove the result by rescaling the basis function and using a ball-packing argument to show that $\ell_k \approx k^{-1/d}$. □

Finally, Theorem 3 follows the discussion in Section 3.6.2 that implies that the resulting low-rank approximation of Θ is identical to that in Theorem 2.

Theorem 13 (Rigorous version of Theorem 3)**.** *In the setting of Theorem 11, let $L_{:,1:k}$ be the rank k matrix defined by the first k columns of the Cholesky factor L of $A := \Theta$ in the reverse maximin ordering. We then have*

$$\left\|\Theta - \Psi(L_{1:k,1:k}L_{1:k,1:k}^{\top})^{-1}\Psi^{\top}\right\| \leq C\|\Theta\|k^{-2s/d}, \quad \textit{for } \Psi = \begin{pmatrix} \mathrm{Id} \\ -L_{(k+1):N,1:k}L_{1:k,1:k}^{-1} \end{pmatrix} \tag{4.81}$$

with constant C depending only on $\|\mathcal{L}\|$, $\|\mathcal{L}^{-1}\|$, d, δ, and s.

4.7 Extensions and comparisons

We conclude this chapter by comparing our results in more detail to closely related approaches and by mentioning some empirical observations that are not covered by the theory presented in this chapter.

4.7.1 $\mathcal{H}$-matrix approximations from sparse Cholesky factorization

The $\mathcal{H}$-matrix data structure [112] uses low-rank approximations for blocks $\Theta_{\bar{I}\bar{J}}$ ($\bar{I}, \bar{J} \subset I$) fulfilling the admissibility condition

$$\min\bigl(\mathrm{diam}\{x_i\}_{i\in\bar{I}}, \mathrm{diam}\{x_i\}_{i\in\bar{J}}\bigr) \leq \eta \,\mathrm{dist}\bigl(\{x_i\}_{i\in\bar{I}}, \{x_i\}_{i\in\bar{J}}\bigr). \tag{4.82}$$

The approximation property of the incomplete Cholesky factorization in maximin ordering (Theorem 1) directly implies bounds on the spectral decay of admissible blocks in the $\mathcal{H}$-matrix framework, as can be seen from the representation

$$\Theta = LL^{\top} \iff \Theta = \sum_{i=1}^{N} L_{:i} \otimes L_{:i} \tag{4.83}$$

of the Cholesky factorization of Θ. If L is sparse according to the maximin sparsity pattern, then $L_{:i} \otimes L_{:i}$ can contribute to the rank of the sub-matrix $\Theta_{\bar{I}\bar{J}}$ only if

$$2\rho l[i] \geq \mathrm{dist}\left(\{x_j\}_{j\in\bar{I}}, \{x_j\}_{j\in\bar{J}}\right) \text{ and } \max\left(\mathrm{dist}\left(x_i, \{x_j\}_{j\in\bar{I}}\right), \mathrm{dist}\left(x_i, \{x_j\}_{j\in\bar{J}}\right)\right) \leq \rho\ell[i]. \tag{4.84}$$

The number of $i \in I$ satisfying (4.84) is at most $C(\eta, d)\rho^d \log N$, which recovers (up to constants) the same rank bounds as obtained in [29] for second-order elliptic PDEs with rough coefficients. However the converse is not true and most hierarchical matrix representations can not be written in terms of a sparse Cholesky factorization of Θ. For example, adding a diagonal matrix to Θ does not affect the ranks of admissible blocks, but it diminishes the screening effect and thus the approximation property of the incomplete Cholesky, as shown in Section 5.5.3.

4.7.2 Comparison to Cholesky factorization in wavelet bases

[94] compute sparse Cholesky factorizations of (discretized) differential/integral operators represented in a wavelet basis. Using a *fine-to-coarse* elimination ordering, they establish that the resulting Cholesky factors decay polynomially with an exponent matching the number of vanishing moments of the underlying wavelet basis.

For differential operators, this coincides algorithmically with the Cholesky factorization described by Theorem 10 and the gamblet transform of [188] and [189], whose estimates guarantee exponential decay. In particular [94] numerically observe a uniform bound on $\operatorname{cond}(B^{(k)})$ which they relate to the approximate sparsity of their proposed Cholesky factorization.

For integral operators, [94] use a *fine-to-coarse* ordering and we use a *coarse-to-fine* ordering. While their results rely on the approximate sparsity of the integral operator represented in the wavelet basis, our approximation remains accurate for multiresolution bases (e.g. the maximin ordering), in which Θ is dense, which avoids the $O(N^2)$ complexity of a basis transform (or the implementation of adaptive quadrature rules to mitigate this cost).

4.7.3 Vanishing moments

Let $\mathcal{P}^{s-1}(\tau)$ denote the set of polynomials of order at most $s-1$ that are supported on $\tau \subset \Omega$. [188] and [189] show that (4.26) and (4.25) hold when $\mathcal{L}$ is an elliptic partial differential operator of order s (as described in Section 4.2.1) and the measurements are local polynomials of order up to $s-1$ (i.e. $\phi_{i,\alpha} = 1_{\tau_i} p_\alpha$ with $p_\alpha \in \mathcal{P}^{s-1}(\tau_i)$). Using these $\phi_{i,\alpha}$ as measurements is equivalent to using wavelets ϕ_i satisfying the vanishing moment condition

$$[\phi_i, p] = 0 \quad \text{for all } i \in I,\, p \in \mathcal{P}^{s-1}. \tag{4.85}$$

The requirement for vanishing moments has three important consequences. First, it requires that the order of the operator be known a priori, so that a suitable number of vanishing moments can be ensured. Second, ensuring a suitable number of vanishing moments greatly increases the complexity of the implementation. Third, in order to provide vanishing moments, the measurements ϕ_i, $i \in J^{(k)}$, have to be obtained from weighted averages over domains of size of order h^k. Therefore, even computing the first entry of the matrix Θ in the multiresolution basis will have complexity $O(N^2)$, since it requires taking an average over almost all of $I \times I$. One of the main analytical result of this paper is to show that these vanishing moment conditions and local averages are not necessary for higher order operators (which, in particular, enables the generalization of the gamblet transform to hierarchies of measurements defined as in Examples 1 and 2).

4.7.4 The cases $s \leq d/2$ or $s \notin \mathbb{N}$

Theorems 1 requires that $s > d/2$ to ensure that the elements of $H^s(\Omega)$ are continuous (by the Sobolev embedding theorem) and that pointwise evaluations of the Green's function are well defined. The accuracy estimate of Theorem 1 can be extended to $s \leq d/2$ by replacing pointwise evaluations of the Green's function by local averages and using variants of the Haar pre-wavelets of as in Example 2 instead of the subsampled Diracs of Example 1 to form Θ. Numerical experiments suggest that the exponential decay of Cholesky factors still holds for $s \leq d/2$ if the local averages of Example 2 are sub-sampled as in Example 1, whereas the low-rank approximation becomes sub-optimal. Our theory further requires s to be an integer, excluding fractional order elliptic PDEs. However, our experiments suggest that this is not strictly necessary, either. As illustrated in Table 5.5, for Matérn kernels we observe no difference (in accuracy vs. complexity) between integer and non-integer values of s. It is an open question how to reconcile this observation with the fact that fractional order partial differential operators are in general nonlocal.

Chapter 5

INCOMPLETE CHOLESKY FACTORIZATION

5.1 Zero fill-in incomplete Cholesky factorization

5.1.1 `ICHOL(0)`

In Chapter 4, we have shown that discretized Green's matrices of elliptic PDEs, as well as their inverses, have exponentially decaying Cholesky factors. By setting small entries to zero, we can approximate these Cholesky factors with exponential accuracy by sparse matrices with near-linearly many nonzero entries. However, computing the exact Cholesky factor in the first place requires access to all $O\left(N^2\right)$ nonzero entries and has computational cost $O\left(N^3\right)$, and is therefore not practical.

A simple approach to decreasing the computational complexity of Cholesky factorization is the *zero fill-in incomplete Cholesky factorization* [170] (`ICHOL(0)`). When performing Gaussian elimination using `ICHOL(0)`, we treat all entries of both the input matrix and the output factors outside a prescribed *sparsity pattern* $S \subset I \times I$ as zero and correspondingly ignore all operations in which they are involved.

5.1.2 Complexity vs accuracy

It is well known that the computational complexity of `ICHOL(0)`can be bounded in terms of the maximal number elements of S either per row or per column:

Theorem 14. *If m upper bounds the number of elements of (the lower triangular part of) S in either each row or each column, then Algorithm 3 has time complexity $O(Nm^2)$.*

Proof. If m upper bounds the number of elements per column, we see that the number of updates performed for a given i is upper bounded by the number of pairs (k, j) for which (k, i) and (j, i) are contained in S. This number is upper bounded by m^2, leading to a complexity $O\left(Nm^2\right)$. If m upper bounds the number of elements per row, then the number of updates is upper bounded by the number of pairs (i, j) for which both (k, i) and (k, j) are part of the sparsity pattern. This number is bounded by m^2, leading to a complexity $O\left(Nm^2\right)$. □

We begin by providing a bound on the computational complexity of `ICHOL(0)`in coarse-to-fine ordering.

Algorithm 3 Incomplete Cholesky factorization with sparsity pattern S.

Input: $A \in \mathbb{R}^{N \times N}$ symmetric, $\mathrm{nz}(A) \subset S$
Output: $L \in \mathbb{R}^{N \times N}$ lower triang. $\mathrm{nz}(L) \subset S$

```
 1: for $(i,j) \notin S$ do
 2:     $A_{ij} \leftarrow 0$
 3: end for
 4: for $i \in \{1, \dots, N\}$ do
 5:     $L_{:i} \leftarrow A_{:i} / \sqrt{A_{ii}}$
 6:     for $j \in \{i+1, \dots, N\} : (i,j) \in S$ do
 7:         for $k \in \{j, \dots, N\} : (k,i),(k,j) \in S$ do
 8:             $A_{kj} \leftarrow A_{kj} - \frac{A_{ki} A_{ji}}{A_{ii}}$
 9:         end for
10:     end for
11: end for
12: return $L$
```

Figure 5.1: **`ICHOL(0)`.** Incomplete Cholesky factorization, with the differences to ordinary Cholesky factorization (Algorithm 1), highlighted in red. Here, for a matrix A, $\mathrm{nz}(A) \coloneqq \{(i,j) \mid A_{ij} \neq 0\}$ denotes the index set of its non-zero entries.

Theorem 15. *In the setting of Examples 1 and 2, there exists a constant $C(d,\delta)$, such that, for*

$$S \subset \{(i,j) \mid i \in J^{(k)}, j \in J^{(l)} h^{-\min(k,l)} \operatorname{dist}\left(\operatorname{supp}(\phi_i), \operatorname{supp}(\phi_j)\right) \leq \rho\}, \qquad (5.1)$$

the application of Algorithm 3 in a coarse-to-fine ordering has computational complexity $C(d,\delta)Nq\rho^d$ in space and $C(d,\delta)Nq^2\rho^{2d}$ in time. In particular, $q \propto \log N / \ln \frac{1}{h^d}$ implies the upper bounds of $C(d,\delta,h)\rho^d N \log N$ on the space complexity, and of $C(d,\delta,h)\rho^{2d} N \log^2 N$ on the time complexity. In particular, the above complexities apply to `ICHOL(0)`in the maximin ordering and sparsity pattern.

Proof. For $i \in J^{(k)}, j \in J^{(l)}$, write $d(i,j) = h^{-\min(k,l)} \operatorname{dist}\left(\operatorname{supp}(\phi_i), \operatorname{supp}(\phi_j)\right)$. We write $i \prec j$ if i precedes j in the coarse-to-fine ordering.

Defining $m \coloneqq \max_{j \in I, 1 \leq k \leq q} \#\{i \in J^{(k)} \mid i \prec j \text{ and } d(i,j) \leq \rho\}$, $|x_i - x_j| \geq \delta^{-1} h^l$ for $i,j \in I^{(l)}$ implies that $m \leq C(d,\delta)\rho^d$. Therefore $\#\{i \in I \mid i \prec j \text{ and } d(i,j) \leq \rho\} \leq qmN$ implies the bound on space complexity. The bound on the time complexity follows from Theorem 14. □

When using the reverse ordering and sparsity pattern, the computational complexity is even lower.

Theorem 16. *In the setting of Example 1, there exists a constant $C(d,\delta)$, such that, for*

$$S \subset \{(i,j) \mid i \in J^{(k)}, j \in J^{(l)} h^{-\max(k,l)} \operatorname{dist}\left(\operatorname{supp}(\phi_i), \operatorname{supp}(\phi_j)\right) \leq \rho\}, \qquad (5.2)$$

the application of Algorithm 3 in a fine-to-coarse ordering has computational complexity $C(d,\delta)N\rho^d$ in space and $C(d,\delta)N\rho^{2d}$ in time. In particular, the above complexities apply to `ICHOL(0)` *in the reverse maximin ordering and sparsity pattern. In the setting of Example 2, the complexities are $C(d,\delta)N\max\left(\log(N), \rho^d\right)$ in space and $C(d,\delta)N\max\left(\log^2(N), \rho^{2d}\right)$ in time.*

Proof. For $i \in J^{(k)}, j \in J^{(l)}$, write $d(i,j) = h^{-\max(k,l)} \operatorname{dist}\left(\operatorname{supp}(\phi_i), \operatorname{supp}(\phi_j)\right)$. We write $i \prec j$ if i precedes j in the fine-to-coarse ordering.

In the setting of Example 1, for $i \in J^{(k)}$, the i-th column has $\lessapprox \rho^d$ nonzero entries, since it contains only points within radius ρh^k, while all points succeeding i in the reverse maximin ordering have a distance of at least δh^k from each other. In the setting of Example 2, the i-th column also has at least a constant number of elements in each of the subsequent levels, which is why even for constant ρ, the number of elements per column and thus the complexity of `ICHOL(0)` will still scale logarithmically with N. $\square$

We can now make a first attempt at combining the estimates on computational complexity and accuracy.

Theorem 17. *In the setting of Theorems 9 and 10, there exists constants C, c depending only on d, Ω, s, $\|\mathcal{L}\|$, $\|\mathcal{L}^{-1}\|$, h, and δ such that for every $\rho > c\log(N)$, there exists a perturbation $E \in \mathbb{R}^{I \times I}$ with $\log(\|E\|_{\mathrm{Fro}}) \leq -C\rho$ such that the result applying Algorithm 3 in the coarse-to-fine [fine-to-coarse] ordering with sparsity set S satisfying*

$$\{(i,j) \mid \operatorname{dist}(i,j) \leq \rho/2\} \subset S \subset \{(i,j) \mid \operatorname{dist}(i,j) \leq 2\rho\} \qquad (5.3)$$

returns a Cholesky factor L^S satisfying

$$\log\left(\left\| LL^\top - L^S L^{S,\top} \right\|\right) \leq -C\rho. \qquad (5.4)$$

Here, L is the exact Cholesky factor. Analogue results hold in the setting of Theorem 11, when using the maximin [reverse maximin] ordering and sparsity pattern.

Proof. Writing $E := LL^{\top} - L^{S}L^{S,\top}$ the results of Theorems 9 and 10 show that $\log(\|E\|_{\mathrm{Fro}}) \leq -C\rho$. The Cholesky factor of $LL^{\top} - E$ is equal to L^{S} and thus recovered exactly by Algorithm 3, when applied to the E-perturbed input matrix. The results on the computational complexity follow from Theorems 15 and 16. □

If we were able to show that the incomplete Cholesky factorization is stable in the sense that an exponentially small perturbation of the input matrix to `ICHOL(0)` leads to a perturbation of its output that is amplified by a factor at most polynomial in the size of the matrix, we could use Theorem 17 to obtain a rigorous result on the complexity vs accuracy tradeoff. Unfortunately, despite overwhelming evidence that this is true for the problems considered in this paper, we are not able to prove this result in general. We also did not such a result in the literature, even though works such as [94] would require it to provide rigorous complexity vs accuracy bounds.

We will however show that it holds true for a slightly modified ordering and sparsity pattern that is obtained when introducing "supernodes" and multicolor orderings. Since these techniques are also useful from a computational standpoint, we will first introduce them as part of a description detailing the efficient implementation of our methods, together with numerical results. We will then address and resolve the question of stability again in Section 5.4.3.

5.1.3 Efficient implementation of `ICHOL(0)`

Theorems 15 and 16 show that in the limit of N going to infinity, using Algorithm 3 and a sparsity pattern satisfying (5.3) is vastly cheaper that computing a dense Cholesky factorization. However, if we were to naïvely implement Algorithm 3 using a standard compressed sparse row representation of S, we would see that the number of floating-point operations per second that our algorithm can perform much lower compared to good implementations of dense Cholesky factorization. This has three main reasons: **irregular memory access**, **lack of memory re-use**, and **parallelism**.

Irregular memory access. While the random access memory (RAM) of present-day computers allows to access memory in arbitrary order, the latency of random memory access is often substantial. In order to mitigate this effect, present-day computers have a hierarchy of *caches*, smaller units of memory that can be accessed much more rapidly. As it is loading data from RAM, the computer constantly tries to predict what data could be requested next and preemptively loads it into the cache. So-called *cache misses*, instances where data requested from the CPU is not preloaded in the cache but instead has to be loaded from RAM can lead to dramatical regression in performance. Arguably the most fundamental heuristic used to predict which data will be required next is known as *spatial locality*, meaning that whenever we request data from RAM, the data immediately before and after will be pre-fetched into the cache, as well. Thus, accessing data in linear order is one of the most reliable ways to avoid cache misses. Without further thought, Algorithm 3 has poor spatial locality. For instance, if we implement using a compressed sparse row (CSR) format to store (the lower triangular part of) A, each iteration of the inner loop over k will access a different row of the matrix that could be located in a part of memory far away from where the previous row was stored. Therefore, *every single update* could access memory in a different location that therefore is unlikely to be pre-fetched, resulting in a large number of cache misses.

Our main remedy to this problem will be to change the order of the three for loops in Algorithm 3 resulting in the *left-looking* and *up-looking* variants of Cholesky factorization, as opposed to the *right-looking* variant presented in Algorithm 3. Choosing the appropriate combination of direction (up, left, or right) and storage format (compressed sparse rows or compressed sparse columns) can greatly decrease the running time of our algorithms.

Lack of memory re-use. Even if we ignore questions of cache-misses and memory latency, there is still the issue of memory bandwidth. In Algorithm 3, for each loading of A_{kj}, A_{ki}, and A_{kj} only a minuscule amount of computation is performed. A_{kj} particular will only be used for a single computation, and it can take arbitrarily long until its next appearance in the computation. This poor *temporal locality* of Algorithm 3 means that even with perfect prefetching into the cache, the computation might be limited by the bandwidth with which memory can be loaded into the cache, rather than the clock speed of the CPU. In contrast, the canonical example for good memory reused are blocked algorithms for matrix-matrix multiplication, the product between two blocks of size $k \times k$ that are small enough to fit into cache only needs to load only $\approx k^2$ floating-point numbers into the cache to perform $\approx k^3$ floating-point operations.

We will mitigate this problem by introducing *supernodes* [158, 209], consisting of successive rows and columns that share the same sparsity pattern. Due to the flexibility for reordering rows and columns within each level of the multiresolution scheme and the geometry of the associated measurements, we can identify supernodes of size $\approx \rho^d$ by only marginally increasing the size of the sparsity pattern. Once the supernodes have been identified, Algorithm 3 can be reexpressed as a smaller number of *dense* linear algebra operations that feature substantially more data re-use.

Parallelism. Modern CPUs have multiple *cores* that have access to the same memory, but can perform tasks independently. Optimized implementations of dense Cholesky factorization are able to exploit this so-called *shared memory parallelism* by identifying subproblems that can be solved in parallel and assigning them to the different cores of the CPU. For large computations, *distributed parallelism* is necessary, where subproblems are solved by different computers that are connected to form a so-called *cluster*. The amount of parallelism available on modern computational platforms is increasing steadily. Thus, for an algorithm to be practically viable, it is crucial that it allows for the efficient use of parallel computation.

We will exploit parallelism by observing that rows and columns with nonoverlapping sparsity patterns can be eliminated in parallel. Using once again the fact that we have complete freedom to choose the ordering within each level, we can use a multi-color ordering [3, 4, 74, 192] inspired by the classical *red-black ordering* [127] in order to maximize the number of operations that can be performed in parallel.

5.2 Implementation of `ICHOL(0)` for dense kernel matrices

5.2.1 Direction and storage

We propose to compute the factorization in-place, using a compressed sparse row format to store the factor. This means that the matrix $L \in \mathbb{R}^{N \times N}$ with sparsity pattern S is stored as a tuple $(\texttt{rowptr}(S), \texttt{colval}(S), \texttt{nzval}(S, L))$ where $\texttt{rowptr}(S) \in \mathbb{N}^{N+1}$, $\texttt{colval}(S) \in \mathbb{N}^{\#(S)}$, and $\texttt{nzval}(S, L) \in \mathbb{R}^{\#(S)}$ such that for each $1 \leq k \leq N$, the nonzero entries of the k-th row of L lie in the columns with indices

$$(\texttt{colval}[j] \mid \texttt{rowptr}[k] \leq j < \texttt{rowptr}[k+1]) \tag{5.5}$$

and are given as

$$(\texttt{nzval}[j] \mid \texttt{rowptr}[k] \leq j < \texttt{rowptr}[k+1]) \,. \tag{5.6}$$

In Figure 5.2, we describe the algorithm for computing the up-looking Cholesky factorization in the CSR format. We observe that this variant shows significantly improved performance in practice. A possible approach to further improve its performance would be to reorder the degrees of freedom on each level in a way that increases locality, for instance ordered along a Z-curve.

5.2.2 Supernodes

We will now discuss a supernodal implementation appropriate for Cholesky factorization of Θ in a coarse-to-fine ordering. In order to improve readability, we do not provide rigorous proofs in this section and refer to the more technical discussion in Section 5.4.3.

The main idea of our implementation of supernodes is to aggregate degrees of freedom with length-scale $\approx \ell$ that are within a ball of radius $\rho\ell$ and appear consecutively in the elimination ordering into a so-called *supernode*.

Definition 9. *Let $\{x_i\}_{1 \leq i \leq N} \in \mathbb{R}^d$. A supernodal aggregation of scale ℓ is given by a set $\left\{\tilde{x}_{\tilde{i}}\right\}_{1 \leq \tilde{i} \leq \tilde{N}} \subset \mathbb{R}^d$ and assignment of each point x_i to exactly one supernode $\tilde{x}_{\tilde{i}}$. We denote this assignment $i \rightsquigarrow \tilde{i}$ and require that for $i \rightsquigarrow \tilde{i}$, we have* $\operatorname{dist}(x_i, \tilde{x}_{\tilde{i}}) \leq \ell$.

Given $\{x_i\}_{1 \leq i \leq N} \in \mathbb{R}^d$, such an ordering can be constructed efficiently using a greedy algorithm.

Construction 1 (Supernodal aggregation)**.** *We successively select the first x_i that is not yet within range of one of the existing supernodes and make it a new supernode until all x_i are within a distance of ℓ of at least one supernode. Then, we assign each x_i to the closest supernode, using an arbitrary mechanism to break ties.*

Construction 1 can be implemented efficiently using, for instance, KD-tree based range search. Under mild regularity conditions such as the one given in Equation (4.6), the number of resulting supernodes is bounded as N/ℓ^d.

In a multi-scale setting, we will use separate aggregation scales for each level:

Definition 10. *For $1 \leq k \leq q$, let $\left\{x_i^{(k)}\right\}_{1 \leq i \leq N^{(k)}} \in \mathbb{R}^d$. A multiscale supernodal aggregation with scales $\left\{\ell^{(k)}\right\}_{1 \leq k \leq q}$. Is given by the union of the supernodal aggregation for each $\left\{x_i^{(k)}\right\}_{1 \leq i \leq N^{(k)}}$ with the respective length scale $\ell^{(k)}$.*

Once a multiscale supernodal aggregation is available, we can use it to derive associated orderings and sparsity patterns.

Algorithm 4 In-place, up-looking `ICHOL(0)`in CSR format

Input: $L \in \mathbb{R}^{N \times N}$ lower triangular, as CSR with `rowptr`, `colval`, `nzval`

```
for $i = 1 : N$ do
    for $\tilde{j}$ = rowptr[$i$] : (rowptr[$i+1$] $-$ 1) do
        $j \leftarrow$ colval[$\tilde{j}$]
        nzval[$\tilde{j}$] $\leftarrow$ nzval[$\tilde{j}$] $-$ dot($i, j$, rowptr, colval, nzval)
        if $j < i$ then
            nzval[$\tilde{j}$] $\leftarrow$ nzval[$\tilde{j}$] / nzval[rowptr[$j+1$] $-$ 1]
        else
            nzval[$\tilde{j}$] $\leftarrow \sqrt{\texttt{nzval}[\tilde{j}]}$
        end if
    end for
end for
```

Algorithm 5 Alg. `dot` computes update as inner product of i-th and j-th row of L.

Input: $L \in \mathbb{R}^{N \times N}$ lower triangular, as CSR with `rowptr`, `colval`, `nzval`

```
$\tilde{i} \leftarrow$ rowptr[$i$]; $\tilde{j} \leftarrow$ rowptr[$j$]
$\hat{i} \leftarrow$ colval[$\tilde{i}$]; $\hat{j} \leftarrow$ colval[$\tilde{j}$]
out $\leftarrow 0$
while $\hat{i}, \hat{j} < \min(i, j)$ do
    if $\hat{i} == \hat{j}$ then
        out $\leftarrow$ out + nzval[$\tilde{i}$] $*$ nzval[$\tilde{j}$]
        $\tilde{i} \leftarrow \tilde{i} + 1$; $\tilde{j} \leftarrow \tilde{j} + 1$
        $\hat{i} \leftarrow$ colval[$\tilde{i}$]; $\hat{j} \leftarrow$ colval[$\tilde{j}$]
    else if $\hat{i} > \hat{j}$ then
        $\tilde{j} \leftarrow \tilde{j} + 1$
        $\hat{j} \leftarrow$ colval[$\tilde{j}$]
    else
        $\tilde{i} \leftarrow \tilde{i} + 1$
        $\hat{i} \leftarrow$ colval[$\tilde{i}$]
    end if
end while
return out
```

Figure 5.2: **Up-looking `ICHOL(0)`in CSR format.** We present an algorithm that computes a Cholesky factorization of a lower triangular matrix in CSR format, in-place. The up-looking factorization greatly improves the spatial locality.

Definition 11. *Given a multiscale supernodal aggregation, a coarse-to-fine [fine-to-coarse] multiscale supernodal ordering is any ordering of the underlying points in which points assigned to the same supernode appear consecutively the supernodes are ordered by scale, coarse-to-fine [fine-to-coarse]. We define the row-supernodal sparsity pattern S^{-}, column-supernodal sparsity pattern $S^{|}$, and two-way-supernodal sparsity pattern S^{+} as*

$$S^{-} := \left\{(\tilde{i}, j) \mid \exists i \rightsquigarrow \tilde{i} : (i,j) \in S\right\} \subset \left\{1, \dots, \tilde{N}\right\} \times \{1, \dots, N\} \tag{5.7}$$

$$S^{|} := \left\{(i, \tilde{j}) \mid \exists j \rightsquigarrow \tilde{j} : (i,j) \in S\right\} \subset \{1, \dots, N\} \times \left\{1, \dots, \tilde{N}\right\} \tag{5.8}$$

$$S^{+} := \left\{(\tilde{i}, \tilde{j}) \mid \exists i \rightsquigarrow \tilde{i}, j \rightsquigarrow \tilde{j} : (i,j) \in S\right\} \subset \left\{1, \dots, \tilde{N}\right\} \times \left\{1, \dots, \tilde{N}\right\}. \tag{5.9}$$

An entry $(\tilde{i}, j)$ or $(j, \tilde{i})$ existing in S^{-} or $S^{|}$ is to be interpreted adding all interactions between elements of $\tilde{i}$ and j to the original sparsity pattern. Similarly, an entry $(\tilde{i}, \tilde{j})$ existing in S^{+} signifies that all interactions between elements of $\tilde{i}$ and $\tilde{j}$ have been added to the sparsity pattern. This means that we increase the sparsity pattern S in order to ensure that in each supernode all rows, all columns, or both share the same sparsity pattern.

For a coarse-to-fine sparsity pattern S as described in Theorem 17 and using a multiscale supernodal aggregation with $\ell^{(k)} = \rho h^{k}$, the triangle inequality implies that the associated S^{+} (interpreted, by abuse of notation, as subset of I^2) satisfies

$$S \subset S^{+} \subset \{(i,j) \mid \operatorname{dist}(i,j) \leq 4(\rho + 1)\}. \tag{5.10}$$

Thus, passing to S^{+} preserves the asymptotic results on accuracy and complexity.

The benefit of using the supernodal ordering and sparsity pattern is that instead of performing $O\left(N \log^2(N) \rho^{2d}\right)$ element-wise operations on a sparse matrix with $O\left(N \log(N) \rho^{d}\right)$ nonzero entries, we can perform $O\left(N \log^2(N) \rho^{-d}\right)$ block-wise operations on a block-sparse matrix with $O\left(N \log^2(N) \rho^{-d}\right)$ nonzero blocks. Since the size of the blocks is approximately $\rho^d \times \rho^d$, the resulting asymptotic complexity is the same. However, the block-wise operations can be performed using optimized libraries for dense linear algebra. In particular, they require only $\approx \rho^{2d}$ memory accesses for every $\approx \rho^{3d}$ floating point operations.

We can readily adapt Algorithm 4 to this setting by letting `nzval` have matrices as entries. To maximize spatial locality, we store these matrices consecutively in column-major format, in a buffer of length $\sum_{m \in \texttt{nzval}} \#\operatorname{rows}(m) \cdot \#\operatorname{cols}(m)$. Using `cholesky` to denote the dense Cholesky factorization returning a lower triangular matrix, the resulting algorithm is presented in Figure 5.3.

Algorithm 6 In-place, up-looking, supernodal `ICHOL(0)`in CSR format

Input: $\tilde{N} \times \tilde{N}$ block-l. triang. Matrix L, as block-CSR with `rowptr`, `colval`, `nzval`

```
1:  for $i = 1 : \tilde{N}$ do
2:      for $\tilde{j}$ = rowptr[$i$] : (rowptr[$i+1$] − 1) do
3:          $j \leftarrow$ colval[$\tilde{j}$]
4:          nzval[$\tilde{j}$] $\leftarrow$ nzval[$\tilde{j}$] − dot($i, j$, rowptr, colval, nzval)
5:          if $j < i$ then
6:              nzval[$\tilde{j}$] $\leftarrow$ nzval[$\tilde{j}$] / nzval[rowptr[$j+1$] − 1]
7:          else
8:              nzval[$\tilde{j}$] $\leftarrow$ cholesky(nzval[$\tilde{j}$])
9:          end if
10:     end for
11: end for
```

Algorithm 7 Alg. `dot` computes update as inner product of i-th and j-th row of L.

Input: Supernodal indices i and j and block-CSR with `rowptr`, `colval`, `nzval`

```
1:  $\tilde{i} \leftarrow$ rowptr[$i$]
2:  $\tilde{j} \leftarrow$ rowptr[$j$]
3:  $\hat{i} \leftarrow$ colval[$\tilde{i}$]
4:  $\hat{j} \leftarrow$ colval[$\tilde{j}$]
5:  out $\leftarrow 0$
6:  while $\hat{i}, \hat{j} < \min(i, j)$ do
7:      if $\hat{i} == \hat{j}$ then
8:          out $\leftarrow$ out + nzval[$\tilde{i}$]$^\top *$ nzval[$\tilde{j}$]
9:          $\tilde{i} \leftarrow \tilde{i} + 1$
10:         $\hat{i} \leftarrow$ colval[$\tilde{i}$]
11:         $\tilde{j} \leftarrow \tilde{j} + 1$
12:         $\hat{j} \leftarrow$ colval[$\tilde{j}$]
13:     else if $\hat{i} > \hat{j}$ then
14:         $\tilde{j} \leftarrow \tilde{j} + 1$
15:         $\hat{j} \leftarrow$ colval[$\tilde{j}$]
16:     else
17:         $\tilde{i} \leftarrow \tilde{i} + 1$
18:         $\hat{i} \leftarrow$ colval[$\tilde{i}$]
19:     end if
20: end while
21: return out
```

Figure 5.3: **Up-looking supernodal `ICHOL(0)`in CSR format.** We only need to replace the square root with Cholesky factorization in Algorithm 4 and add a transpose in Algorithm 5 to obtain a supernodal factorization.

5.2.3 Multicolor ordering

We will now address the parallel implementation of Algorithms 4 and 6, focusing on shared memory parallelism. The key observation is that depending on the sparsity pattern, many iterations of the outermost for-loop of Algorithms 4 and 6 can be performed in arbitrary order, and in particular in parallel, without changing the result.

Definition 12. *A set of indices $J \subset \{1, \dots, N\}$ is pairwise independent under the sparsity pattern S, if $\{(i, j) \mid i, j \in J\} \cap S = \emptyset$.*

If a consecutive set of indices is mutually independent, the corresponding iterations of the outermost for-loops of Algorithms 4 and 6 can be performed in parallel.

The amount of parallelism afforded by this mechanism depends strongly on the order of the degrees of freedom. This has motivated the development of *multicolor orderings* [3, 4, 74, 192] that attempt to maximize the size of consecutive mutually independent sets of indices. A downside of these techniques, which were designed for the preconditioning of linear system, is the orderings that lead to the largest amount of parallelism often do not achieve a good preconditioning effect. This limitation has led [50] to develop altogether different, asynchronous, and iterative approaches.

A key advantage of the method presented is this chapter is that its exponential accuracy holds for an *arbitrary* ordering of the degrees of freedom on each scale. This allows us to greatly increase the amount of parallelism by adopting a multiscale, multicolor ordering of the supernodes.

Construction 2 (Multiscale multicolor ordering)**.** *Within each level, we initialize each (supernodal) index uncolored. We then greedily select a maximal set of uncolored and pairwise independent, with respect to S ($S^{\dagger}$) indices and assign a new color to them. We proceed until every index is colored and order the indices within each level by color.*

In the setting of Theorem 17, the number of required colors is $O\left(\log(N)\rho^{d}\right)$ when coloring individual entries and $O\left(\log(N)\right)$ when coloring supernodes, leading to practically unlimited amounts of parallelism for large problems.

5.3 Implementation of `ICHOL(0)` for sparse stiffness matrices

5.3.1 Limitations of two-way supernodes

We now shift our attention to the case of Theorem 17 where we want to compute the sparse Cholesky factorization of the exponentially decaying inverse $A := \Theta^{-1}$ using a fine-to-coarse ordering. These matrices appear in the rigorous approach for computing with sums of Gaussian processes outlined in Section ssec:noise and serve as a model for the behavior of stiffness matrices arising from Galerkin discretization of the differential operator $\mathcal{L}$. Since we typically have explicit access to the sparse input matrix A in these applications, we will almost always use the incomplete Cholesky factorization as a preconditioner in conjunction with conjugate gradient (CG) [218]. By increasing the value ρ, we can improve the convergence speed at the cost of an increasing cost of factorization and of applying the preconditioner.

For smaller values of ρ, we can get decent results with a fairly simple implementation using Algorithm 4 combined with the multicolor ordering as described in Construction 2. On an Nvidia GPU with sufficient amounts of RAM, we can even use the parallelized implementation of Algorithm 4, as well as sparse variants of Algorithm 2 provided as part of the cuSPARSE library [179, Algorithm 2].

For larger ρ, supernodal approaches necessary to obtain state-of-the-art performance. Unfortunately, applying two-way supernodes to the fine-to-coarse sparsity pattern of Theorem 17 degrades the asymptotic complexity of the algorithm. In this setting and in the limit of large N, the nonzero entries introduced by S^{+} cannot be accounted for by *any* finite increase of ρ. Thus, two-way supernodes increase the computational complexity by a factor $\log^2(N)$, making them a suboptimal choice for large problems.

5.3.2 Column supernodes

In order to avoid the degradation of asymptotic complexity while still reaping the benefits from dense matrix operations, we are going to use "one-way" column supernodes with sparsity pattern $S^{|}$. This is, in fact, what is usually understood by the term supernodal Cholesky factorization [64].

We will store our matrix L as a tuple $\left(\texttt{colptr}(S^{|}), \texttt{rowval}(S^{|}), \texttt{nzval}(S^{|}, L)\right)$ where `colptr` is a vector of $\tilde{N} + 1$ integers, `rowval` is a vector of $\# S^{|}$ integers, and `nzval` is a vector of $\tilde{N}$ matrices. If the j-th supernode has n degrees of freedom associated to it and the j-th column of $S^{|}$ has m nonzero rows where we have $\texttt{nzval}[k] \in \mathbb{R}^{m \times n}$. $\texttt{rowval}[\texttt{colptr}[j] : (\texttt{colptr}[j+1] - 1)]$ returns the list of nonzero rows of the j-th supernodal column and `nzval`$[j]$ stores all of its nonzero values, in a row-major ordering. Note that if each supernodal column only contains a single column, this reduces to the compressed sparse column format. We will now present the popular left-looking column-supernodal Cholesky factorization. Here, for any supernodal index j, `members`(j) returns the vector of degrees of freedom that are assigned to j. For a vector I of degrees of freedom and a supernode j, `rows`(I) returns the indices of the rows in `nzval`$[j]$ that correspond to members of I. Finally, for a supernodal index j, `parents`(j) returns the list of supernodes ordered before j (excluding j) that have at least one nonzero row i, for which $i \rightsquigarrow j$. The left-looking column-supernodal factorization is presented in Algorithm 8. While this algorithm requires some indexing into the supernodes, the majority of the work is done by calling dense linear algebra routines in Lines 5 and 9.

Algorithm 8 Left-looking, column-supernodal `ICHOL(0)`

Input: Column-supernodal block-lower triangular Matrix L with $\tilde{N}$ supernodes

```
1:  for $j = 1 : \tilde{N}$ do
2:      for $\tilde{j} \in$ parents[j] do
3:          $I \leftarrow \texttt{rowval}[\texttt{colptr}[j] : (\texttt{colptr}[j+1]-1)]$
4:          $\tilde{I} \leftarrow \texttt{rowval}[\texttt{colptr}[\tilde{j}] : (\texttt{colptr}[\tilde{j}+1]-1)]$
5:          in $\leftarrow \texttt{nzval}[\tilde{j}][\texttt{rows}(\tilde{j}, I \cap \tilde{I}), :]$
6:          out $\leftarrow$ in $* (\texttt{nzval}[\tilde{j}][\texttt{rows}(\tilde{j}, \texttt{members}(j) \cap \tilde{I})])^{\top}$
7:          $\texttt{nzval}[j][\texttt{rows}(j, I \cap \tilde{I}), \texttt{rows}(\tilde{j}, \texttt{members}(j) \cap \tilde{I})] \leftarrow \texttt{nzval}[j] -$ out
8:      end for
9:      $\texttt{nzval}[j] \leftarrow \texttt{nzval}[j] / \texttt{cholesky}(\texttt{nzval}[j][\texttt{members}(j), :])^{\top}$
10: end for
```

5.3.3 Parallelism with column-supernodes

Just like in the case of the up-looking Cholesky factorization, any set of consecutive supernodes that is pairwise independent under the sparsity set $S^{\dagger}$ can be treated in parallel in the outermost for-loop in Algorithm 8. Just as before, the number of required colors is $O\left(\log(N)\rho^{d}\right)$ when coloring individual entries and $O\left(\log(N)\right)$ when coloring supernodes, leading to practically unlimited amounts of parallelism for large problems.

5.4 Proof of stability of `ICHOL(0)`

5.4.1 Overview

Theorem 17 implies that the application of Algorithm 3 to a suitable $O(\epsilon)$-perturbation $\Theta - E$ returns an $O(\epsilon)$-accurate Cholesky factorization of Θ in computational complexity $O(N\log^2(N)\log^{2d}(N/\epsilon))$. In practice, we do not have access to E, so we need to rely on the stability of Algorithm 3 to deduce that Θ and $\Theta - E$ (used as inputs) would yield similar outputs for sufficiently small E. Even though such a stability property of `ICHOL(0)` would also be required by prior works on incomplete LU-factorization such as [94], we did not find this type of result in the literature. We also found it surprisingly difficult to prove (and were unable to do so) when using the maximin ordering and sparsity pattern, although we always observed stability of Algorithm 3 in practice, for reasonable values of ρ.

The key problem is that the standard perturbation bounds for Schur complements are multiplicative. Therefore, applying them N times (after each elimination) results in a possible growth of the approximation error that is exponential in N and cannot be compensated for by a logarithmic increase in ρ.

However, we have already seen in Section 5.2 that when using a supernodal multicolor ordering, the incomplete Cholesky factorization can be expressed in a smaller number of groups of independent dense linear algebra operations. In this section, we are going to prove rigorously that the number of colors used by the multicolor ordering is upper bounded as $O(\log(N))$ and that this allows us to control the approximation of the supernodal factorization by only invoking $O(\log(N))$ Schur complement perturbation bounds. Therefore, the error amplification is polynomial in N and can be controlled by choosing $\rho \gtrapprox \log(N)$. By relating ordinary and supernodal Cholesky factorization, we are able to deduce the same error bounds for the ordinary Cholesky factorization when using a supernodal multicolor ordering and sparsity pattern.

5.4.2 Revisiting the supernodal multicolor ordering

We begin by reintroducing the supernodal multicolor ordering of Section 5.2 in slightly different notation.

For $r > 0$, $1 \leq k \leq q$ and $i \in J^{(k)}$, write

$$B_r^{(k)}(i) \coloneqq \{j \in J^{(k)} \mid d(i,j) \leq r\}. \tag{5.11}$$

Construction 3 (Supernodal multicolor ordering and sparsity pattern). *Let* $\Theta \in \mathbb{R}^{I \times I}$ *with* $I \coloneqq \bigcup_{1 \leq k \leq q} J^{(k)}$ *and let* $d(\cdot,\cdot)$ *be a hierarchical pseudometric. For* $\rho \geq 1$, *define the* supernodal multicolor ordering $\prec_\rho$ *and* sparsity pattern S_ρ *as follows. For* *each $k \in \{1, \ldots, q\}$, select a subset $\tilde{J}^{(k)} \subset J^{(k)}$ of indices such that*

$$\forall \tilde{i}, \tilde{j} \in \tilde{J}^{(k)}, \qquad \tilde{i} \neq \tilde{j} \implies B^{(k)}_{\rho/2}\left(\tilde{i}\right) \cap B^{(k)}_{\rho/2}\left(\tilde{j}\right) = \emptyset, \tag{5.12}$$

$$\forall i \in J^{(k)}, \qquad \exists \tilde{i} \in \tilde{J}^{(k)} : i \in B^{(k)}_{\rho}\left(\tilde{i}\right). \tag{5.13}$$

Assign every index in $J^{(k)}$ to the element of $\tilde{J}^{(k)}$ closest to it, using an arbitrary method to break ties. That is, writing $j \rightsquigarrow \tilde{j}$ for the assignment of j to $\tilde{j}$,

$$\tilde{j} \in \operatorname*{arg\,min}_{\tilde{j}' \in \tilde{J}^{(k)}} d\left(j, \tilde{j}'\right), \tag{5.14}$$

for all $j \in J^{(k)}$ and $\tilde{j} \in \tilde{J}^{(k)}$ such that $j \rightsquigarrow \tilde{j}$. Define $\tilde{I} \coloneqq \bigcup_{1 \leq k \leq q} \tilde{J}^{(k)}$ and define the auxiliary sparsity pattern $\tilde{S}_\rho \subset \tilde{I} \times \tilde{I}$ by

$$\tilde{S}_\rho \coloneqq \left\{\left(\tilde{i}, \tilde{j}\right) \in \tilde{I} \times \tilde{J} \,\middle|\, \exists i \rightsquigarrow \tilde{i}, j \rightsquigarrow \tilde{j} : d(i, j) \leq \rho\right\}. \tag{5.15}$$

Define the sparsity pattern $S_\rho \subset I \times I$ as

$$S_\rho \coloneqq \left\{(i, j) \in I \times I \,\middle|\, \exists \tilde{i}, \tilde{j} \in \tilde{I} : i \rightsquigarrow \tilde{i}, j \rightsquigarrow \tilde{j}, \left(\tilde{i}, \tilde{j}\right) \in \tilde{S}_\rho\right\} \tag{5.16}$$

and call the elements of $\tilde{J}^{(k)}$ supernodes. *Color each $\tilde{j} \in \tilde{J}^{(k)}$ in one of $p^{(k)}$ colors such that no $\tilde{i}, \tilde{j} \in \tilde{J}^{(k)}$ with $\left(\tilde{i}, \tilde{j}\right) \in \tilde{S}_\rho$ have the same color. For $i \in J^{(k)}$ write node(i) for the $\tilde{i} \in \tilde{J}^{(k)}$ such that $i \rightsquigarrow \tilde{i}$ and write color$(\tilde{i})$ for the color of $\tilde{i}$. Define the supernodal multicolor ordering $\prec_\rho$ by reordering the elements of I such that*

(1) $i \prec_\rho j$ for $i \in J^{(k)}$, $j \in J^{(l)}$ and $k < l$;

(2) within each level $J^{(k)}$, we order the elements of supernodes colored in the same color consecutively, i.e. given $i, j \in J^{(k)}$ such that color(node(i)) $\neq$ color(node(j)), $i \prec_\rho j \implies i' \prec_\rho j'$ for color(node(i')) = color(node(i)), and color(node(j')) = color(node(j)); and

(3) the elements of each supernode appear consecutively, i.e. given $i, j \in J^{(k)}$ such that node$(i) \neq$ node(j), $i \prec_\rho j \implies i' \prec_\rho j'$ for node(i') = node(i), and node(j') = node(j).

Starting from a hierarchical ordering and sparsity pattern, the modified ordering and sparsity pattern can be obtained efficiently:

Lemma 14. *In the setting of Examples 1 and 2, given $\{(i, j) \mid d(i, j) \leq \rho\}$, there exist constants C and $p_{\max}$ depending only on the dimension d and the cost of computing $d(\cdot, \cdot)$ such that the ordering and sparsity pattern presented in Construction 3 can be constructed with $p^{(k)} \leq p_{\max}$, for each $1 \leq k \leq q$, in computational complexity $Cq\rho^d N$.*

Proof. The aggregation into supernodes can be done via a greedy algorithm by keeping track of all nodes that are not already within distance $\rho/2$ of a supernode and removing them one at a time. We can then go through ρ-neighbourhoods and remove points within distance $\rho/2$ from our list of candidates for future supernodes. To create the coloring, we use the greedy graph coloring of [125] on the undirected graph G with vertices $\tilde{J}^{(k)}$ and edges $\left\{(\tilde{i}, \tilde{j}) \in \tilde{S}_\rho \,\middle|\, \tilde{i}, \tilde{j} \in \tilde{J}^{(k)}\right\}$. Defining $\deg(G)$ as the maximum number of edges connected to any vertex of G, the computational complexity of greedy graph coloring is bounded above by $\deg(G)\#\left(J^{(k)}\right)$ and the number of colors used by $\deg(G) + 1$. A sphere-packing argument shows that $\deg(G)$ is at most a constant depending only on the dimension d, which yields the result. □

5.4.3 Proof of stability of incomplete Cholesky factorization in the supernodal multicolor ordering

We will now bound the approximation error of the Cholesky factors obtained from Algorithm 3, using the supernodal multicolor ordering and sparsity pattern described in Construction 3. For $\tilde{i}, \tilde{j} \in \tilde{I}$, let $\Theta_{\tilde{i},\tilde{j}}$ be the submatrix $(\Theta_{ij})_{i\in\tilde{i}, j\in\tilde{j}}$ and let $\sqrt{M}$ be the (dense and lower-triangular) Cholesky factor of a matrix M.

Algorithm 3 with supernodal multicolor ordering $\prec_\rho$ and sparsity pattern S_ρ is equivalent to the block-incomplete Cholesky factorization described in Algorithm 9 where the function `Restrict!`(Θ, S_ρ) sets all entries of Θ outside of S_ρ to zero.

Algorithm 9 Supernodal incomplete Cholesky factorization

Input: $\Theta \in \mathbb{R}^{I\times I}$ symmetric
Output: $L \in \mathbb{R}^{I\times I}$ lower triangular

```
Restrict!$(\Theta, S_\rho)$
for $\tilde{i} \in \tilde{I}$ do
    $L_{:,\tilde{i}} \leftarrow \Theta_{:,\tilde{i}} / \sqrt{\Theta_{\tilde{i},\tilde{i}}}^\top$
    for $\tilde{j} \succ \tilde{i} : (\tilde{i}, \tilde{j}) \in \tilde{S}$ do
        for $k \succ \tilde{j} : (\tilde{k}, \tilde{i}), (\tilde{k}, \tilde{j}) \in \tilde{S}$ do
            $\Theta_{\tilde{k},\tilde{j}} \leftarrow \Theta_{\tilde{k},\tilde{j}} - \Theta_{\tilde{k},\tilde{i}} (\Theta_{\tilde{i},\tilde{i}})^{-1} \Theta_{\tilde{j},\tilde{i}}$
        end for
    end for
end for
return $L$
```

We will now reformulate the above algorithm using the fact that the elimination of nodes of the same color, on the same level of the hierarchy, happens consecutively. Let p be the maximal number of colors used on any level of the hierarchy. We can then write $I = \bigcup_{1 \leq k \leq q, 1 \leq l \leq p} J^{(k,l)}$, where $J^{(k,l)}$ is the set of indices on level k colored in the color l. Let $\Theta_{(k,l),(m,n)}$ be the restriction of Θ to $J^{(k,l)} \times J^{(m,n)}$ and write $(m,n) \prec (k,l) \iff m < k$ or $(m = k$ and $n < l)$. We can then rewrite Algorithm 9 as:

Algorithm 10 Supernodal incomplete Cholesky factorization

Input: $\Theta \in \mathbb{R}^{I \times I}$ symmetric
Output: $L \in \mathbb{R}^{I \times I}$ lower triangular

```
for $1 \leq k \leq q$ do
  for $1 \leq l \leq p$ do
    Restrict!($\Theta, S_\rho$)
    $L_{(:,:),(k,l)} \leftarrow \Theta_{(:,:),(k,l)} / \sqrt{\Theta_{(k,l),(k,l)}}^\top$
    $\Theta \leftarrow \Theta - \Theta_{(:,:),(k,l)} \left(\Theta_{(k,l),(k,l)}\right)^{-1} \Theta_{(k,l),(:,:)}$
  end for
end for
return $L$
```

For $1 \leq k \leq q, 1 \leq l \leq p$ and a matrix $M \in \mathbb{R}^{I \times I}$ with $M_{(:,:),(m,n)}, M_{(m,n),(:,:)} = 0$ for all $(m,n) \prec (k,l)$, let $\mathbb{S}[M]$ be the matrix obtained by applying `Restrict!`(M, S_ρ) followed by the Schur complementation $M \leftarrow M - M_{(:,:),(k,l)} \left(M_{(k,l),(k,l)}\right)^{-1} M_{(k,l),(:,:)}$. We now prove a stability estimate for the operator $\mathbb{S}$. Let $M_{k,(m,n)}$ be the restriction of a matrix $M \in \mathbb{R}^{I \times I}$ to $J^{(k)} \times J^{(m,n)}$.

Lemma 15. *For $1 \leq k^\circ \leq q$ and $1 \leq l^\circ \leq p$ let $\Theta, E \in \mathbb{R}^{I \times I}$ be such that*

$$\Theta_{(:,:),(m,n)}, \Theta_{(m,n),(:,:)} = 0 \, \textit{for all } (m,n) \prec (k^\circ, l^\circ), \tag{5.17}$$

and (writing $\Theta_{k,l}$ for the $J^{(k)} \times J^{(l)}$ submatrix of Θ and $\lambda_{\max}$ for maximal singular values) define

$$\lambda_{\min} := \lambda_{\min}(\Theta_{k^\circ,k^\circ}), \qquad \lambda_{\max} := \max_{k^\circ \leq k \leq q} \lambda_{\max}(\Theta_{k^\circ,k}). \tag{5.18}$$

If

$$\max_{k^\circ \leq k,l \leq q} \|E_{k,l}\|_{\mathrm{Fro}} \leq \epsilon \leq \frac{\lambda_{\min}}{2}, \tag{5.19}$$

then the following perturbation estimate holds:

$$\max_{k^\circ \le k, l \le q} \left\| \left(\mathbb{S}[\Theta] - \mathbb{S}[\Theta + E] \right)_{k,l} \right\|_{\mathrm{Fro}} \le \left(\frac{3}{2} + 2\frac{\lambda_{\max}}{\lambda_{\min}} + 8\frac{\lambda_{\max}^2}{\lambda_{\min}^2} \right) \epsilon. \tag{5.20}$$

Proof. Write $\tilde{\Theta}$, $\tilde{E}$ for the versions of Θ, E set to zero outside of S_ρ. For $k^\circ \le k, l \le q$,

$$(\mathbb{S}[\Theta + E] - \mathbb{S}[\Theta])_{k,l} \tag{5.21}$$
$$= \tilde{\Theta}_{k,l} + \tilde{E}_{k,l} - \left(\tilde{\Theta} + \tilde{E}\right)_{k,(k^\circ,l^\circ)} \left(\tilde{\Theta} + \tilde{E}\right)^{-1}_{(k^\circ,l^\circ),(k^\circ,l^\circ)} \left(\tilde{\Theta} + \tilde{E}\right)_{(k^\circ,l^\circ),l} \tag{5.22}$$
$$- \tilde{\Theta}_{k,l} + \tilde{\Theta}_{k,(k^\circ,l^\circ)} \tilde{\Theta}^{-1}_{(k^\circ,l^\circ),(k^\circ,l^\circ)} \tilde{\Theta}_{(k^\circ,l^\circ),l} \tag{5.23}$$
$$= \tilde{E}_{k,l} + \left(\tilde{\Theta} + \tilde{E}\right)_{k,(k^\circ,l^\circ)} \left(\tilde{\Theta} + \tilde{E}\right)^{-1}_{(k^\circ,l^\circ),(k^\circ,l^\circ)} \tilde{E}_{(k^\circ,l^\circ),(k^\circ,l^\circ)} \tilde{\Theta}^{-1}_{(k^\circ,l^\circ),(k^\circ,l^\circ)} \left(\tilde{\Theta} + \tilde{E}\right)_{(k^\circ,l^\circ),l} \tag{5.24}$$
$$- \left(\tilde{\Theta} + \tilde{E}\right)_{k,(k^\circ,l^\circ)} \tilde{\Theta}^{-1}_{(k^\circ,l^\circ),(k^\circ,l^\circ)} \left(\tilde{\Theta} + \tilde{E}\right)_{(k^\circ,l^\circ),l} + \tilde{\Theta}_{k,(k^\circ,l^\circ)} \tilde{\Theta}^{-1}_{(k^\circ,l^\circ),(k^\circ,l^\circ)} \tilde{\Theta}_{(k^\circ,l^\circ),l} \tag{5.25}$$
$$= \tilde{E}_{k,l} + \left(\tilde{\Theta} + \tilde{E}\right)_{k,(k^\circ,l^\circ)} \left(\tilde{\Theta} + \tilde{E}\right)^{-1}_{(k^\circ,l^\circ),(k^\circ,l^\circ)} \tilde{E}_{(k^\circ,l^\circ),(k^\circ,l^\circ)} \tilde{\Theta}^{-1}_{(k^\circ,l^\circ),(k^\circ,l^\circ)} \left(\tilde{\Theta} + \tilde{E}\right)_{(k^\circ,l^\circ),l} \tag{5.26}$$
$$- \tilde{E}_{k,(k^\circ,l^\circ)} \tilde{\Theta}^{-1}_{(k^\circ,l^\circ),(k^\circ,l^\circ)} \tilde{\Theta}_{(k^\circ,l^\circ),l} - \tilde{\Theta}_{k,(k^\circ,l^\circ)} \tilde{\Theta}^{-1}_{(k^\circ,l^\circ),(k^\circ,l^\circ)} \tilde{E}_{(k^\circ,l^\circ),l} \tag{5.27}$$
$$- \tilde{E}_{k,(k^\circ,l^\circ)} \tilde{\Theta}^{-1}_{(k^\circ,l^\circ),(k^\circ,l^\circ)} \tilde{E}_{(k^\circ,l^\circ),l}, \tag{5.28}$$

where the second equality follows from the matrix identity

$$(A + B)^{-1} = A^{-1} - (A + B)^{-1} BA^{-1}. \tag{5.29}$$

Now recall that, for all $A \in \mathbb{R}^{n \times m}, B \in \mathbb{R}^{m \times s}$, $\|M\| \le \|M\|_{\mathrm{Fro}}$ and $\|AB\|_{\mathrm{Fro}} \le \|A\|\|B\|_{\mathrm{Fro}}$. Therefore, $\|(A + E)^{-1}\| \le 2/\lambda_{\min}$ and $\|A + E\| \le 2\lambda_{\max}$. Combining these estimates and using the triangle inequality yields

$$\left\| (\mathbb{S}[A + E] - \mathbb{S}[A])_{k,l} \right\|_{\mathrm{Fro}} \tag{5.30}$$
$$\le \|E_{k,l}\|_{\mathrm{Fro}} + 8\frac{\lambda_{\max}^2}{\lambda_{\min}^2} \|E_{k^\circ,k^\circ}\|_{\mathrm{Fro}} + \frac{\lambda_{\max}}{\lambda_{\min}} (\|E_{k,l}\|_{\mathrm{Fro}} + \|E_{l,k}\|_{\mathrm{Fro}}) \tag{5.31}$$
$$+ \lambda_{\min}^{-1} \|E_{k,k^\circ}\|_{\mathrm{Fro}} \|E_{k^\circ,l}\|_{\mathrm{Fro}} \tag{5.32}$$
$$\le \left(1 + 8\frac{\lambda_{\max}^2}{\lambda_{\min}^2} + 2\frac{\lambda_{\max}}{\lambda_{\min}} + \frac{\epsilon}{\lambda_{\min}} \right) \epsilon \tag{5.33}$$
$$\le \left(\frac{3}{2} + 2\frac{\lambda_{\max}}{\lambda_{\min}} + 8\frac{\lambda_{\max}^2}{\lambda_{\min}^2} \right) \epsilon. \tag{5.34}$$

□

Recursive application of the above lemma gives a stability result for the incomplete Cholesky factorization.

Lemma 16. *For $\rho > 0$, let $\prec_\rho$ and S_ρ be a supernodal ordering and sparsity pattern such that the maximal number of colors used on each level is at most p. Let L^{S_ρ} be an invertible lower-triangular matrix with nonzero pattern S_ρ and define $M := L^{S_\rho} L^{S_\rho,\top}$. Assume that M satisfies Condition 2 with constant κ. Then there exists a universal constant C such that, for all $0 < \epsilon < \frac{\lambda_{\min}(M)}{2q^2(C\kappa)^{2qp}}$ and all $E \in \mathbb{R}^{I \times I}$ with $\|E\|_{\mathrm{Fro}} \leq \epsilon$,*

$$\left\|M - \tilde{L}^{S_\rho} \tilde{L}^{S_\rho,\top}\right\|_{\mathrm{Fro}} \leq q^2 (C\kappa)^{2qp} \epsilon, \tag{5.35}$$

where $\tilde{L}^{(S_\rho)}$ is the Cholesky factor obtained by applying Algorithm 10 to $M + E$.

Proof. The result follows from applying Lemma 15 at each step of Algorithm 10. □

5.4.4 Conclusion

Using the stability result in Lemma 16, we can finally prove that when using the supernodal multicolor ordering and sparsity pattern incomplete Cholesky factorization applied to Θ attains an ϵ-accurate Cholesky factorization in computational complexity $O\left(N \log^2(N) \log^{2d}(N/\epsilon)\right)$.

Theorem 18. *In the setting of Examples 1 and 2, there exists a constant C depending only on $d, s, \|\mathcal{L}\|, \|\mathcal{L}^{-1}\|$, h, and δ such that, given the ordering $\prec_\rho$ and sparsity pattern S_ρ defined as in Construction 3 with $\rho \geq C \log(N/\epsilon)$, the incomplete Cholesky factor L obtained from Algorithm 3 has accuracy*

$$\|LL^T - \Theta\|_{\mathrm{Fro}} \leq \epsilon. \tag{5.36}$$

Furthermore, Algorithm 3 has complexity of at most $CN\rho^{2d} \log^2(N)$ in time and at most $CN\rho^d \log(N)$ in space.

Proof of Theorem 18. Theorem 9 implies that by choosing $\rho \geq \tilde{C} \log(N/\epsilon)$, there exists a lower-triangular matrix $\tilde{L}^{S_\rho}$ with $\left\|\Theta - \tilde{L}^{S_\rho} \tilde{L}^{S_\rho,\top}\right\|_{\mathrm{Fro}} \leq \epsilon$ and sparsity pattern S_ρ. Theorem 7 implies that the Examples 1 and 2 satisfy $\lambda_{\min} \geq 1/\mathrm{poly}(N)$. Therefore, choosing $\rho \geq \tilde{C} \log N$ ensures that $\epsilon < \frac{\lambda_{\min}(\Theta)}{2}$ and thus that $\tilde{\Theta} := \tilde{L}^{S_\rho} \tilde{L}^{S_\rho,\top}$ satisfies Condition 2 with constant $2C_\Phi$, where C_Φ is the corresponding constant for Θ. By possibly changing $\tilde{C}$ again, $\rho \geq \tilde{C} \log N$ ensures that

$$\epsilon \leq \frac{\lambda_{\min}(\Theta)}{2q^2 \left(C\kappa(\tilde{\Theta})\right)^{2qp}},$$

where C is the constant of Lemma 16, since $q \approx \log N$ and, by Lemma 14, p is bounded independently of N. Thus, by Lemma 16, the Cholesky factor L^{S_ρ} obtained from applying Algorithm 10 to $\Theta = \tilde{\Theta} + \left(\Theta - \tilde{\Theta}\right)$ satisfies

$$\left\|\tilde{\Theta} - L^{S_\rho} L^{S_\rho,\top}\right\|_{\mathrm{Fro}} \leq q^2 \left(4C\kappa\right)^{2qp} \epsilon \leq \mathrm{poly}(N)\epsilon, \tag{5.37}$$

where κ is the constant with which Θ satisfies Condition 2 and the polynomial depends only on C, κ, and p. Since, for the ordering $\prec_\rho$ and sparsity pattern S_ρ, the Cholesky factors obtained via Algorithms 3 and 10 coincide, we obtain the result. □

This result holds for both element-wise and supernodal factorization, in either its left, up, or right-looking forms. As remarked in Section 5.3.1, using the two-way supernodal sparsity pattern for factorization of Θ^{-1} in the fine-to-coarse ordering degrades the asymptotic complexity. Therefore, the above result does not immediately prove the accuracy of the Cholesky factorization in this setting, with optimal complexity. However, the column-supernodal factorization described in Section 5.3 can similarly be described in terms of $O\left(\log(N)\right)$ Schur-complementations. Thus, the above proof can be modified to show that when using the column-supernodal multicolor ordering and sparsity pattern, `ICHOL(0)`applied to Θ^{-1} computes an ϵ-approximation in computational complexity $O\left(N \log(N/\epsilon)\right)$.

5.5 Numerical example: Compression of dense kernel matrices

5.5.1 Selection of the sparsity pattern and ordering

This section introduces an $O(\rho^d N \log^2 N)$-complexity algorithm (Algorithm 11) for computing the maximin ordering and sparsity pattern introduced in Section 3.5. This algorithm does not explicitly query the position of the $\{x_i\}_{i\in I}$ and only uses pairwise distances by processing points one by one by updating a mutable binary heap, keeping track of the point to be processed at each step. With this approach, our proposed algorithm is oblivious to the dimension d of the ambient space and, in particular, can automatically exploit low-dimensional structure in the point cloud $\{x_i\}_{i\in I}$. In order to avoid computing all $O(N^2)$ pairwise distances, Algorithm 11 uses the sparsity pattern obtained on the coarser scales to restrict computation at the finer scales to local neighborhoods.

Theorem 19. *The output of Algorithm 11 are the maximin ordering and sparsity pattern described in Section 3.5, with starting set given by $\partial\Omega$. Furthermore, in the*

Algorithm 11 Ordering and sparsity pattern algorithm.

Input: Real $\rho \geq 2$ and Oracles $\texttt{dist}(\cdot,\cdot), \texttt{dist}_{\partial\Omega}(\cdot)$ such that $\texttt{dist}(i,j) = \operatorname{dist}\left(x_i, x_j\right)$ and $\texttt{dist}_{\partial\Omega}(i) = \operatorname{dist}(x_i, \partial\Omega)$

Output: An array $l[:]$ of distances, an array P encoding the multiresolution ordering, and an array of index pairs S containing the sparsity pattern.

```
1: $P = \emptyset$
2: for $i \in \{1, \ldots, N\}$ do
3:     $l[i] \leftarrow \texttt{dist}_{\partial\Omega}(i)$
4:     $p[i] \leftarrow \emptyset$
5:     $c[i] \leftarrow \emptyset$
6: end for
7: {Creates a mutable binary heap, containing pairs of indices and distances as elements:}
8: $H \leftarrow \texttt{MutableMaximalBinaryHeap}\left(\{(i, l[i])\}_{i \in \{1,\ldots,N\}}\right)$
9: {Instates the Heap property, with a pair with maximal distance occupying the root of the heap:}
10: $\texttt{heapSort!}(H)$
11: {Processing the first index:}
12: {Get the root of the heap, remove it, and restore the heap property:}
13: $(i, l) = \texttt{pop}(H)$
14: {Add the index as the next element of the ordering} $\texttt{push}(P, i)$
15: for $j \in \{1, \ldots, N\}$ do
16:     $\texttt{push}(c[i], j)$
17:     $\texttt{push}(p[j], i)$
18:     $\texttt{sort!}(c[i], \texttt{dist}(\cdot, i))$
19:     $\texttt{decrease!}(H, j, \texttt{dist}(i, j))$
20: end for
21: {Processing remaining indices:}
22: while $H \neq \emptyset$ do
23:     {Get the root of the heap, remove it, and restore the heap property:} $(i, l) = \texttt{pop}(H)$  $\ell[i] \leftarrow l$
24:     {Select the parent node that has all possible children of $i$ amongst its children, and is closest to $i$:}
25:     $k = \arg\min_{j \in p[i] : \texttt{dist}(i,j) + \rho\ell[i] \leq \rho\ell[j]} \texttt{dist}(i, j)$
26:     {Loop through those children of $k$ that are close enough to $k$ to possibly be children of $i$:}
27:     for $j \in c[k] : \texttt{dist}(j, k) \leq \texttt{dist}(i, k) + \rho\ell[i]$ do
28:         $\texttt{decrease!}(H, j, \texttt{dist}(i, j))$
29:         if $\texttt{dist}(i, j) \leq \rho l[i]$ then
30:             $\texttt{push}(c[i], j)$
31:             $\texttt{push}(p[j], i)$
32:         end if
33:     end for
34:     {Add the index as the next element of the ordering}
35:     $\texttt{push}(P, i)$
36:     {Sort the children according to distance to the parent node, so that the closest children can be found more easily} $\texttt{sort!}(c[i], \texttt{dist}(\cdot, i))$
37: end while
38: {Aggregating the lists of children into the sparsity pattern:}
39: for $i \in \{1, \ldots, N\}$ do
40:     for $j \in c[i]$ do
41:         $\texttt{push!}(S, (i, j))$
42:         $\texttt{push!}(S, (j, j))$
43:     end for
44: end for
```

setting of Theorem 11, if the oracles $\texttt{dist}(\cdot,\cdot)$ *and* $\texttt{dist}_{\partial\Omega}(\cdot)$ *can be queried in complexity* $O(1)$*, then the complexity of Algorithm 11 is bounded by* $C\rho^d N \log^2 N$*, where* C *is a constant depending only on* d*,* Ω *and* δ*.*

Theorem 19 is proved in Section .2.1. As discussed therein, in the case $\Omega = \mathbb{R}^d$, Algorithm 11 has the advantage that its computational complexity depends only on the intrinsic dimension of the dataset, which can be much smaller than d itself.

5.5.2 The case of the whole space ($\Omega = \mathbb{R}^d$)

Many applications in Gaussian process statistics and machine learning are in the $\Omega = \mathbb{R}^d$ setting. In that setting, the Matérn family of kernels (5.42) is a popular choice that is equivalent to using the whole-space Green's function of an elliptic PDE as covariance function [248, 249]. Let $\bar{\Omega}$ be a bounded domain containing the $\{x_i\}_{i\in I}$. The case $\Omega = \mathbb{R}^d$ is not covered in Theorem 9 and the resulting theorems because in this case, the screening effect is weakened near the boundary of $\bar{\Omega}$ by the absence of measurements points outside of $\bar{\Omega}$. Therefore, distant points close to the boundary of $\bar{\Omega}$ will have stronger conditional correlations than similarly distant points in the interior of $\bar{\Omega}$ (see Figure 5.4). As observed by [207] and [60], Markov random field (MRF) approaches that use a discretization of the underlying PDE face similar challenges at the boundary. While the weakening of the exponential decay at the boundary worsens the accuracy of our method, the numerical results in Section 5.5.4 (which are all obtained without imposing boundary conditions) suggest that its overall impact is limited. In particular, as shown in Figure 5.4, it does not cause significant artifacts in the quality of the approximation near the boundary. This differs from the significant boundary artifacts of MRF methods, which have to be mitigated against by careful calibration of the boundary conditions [60, 207]. Although the numerical results presented in this section are mostly obtained with $x_i \sim \mathrm{UNIF}([0,1]^d)$, in many practical applications, the density of measurement points will slowly (rather than abruptly) decrease towards zero near the boundary of the sampled domain, which drastically decreases the boundary errors shown above. Accuracy can also be enhanced by adding *artificial* points $\{x_i\}_{i\in\tilde{I}}$ at the boundary. By applying the Cholesky factorization to $\{x_i\}_{i\in I\cup\tilde{I}}$, and then restricting the resulting matrix to $I \times I$, we can obtain a very accurate approximate matrix-vector multiplication. This approximation can be efficiently inverted using iterative methods such as conjugate gradient [218], preconditioned with the Cholesky factorization obtained from the original set of points.

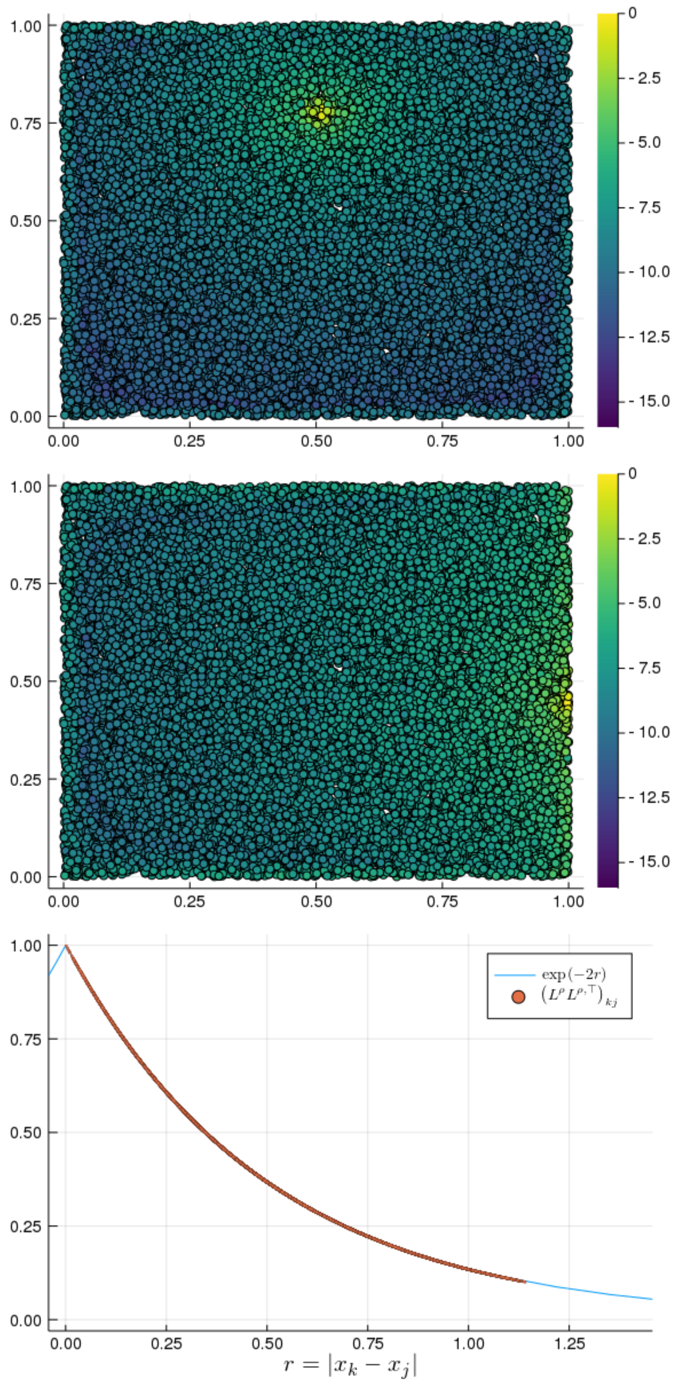


Figure 5.4: **Weaker screening between boundary points.** Left and center: i^{th} (left) and j^{th} (center) column of the Cholesky factor L (normalized to unit diagonal) of Θ in maximin ordering, where x_i is an interior point and x_j is near the boundary. Although $l[i]$ is of the order of $l[j]$, the exponential decay of $L_{:,j}$ near the boundary is significantly weakened by the absence of Dirichlet boundary conditions. Right: approximate correlations $\left\{(L^{\rho} L^{\rho,\top})_{kj}\right\}_{k \in I}$ (with $\rho = 3.0$) and true covariance function $\exp(-2r)$ with $r = |x_k - x_j|$. Correlations between x_j and remaining points are captured accurately, despite the weakened exponential decay near the boundary.

5.5.3 Nuggets and measurement errors

In the Gaussian process regression setting it is common to model measurement error by adding a *nugget* $\sigma^2\mathrm{Id}$ to the covariance matrix:

$$\tilde{\Theta} = \Theta + \sigma^2\mathrm{Id}. \tag{5.38}$$

The addition of a diagonal matrix diminishes the screening effect and thus the accuracy of Algorithm 3. This problem can be avoided by rewriting the modified covariance matrix $\tilde{\Theta}$ as

$$\tilde{\Theta} = \Theta(\sigma^2 A + \mathrm{Id}), \tag{5.39}$$

where $A := \Theta^{-1}$. As noted in Section 2.2.3, A can be interpreted as a discretized partial differential operator and has therefore near-sparse Cholesky factors in the reverse elimination ordering. Adding a multiple of the identity to A amounts to adding a zeroth-order term to the underlying PDE and thus preserves the sparsity of the Cholesky factors. This leads to the sparse decomposition

$$\tilde{\Theta} = LL^\top P^{\updownarrow}\tilde{L}\tilde{L}^\top P^{\updownarrow}, \tag{5.40}$$

where $P^{\updownarrow}$ is the order-reversing permutation and $\tilde{L}$ is the Cholesky factor of $P^{\updownarrow}(\sigma^2 A + \mathrm{Id})P^{\updownarrow}$. Figure 5.5 shows that the exponential decay of these Cholesky factors is robust with respect to σ.

5.5.4 Numerical results

We will now present numerical evidence in support of our results. All experiments reported below were run on a workstation using an Intel®Core™i7-6400 CPU with 4.00GHz and 64 GB of RAM, without using parallelism. In the following, $\mathrm{nnz}(L)$ denotes the number of nonzero entries of the lower-triangular factor L; $t_{\texttt{SortSparse}}$ denotes the time taken to compute the maximin ordering $\prec$ and sparsity pattern S_ρ using Algorithm 11; $t_{\texttt{Entries}}$ denotes the time taken to compute the entries of Θ on S_ρ; and $t_{\texttt{ICHOL(0)}}$ denotes the time taken to perform 4 (`ICHOL(0)`), all measured in seconds. The relative error in Frobenius norm is approximated by

$$E := \frac{\|LL^\top - \Theta\|_{\mathrm{Fro}}}{\|\Theta\|_{\mathrm{Fro}}} \approx \frac{\sqrt{\sum_{k=1}^{m}\left\|\left(LL^\top - \Theta\right)_{i_k j_k}\right\|^2}}{\sqrt{\sum_{k=1}^{m}\|\Theta_{i_k j_k}\|^2}}, \tag{5.41}$$

where the $m = 500000$ pairs of indices $i_k, j_k \sim \mathrm{UNIF}(I)$ are independently and uniformly distributed in I. This experiment is repeated 50 times and the resulting mean and standard deviation (in brackets) are reported. For measurements in $[0,1]^d$,

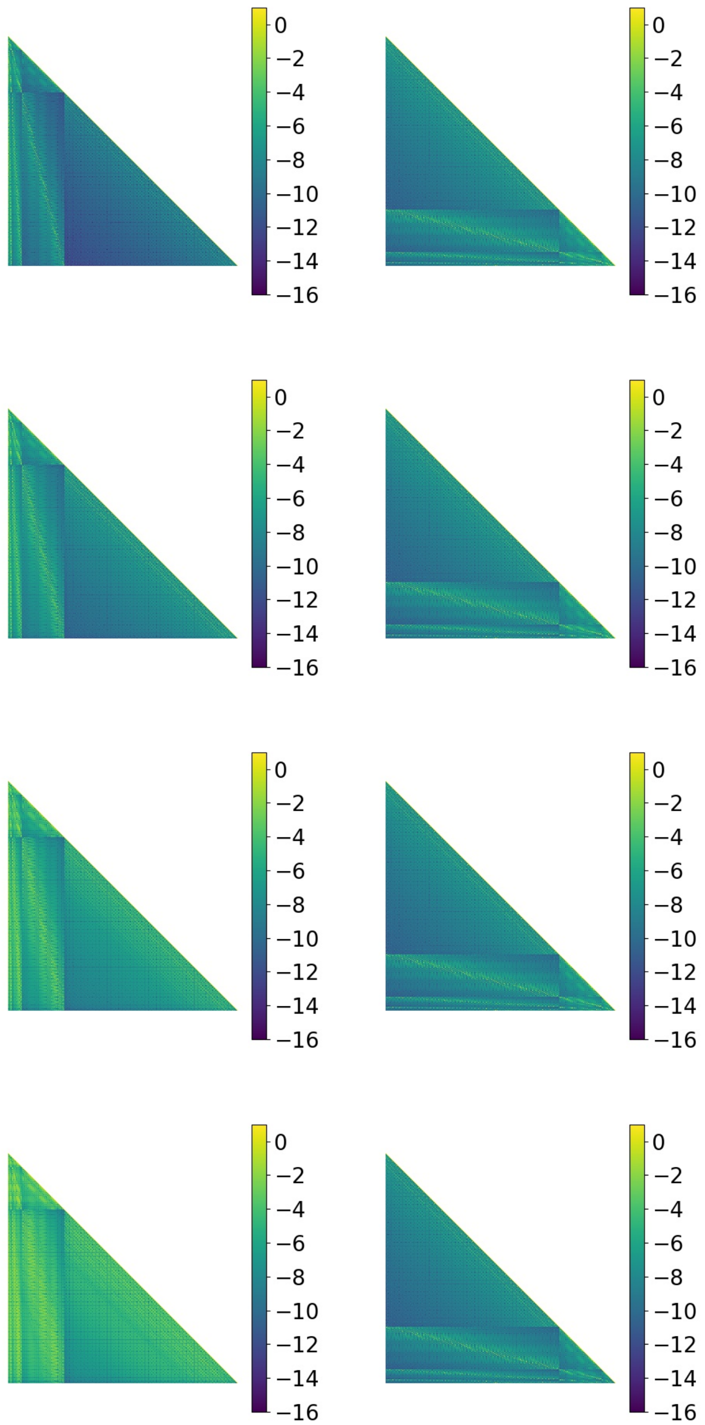


Figure 5.5: **(Lack of) robustness to varying size of the nugget.** We plot the $\log_{10}$ of the magnitude of the Cholesky factors of $\Theta + \sigma^2 \mathrm{Id}$ in maximin ordering (first column) and of $A + \sigma^2$ in reverse maximin ordering (second column). As we increase $\sigma^2 \in [0.0, 0.1, 1.0, 10.0]$ from left to right the decay of the Cholesky factors of $\Theta + \sigma^2 \mathrm{Id}$ deteriorates, and that of the factors of $A + \sigma^2 \mathrm{Id}$ is preserved.

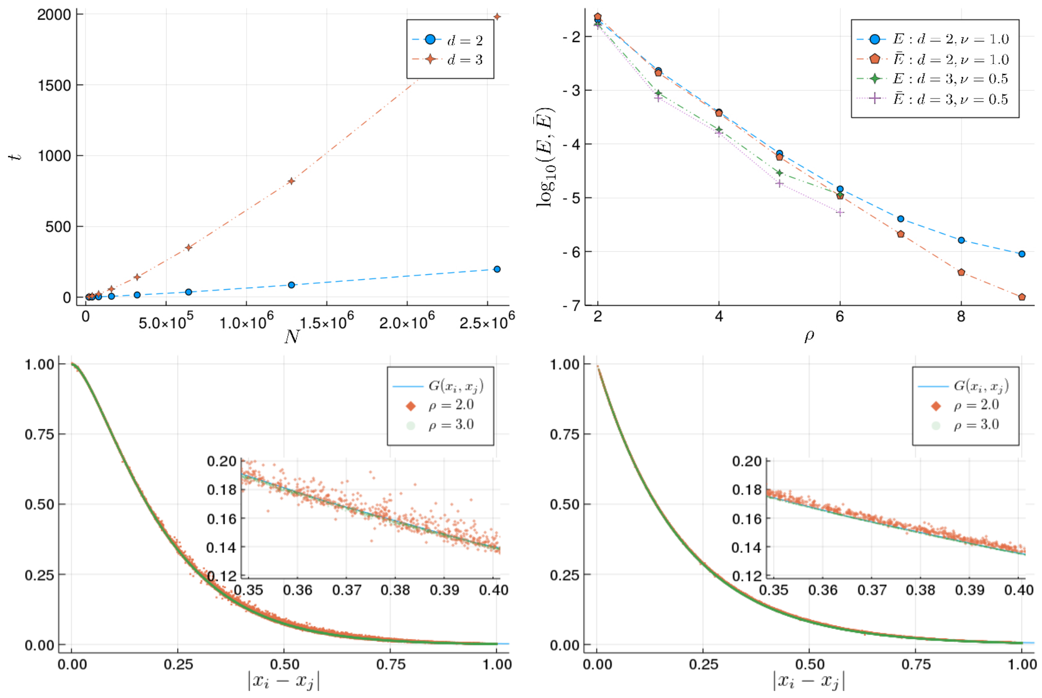


Figure 5.6: **Accuracy and computational cost.** First panel: the increase in computational time taken by the Cholesky factorization, as N increases (for $\rho = 3.0$). Second panel: the exponential decay of the relative error in Frobenius norm, as ρ is increased. In the third ($d = 2$) and fourth panel ($d = 3$), we see the comparison of the approximate and true covariance for $\rho = 2.0$ and $\rho = 3.0$.

in order to isolate the boundary effects, we also consider the quantity $\bar{E}$ which is defined as E, but with only those sample i_k, j_k for which $x_{i_k}, x_{j_k} \in [0.05, 0.95]^d$. Most of our experiments will use the Matérn class of covariance functions [167], defined by

$$G_{l,\nu}^{\text{Matérn}}(x, y) := \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}|x - y|}{l} \right)^{\nu} K_{\nu} \left(\frac{\sqrt{2\nu}|x - y|}{l} \right), \tag{5.42}$$

where K_ν is the modified Bessel function of second kind [1, Section 9.6] and ν, l are parameters describing the degree of smoothness, and the length-scale of interactions, respectively [203]. In Figure 5.7, the Matérn kernel is plotted for different degrees of smoothness. The Matérn covariance function is used in many branches of statistics and machine learning to model random fields with finite order of smoothness [109, 203].

As observed by [248, 249], the Matérn kernel is the Green's function of an elliptic PDE of possibly fractional order $2(\nu + d/2)$ in the whole space. Therefore, for $2(\nu + d/2) \in \mathbb{N}$, the Matérn kernel falls into the framework of our theoretical

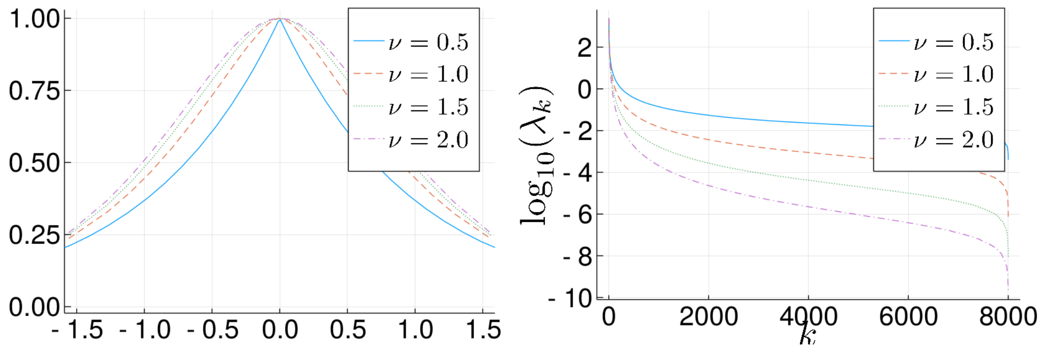


Figure 5.7: **The Matérn class of covariance functions.** Matérn kernels for different values of ν (left), and the spectrum of Θ, for 2000 points $x_i \in [0,1]^2$ (right). Smaller values of ν correspond to stronger singularities at zero and hence lower degrees of smoothness of the associated Gaussian process.

Table 5.1: $G^{\text{Matérn}}_{\nu,l}$, with $\nu = 0.5$, $l = 0.2$, $\rho = 3.0$, and $d = 2$.

N	$\text{nnz}(L)/N^2$	$\text{rank}(L)$	$t_{\texttt{SortSparse}}$	$t_{\texttt{Entries}}$	$t_{\texttt{ICHOL(0)}}$	E	$\bar{E}$
20000	5.26e-03	20000	0.71	0.81	0.42	1.25e-03 (3.68e-06)	1.11e-03 (3.01e-06)
20000	5.26e-03	20000	0.71	0.81	0.42	1.25e-03 (3.68e-06)	1.11e-03 (3.01e-06)
40000	2.94e-03	40000	1.21	1.19	1.00	1.27e-03 (3.32e-06)	1.12e-03 (3.56e-06)
80000	1.62e-03	80000	2.72	2.82	2.55	1.30e-03 (3.20e-06)	1.21e-03 (3.29e-06)
160000	8.91e-04	160000	6.86	6.03	6.11	1.28e-03 (3.57e-06)	1.16e-03 (3.32e-06)
320000	4.84e-04	320000	17.22	13.79	15.66	1.23e-03 (3.19e-06)	1.11e-03 (2.40e-06)
640000	2.63e-04	640000	41.40	31.02	36.02	1.24e-03 (2.58e-06)	1.09e-03 (3.02e-06)
1280000	1.41e-04	1280000	98.34	65.96	85.99	1.23e-03 (3.72e-06)	1.10e-03 (3.74e-06)
2560000	7.55e-05	2560000	233.92	148.43	197.52	1.16e-03 (2.82e-06)	1.04e-03 (3.36e-06)

results, up to the behavior at the boundary discussed in Section 5.5.2. Since the locations of our points will be chosen at random, some of the points will be very close to each other, resulting in an almost singular matrix Θ that can become nonpositive under the approximation introduced by `ICHOL(0)`. If Algorithm 3 encounters a nonpositive pivot A_{ii}, then we set the corresponding column of L to zero, resulting in a low-rank approximation of the original covariance matrix. We report the rank of L in our experiments and note that we obtain a full-rank approximation for moderate values of ρ.

We begin by investigating the scaling of our algorithm as N increases. To this end, we consider $\nu = 0.5$ (the exponential kernel), $l = 0.2$ and choose N randomly distributed points in $[0,1]^d$ for $d \in \{2,3\}$. The results are summarized in Tables 5.1 and 5.4, and in Figure 5.6, and confirm the near-linear computational complexity of our algorithm.

Next, we investigate the trade-off between the computational efficiency and accuracy of the approximation. To this end, we choose $d = 2$, $\nu = 1.0$ and $d = 3$, $\nu = 0.5$,

Table 5.2: $G_{\nu,l}^{\text{Matérn}}$, with $\nu = 0.5$, $l = 0.2$, $\rho = 3.0$, and $d = 3$.

N	$\mathrm{nnz}(L)/N^2$	$\mathrm{rank}(L)$	$t_{\texttt{SortSparse}}$	$t_{\texttt{Entries}}$	$t_{\texttt{ICHOL(0)}}$	E	$\bar{E}$
20000	1.30e-02	20000	1.61	1.44	2.94	1.49e-03 (5.00e-06)	1.20e-03 (5.09e-06)
40000	7.60e-03	40000	3.26	3.32	8.33	1.21e-03 (4.29e-06)	9.91e-04 (3.72e-06)
80000	4.35e-03	80000	7.46	7.64	22.46	1.06e-03 (3.74e-06)	8.51e-04 (2.93e-06)
160000	2.45e-03	160000	20.95	18.42	57.64	9.81e-04 (2.33e-06)	7.88e-04 (3.23e-06)
320000	1.37e-03	320000	53.58	40.72	141.46	9.27e-04 (2.26e-06)	7.53e-04 (2.72e-06)
640000	7.61e-04	640000	133.55	96.67	350.10	8.98e-04 (3.25e-06)	7.25e-04 (3.02e-06)
1280000	4.19e-04	1280000	312.43	212.57	820.07	8.59e-04 (2.79e-06)	7.00e-04 (2.87e-06)
2560000	2.29e-04	2560000	795.68	480.17	1981.92	8.96e-04 (2.76e-06)	7.73e-04 (4.28e-06)

Table 5.3: $G_{\nu,l}^{\text{Matérn}}$, with $\nu = 1.0$, $l = 0.2$, $N = 10^6$, and $d = 2$.

	$\mathrm{nnz}(L)/N^2$	$\mathrm{rank}(L)$	$t_{\texttt{SortSparse}}$	$t_{\texttt{Entries}}$	$t_{\texttt{ICHOL(0)}}$	E	$\bar{E}$
$\rho = 2.0$	8.78e-05	254666	38.06	33.72	17.54	2.04e-02 (1.73e-02)	2.34e-02 (2.75e-02)
$\rho = 3.0$	1.76e-04	964858	71.07	67.85	61.35	2.32e-03 (6.02e-06)	2.09e-03 (7.50e-06)
$\rho = 4.0$	2.90e-04	999810	115.07	112.56	152.93	3.92e-04 (1.44e-06)	3.72e-04 (2.32e-06)
$\rho = 5.0$	4.26e-04	999999	165.91	166.60	312.19	6.70e-05 (2.98e-07)	5.68e-05 (2.55e-07)
$\rho = 6.0$	5.83e-04	1000000	227.62	229.76	566.94	1.45e-05 (6.69e-08)	1.08e-05 (5.01e-08)
$\rho = 7.0$	7.59e-04	1000000	292.52	300.65	944.33	4.05e-06 (4.96e-08)	2.10e-06 (1.69e-08)
$\rho = 8.0$	9.53e-04	1000000	363.90	380.07	1476.71	1.62e-06 (2.30e-08)	4.08e-07 (9.47e-09)
$\rho = 9.0$	1.16e-03	1000000	447.47	467.07	2200.32	8.98e-07 (1.44e-08)	1.42e-07 (5.14e-09)

Table 5.4: $G_{\nu,l}^{\text{Matérn}}$, with $\nu = 0.5$, $l = 0.2$, $N = 10^6$, and $d = 3$.

	$\mathrm{nnz}(L)/N^2$	$\mathrm{rank}(L)$	$t_{\texttt{SortSparse}}$	$t_{\texttt{Entries}}$	$t_{\texttt{ICHOL(0)}}$	E	$\bar{E}$
$\rho = 2.0$	1.87e-04	998046	87.83	56.44	85.20	1.69e-02 (6.89e-04)	1.60e-02 (3.36e-04)
$\rho = 3.0$	5.17e-04	1000000	226.84	158.42	599.86	8.81e-04 (3.21e-06)	7.15e-04 (2.99e-06)
$\rho = 4.0$	1.05e-03	1000000	446.52	326.27	2434.52	1.85e-04 (5.37e-07)	1.59e-04 (5.30e-07)
$\rho = 5.0$	1.82e-03	1000000	747.65	567.06	7227.45	2.89e-05 (1.94e-07)	1.84e-05 (1.15e-07)
$\rho = 6.0$	2.82e-03	1000000	1344.59	928.27	17640.58	1.15e-05 (1.06e-07)	5.34e-06 (5.34e-08)

Table 5.5: We tabulate the approximation rank and error for $\rho = 5.0$ and $N = 10^6$ points uniformly distributed in $[0,1]^3$. The covariance function is $G_{\nu,0.2}^{\text{Matérn}}$ for ν ranging around $\nu = 0.5$ and $\nu = 1.5$. Even though the intermediate values of ν correspond to a fractional order elliptic PDE, the behavior of the approximation stays the same.

	$\nu = 0.3$	$\nu = 0.5$	$\nu = 0.7$	$\nu = 0.9$	$\nu = 1.1$	$\nu = 1.3$	$\nu = 1.5$	$\nu = 1.7$
$\mathrm{rank}(L)$	1000000	1000000	1000000	1000000	1000000	1000000	1000000	999893
E	7.04e-05	2.89e-05	2.49e-05	3.58e-05	6.03e-05	8.77e-05	1.18e-04	1.46e-04
	(3.98e-07)	(1.79e-07)	(1.11e-07)	(1.19e-07)	(2.37e-07)	(3.06e-07)	(4.52e-07)	(5.39e-07)
$\bar{E}$	5.19e-05	1.85e-05	1.77e-05	2.82e-05	4.88e-05	6.87e-05	9.06e-05	1.13e-04
	(2.26e-07)	(1.18e-07)	(8.11e-08)	(1.30e-07)	(2.37e-07)	(3.50e-07)	(5.14e-07)	(5.45e-07)

Table 5.6: $G^{\text{Cauchy}}_{l,\alpha,\beta}$ for $(l,\alpha,\beta) = (0.4, 0.5, 0.025)$ (first table) and $(l,\alpha,\beta) = (0.2, 1.0, 0.20)$ (second table), for $N = 10^6$ and $d = 2$.

	$\rho = 2.0$	$\rho = 3.0$	$\rho = 4.0$	$\rho = 5.0$	$\rho = 6.0$	$\rho = 7.0$	$\rho = 8.0$	$\rho = 9.0$
rank(L)	999923	1000000	1000000	1000000	1000000	1000000	1000000	1000000
E	4.65e-04	5.98e-05	2.36e-05	1.19e-05	4.84e-06	4.17e-06	2.25e-06	1.42e-06
	(4.23e-07)	(1.56e-07)	(9.53e-08)	(6.32e-08)	(4.14e-08)	(4.99e-08)	(1.86e-08)	(1.64e-08)
$\bar{E}$	3.81e-04	3.49e-05	9.83e-06	4.65e-06	1.47e-06	8.49e-07	4.25e-07	2.12e-07
	(4.98e-07)	(1.59e-07)	(5.56e-08)	(2.63e-08)	(7.73e-09)	(1.04e-08)	(4.81e-09)	(3.24e-09)

	$\rho = 2.0$	$\rho = 3.0$	$\rho = 4.0$	$\rho = 5.0$	$\rho = 6.0$	$\rho = 7.0$	$\rho = 8.0$	$\rho = 9.0$
rank(L)	999547	1000000	1000000	1000000	1000000	1000000	1000000	1000000
E	1.08e-03	1.36e-04	2.89e-05	2.35e-05	5.33e-06	3.25e-06	2.53e-06	1.68e-06
	(5.02e-06)	(6.27e-07)	(2.63e-07)	(3.01e-07)	(6.15e-08)	(5.74e-08)	(4.84e-08)	(4.25e-08)
$\bar{E}$	7.23e-04	8.96e-05	1.17e-05	5.65e-06	1.09e-06	5.84e-07	4.03e-07	2.40e-07
	(4.07e-06)	(2.63e-07)	(7.10e-08)	(1.47e-07)	(7.71e-09)	(5.48e-09)	(3.44e-09)	(2.23e-09)

Table 5.7: $G^{\text{Matérn}}_{\nu,l}$ for $\nu = 0.5$, $l = 0.2$, and $\rho = 3.0$ with $N = 10^6$ points chosen as in Figure 5.8.

	$\delta_z = 0.0$	$\delta_z = 0.1$	$\delta_z = 0.2$	$\delta_z = 0.3$	$\delta_z = 0.4$	$\delta_z = 0.5$	$\delta_z = 0.6$
$\frac{\text{nnz}(L)}{N^2}$	1.76e-04	1.77e-04	1.78e-04	1.80e-04	1.82e-04	1.84e-04	1.85e-04
$t_{\texttt{ICHOL(0)}}$	61.92	62.15	62.81	64.27	64.87	65.50	66.12
rank(L)	1000000	1000000	1000000	1000000	1000000	1000000	1000000
E	1.17e-03	1.11e-03	1.28e-03	1.60e-03	1.72e-03	1.89e-03	2.11e-03
	(2.74e-06)	(3.00e-06)	(2.73e-06)	(4.28e-06)	(3.95e-06)	(5.11e-06)	(5.07e-06)

corresponding to fourth-order equations in two and three dimensions. We choose $N = 10^6$ data points $x_i \sim \text{UNIF}([0,1]^d)$ and apply our method with different values of ρ. The results of these experiments are tabulated in Tables 5.3 and 5.4 and the impact of ρ on the approximation error is visualized in Figure 5.6.

While our theoretical results only cover integer-order elliptic PDEs, we observe no practical difference between the numerical results for Matérn kernels corresponding to integer- and fractional-order smoothness. As an illustration, for the case $d = 3$, we provide approximation results for ν ranging around $\nu = 0.5$ (corresponding to a fourth-order elliptic PDE) and $\nu = 1.5$ (corresponding to a sixth-order elliptic PDE). As seen in Table 5.5, the results vary continuously as ν changes, with no qualitative differences between the behavior for integer- and fractional-order PDEs. To further illustrate the robustness of our method, we consider the Cauchy class of covariance functions introduced in [95]

$$G^{\text{Cauchy}}_{l,\alpha,\beta}(x,y) := \left(1 + \left(\frac{|x-y|}{l}\right)^{\alpha}\right)^{-\frac{\beta}{\alpha}}. \tag{5.43}$$

As far as we are aware, the Cauchy class has not been associated to any elliptic PDE. Furthermore, it does not have exponential decay in the limit $|x-y| \to \infty$, which allows us to emphasize the point that the exponential decay of the error is *not* due to

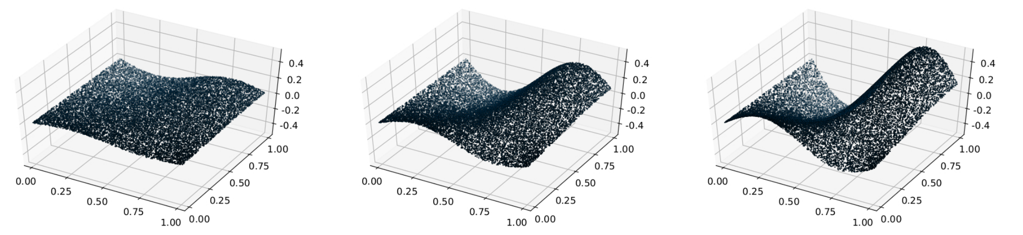


Figure 5.8: **Manifold data.** A two-dimensional point cloud deformed into a two-dimensional submanifold of $\mathbb{R}^3$, with $\delta_z \in \{0.1, 0.3, 0.5\}$.

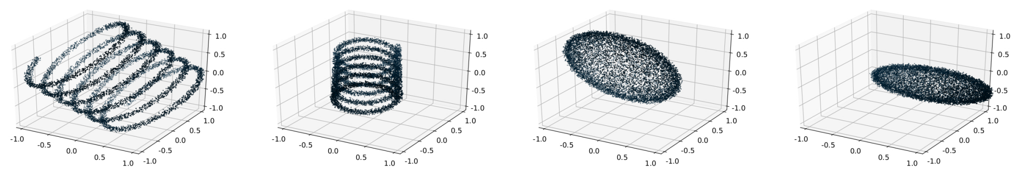


Figure 5.9: **A high-dimensional example.** We construct a high-dimensional dataset with low-dimensional structure by rotating the above structures at random into a 20-dimensional ambient space.

Table 5.8: $G_{\nu,l}^{\text{Matérn}}$ for $\nu = 0.5$, $l = 0.5$, and $N = 10^6$ points as in Figure 5.9.

	$\mathrm{nnz}(L)/N^2$	$\mathrm{rank}(L)$	$t_{\texttt{SortSparse}}$	$t_{\texttt{Entries}}$	$t_{\texttt{ICHOL(0)}}$	E
$\rho = 2.0$	1.62e-04	997635	80.60	57.11	52.49	1.57e-02 (1.13e-03)
$\rho = 3.0$	3.76e-04	1000000	173.86	135.61	248.78	2.88e-03 (1.14e-05)
$\rho = 4.0$	6.76e-04	1000000	302.98	247.74	748.62	8.80e-04 (4.97e-06)
$\rho = 5.0$	1.05e-03	1000000	462.98	397.42	1802.44	3.44e-04 (2.54e-06)
$\rho = 6.0$	1.49e-03	1000000	645.56	556.72	3696.31	1.44e-04 (8.76e-07)
$\rho = 7.0$	2.02e-03	1000000	891.08	758.88	6855.23	7.61e-05 (5.66e-07)
$\rho = 8.0$	2.62e-03	1000000	1248.90	990.86	11598.66	4.57e-05 (4.36e-07)

the exponential decay of the covariance function itself. Chapter 5.6 gives the results for $(l, \alpha, \beta) = (0.4, 0.5, 0.025)$ and $(l, \alpha, \beta) = (0.2, 1.0, 0.2)$.

In Gaussian process regression, the ambient dimension d is typically too large to ensure the computational efficiency of our algorithm. However, since our algorithm only requires access to pairwise distances between points, it can take advantage of the possibly lower intrinsic dimension of the dataset. We might be concerned that in this case, interaction through the higher dimensional ambient space will disable the screening effect. As a first demonstration that this is not the case, we will draw $N = 10^6$ points in $[0,1]^2$ and equip them with a third component according to $x_i^{(3)} := -\delta_z \sin(6x_i^{(1)}) \cos(2(1 - x_i^{(2)})) + \xi_i 10^{-3}$, for ξ_i i.i.d. standard Gaussian. Figure 5.8 shows the resulting point sets for different values of δ_z, and Table 5.7 shows that the approximation is robust to increasing values of δ_z.

An appealing feature of our method is that it can be formulated in terms of the pairwise distances alone. This means that the algorithm will automatically exploit any low-dimensional structure in the dataset. In order to illustrate this feature, we artificially construct a dataset with low-dimensional structure by randomly rotating four low-dimensional structures into a 20-dimensional ambient space (see Figure 5.9). Chapter 5.8 shows that the resulting approximation is even better than the one obtained in dimension 3, illustrating that our algorithm did indeed exploit the low intrinsic dimension of the dataset.

5.6 Numerical example: Preconditioning finite element matrices

5.6.1 Overview

In this section, we describe applications of the methods in Section 5.3 to the stiffness matrices arising from finite element discretizations of elliptic partial differential equations. We interpret the stiffness matrix as $A = \Theta^{-1}$, reorder the degrees of freedom from fine to coarse analog to Example 1, and apply `ICHOL(0)`. We consider both a classical Poisson problem

$$\begin{cases} -\nabla \cdot (a(x)\nabla u(x)) = g(x), \\ u(x) = 0 \ \ \forall x \in \partial\Omega \end{cases} \tag{5.44}$$

and a linear elasticity problem

$$\begin{cases} \mu(x)\Delta u(x) + \dfrac{\mu(x)}{1-2\nu}\nabla(\nabla \cdot u(x)) = g(x), \\ u(x) = 0 \ \ \forall x \in \mathcal{B}. \end{cases} \tag{5.45}$$

putting particular emphasis on settings where the ill-conditioning of the differential operator arises not only from the unboundedness of the differential operators ∇ and Δ, but also from the large variability in magnitudes of the conductivity a and Young's modulus μ. On top of this *high contrast*, the coefficient fields are also *rough*, meaning that they are not expected to have any degree of smoothness or even continuity.

This violates the setting of our theoretical results in three ways:

1. Rather than with the inverse of a Galerkin discretization of $\mathcal{L}^{-1}$, we are working with a discretization of $\mathcal{L}$ itself.
2. While our theoretical results fully cover rough coefficients, the constants in our estimates depend on $\mathcal{L}$ and A, and are therefore expected to deteriorate in the presence of high contrast.

ρ	2.0	2.5	3.0	3.5	4.0
#iter averaging	184	135	85	80	67
#iter subsampling	433	278	134	117	72

Table 5.9: **Averaging vs Subsampling.** For homogeneous Poisson problems and small values of ρ, averaging can improve the preconditioning effect compared to subsampling.

3. We use a reordering akin to Example 1 as opposed to an averaging averaging scheme as in Example 2 even though the Laplace operator ($s = 1$) with spatial dimension $d \in \{2, 3\}$ violates the condition $s > d/2$ of Example 1.

Regarding the first point, analog results could be derived by repeating the proofs of Chapter 4 in the fully discrete setting. Regarding the second point, in agreement with earlier numerical results by [188], we observe that the impact of unstructured high-contrast coefficients on our method is far less severe than could be expected based on the theoretical results in Chapter 4. We suspect that the impact of the contrast on the accuracy of our method is highly dependent on the geometric structure of the coefficients. While there should be adversarial choices of geometry with devastating effects, unstructured coefficient fields seem to be surprisingly benign. Regarding the third point, while the numerical homogenization results of Theorem 12 break down if the condition $s > d/2$ is violated, the exponential decay result does seem to hold. For Poisson problems with slowly varying, low-contrast coefficients and small values of ρ, we obseve that using an averaging scheme improves the preconditioning effect, as shown in Table 5.9.

On the other hand, in the presence of high contrast coefficients reordering according to Example 1 tends to perform better than averaging according to Example 2 since the resulting incomplete Cholesky factorization is less prone to encountering non-positive pivots. We have no theoretical explanation for this empirical observation.

5.6.2 Accuracy and computational complexity

We begin by investigating empirically the sparsity of the Cholesky factors and the scaling of the computational complexity of our factorization. In Figure 5.10, we show the sparsity pattern for different ρ together with the approximation error. We next investigate the computational time of computing the Cholesky factorization and how it scales with N. As shown in the log-log plot in Figure 5.11, for fixed ρ, our factorization achieves the linear scaling predicted by the theory.

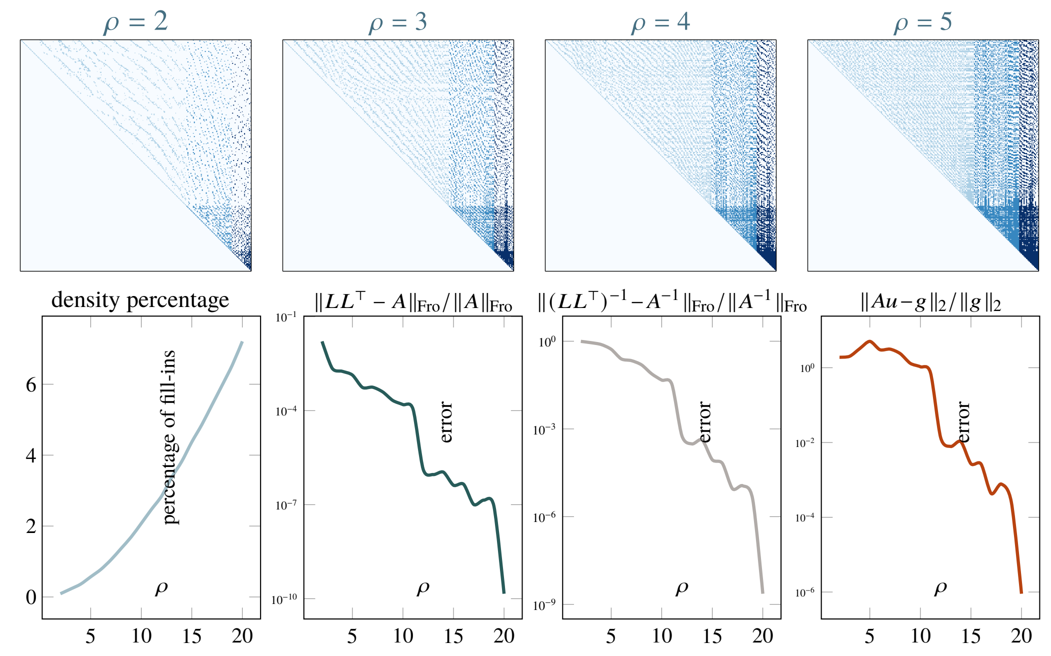


Figure 5.10: **Sparsity pattern, error and factor density**. When computing the factorization of a Laplacian on a $16k \times 16k$ grid in the reverse maximin ordering of Definition 6, the Cholesky factors are approximately sparse according to the reverse maximin sparsity pattern of Definition 7, despite the condition $s > d/2$ of Example 1 not being satisfied.

Exact sparse Cholesky factorization, as discussed in Section 3.2, is a strong competitor for many practical two-dimensional problems. However, its superlinear scaling of both time and space complexity limits its applicability to three-dimensional problems. In Figure 5.12, we compare our method against the highly optimized CHOLMOD library on series of three-dimensional problems of increasing size. While CHOLMOD is superior for small problems, our method shows superior performance on problems with $N > 10^6$. We note that CHOLMOD has the advantage that it is not impacted by the geometric structure and magnitude of the coefficients and would therefore preserve its computational cost if we were to choose a more challenging problem of the same size. Nevertheless, it becomes infeasible for truly large-scale problems, while our method preserves its near-linear scaling.

5.6.3 Comparison with algebraic multigrid methods

A popular choice for solving elliptic partial differential equations are algebraic multigrid method [41, 42, 253]. Where geometric multigrid methods [81, 111, 113] use predetermined multiresolution schemes that are vulnerable to rough and high contrast coefficients [20], algebraic multigrid methods use an operator-adapted

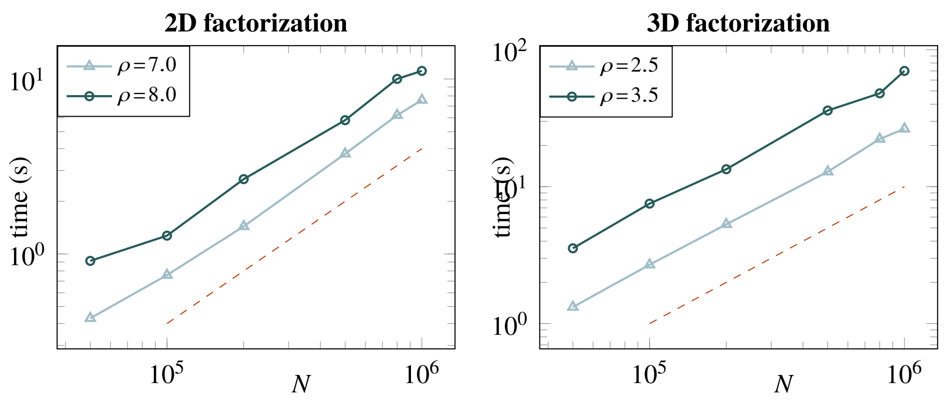


Figure 5.11: **Scalability.** In 2D and 3D, our IC factorization time matches the expected $O(N\rho^{2d})$ time complexity for a matrix size $N \times N$ and a sparsity parameter ρ; the dashed line indicates a slope of 1 in this log-log plot.

multiresolution analysis. While this improves their performance on many problems, rough coefficients with high contrast still pose challenges [7, 255].

We compare our method to the popular AMG implementations AMGCL [67] and Trilinos [237]. For AMGCL, we chose the Ruge-Stüben method [230] to construct the hierarchy and sparse inverse approximation [102] for relaxation. For Trilinos, we use smoothed aggregation [240] for coarsening and symmetric Gauss-Seidel for relaxation.

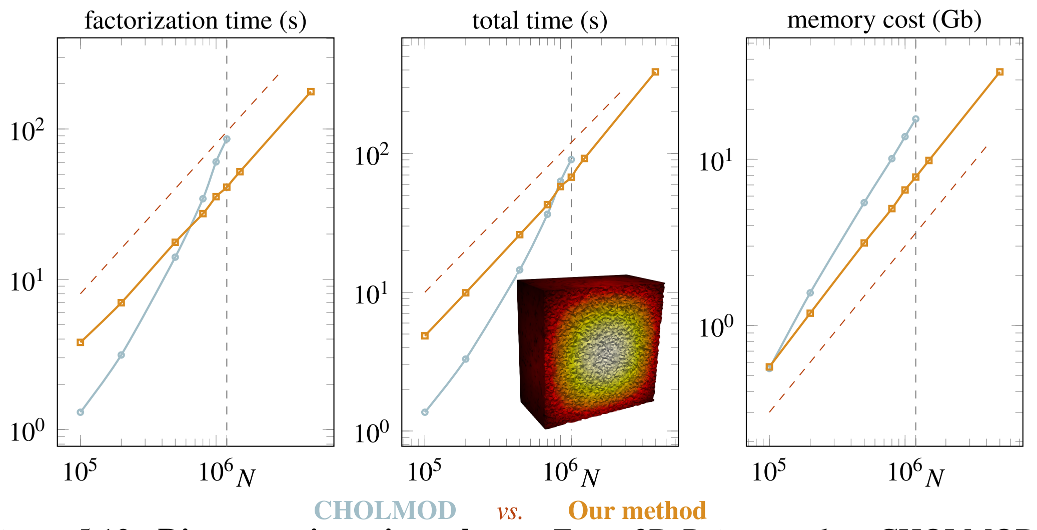
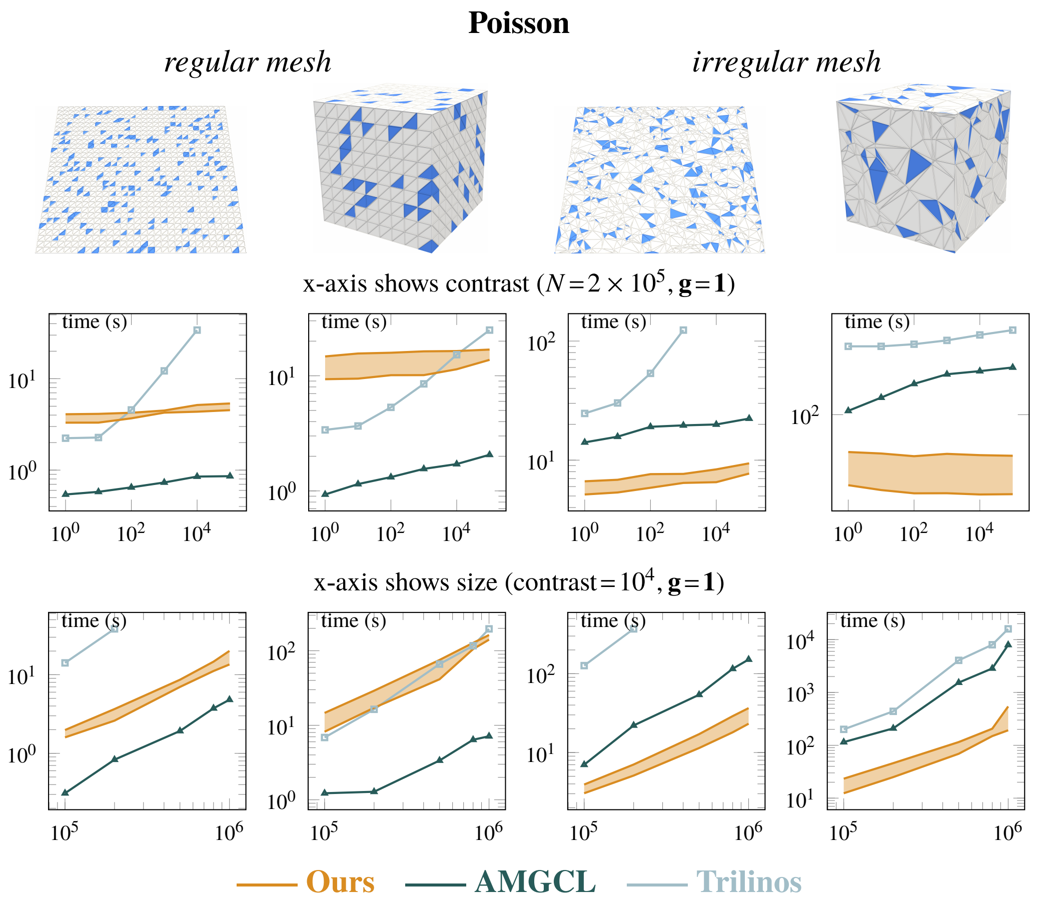


Figure 5.12: **Direct vs. iterative solvers.** For a 3D Poisson solve, CHOLMOD scales non-linearly in the linear system size N for factorization time, total solve time (which include factorization and back-substitution), and memory use, and fails for $N > 1$M; instead, a PCG-based iterative solve using our preconditioner exhibits consistent linear behaviors on all three measurements.

We begin our comparison on the Poisson problem (5.44) and the elasticity problem (5.45) in two and three dimensions on a bimaterial generated by randomly assigned stiff regions. The stiff regions form about $1/8$ of the problem in two dimensions and $1/64$ of the problem in three dimensions. We refer to the ratio of coefficient magnitudes in *stiff* and normal regions as the contrast. We furthermore investigate examples on both regular and (uniformly) randomly generated grids.



As we can see in Figure 5.12 our method beats both of these methods in all problems with the exception of Poisson problems on regular grids, where it is outperformed by AMGCL.

In many applications, our method will be used repeatedly during the solution of a nonlinear problem via implicit time-stepping or Newton's method. In Fiture 5.13, we use our method to accelerate computations in quasi-statics: We stretch an armadillo model with about $340k$ nodes (thus over $1M$ degrees of freedom), for which each tetrahedron is randomly assigned one of two neo-Hookean materials whose contrast between Young's moduli is 10^4. In each step of our Newton descent to find the final shape, we project the neo-Hookean elastic stiffness matrix to enforce semi-positive

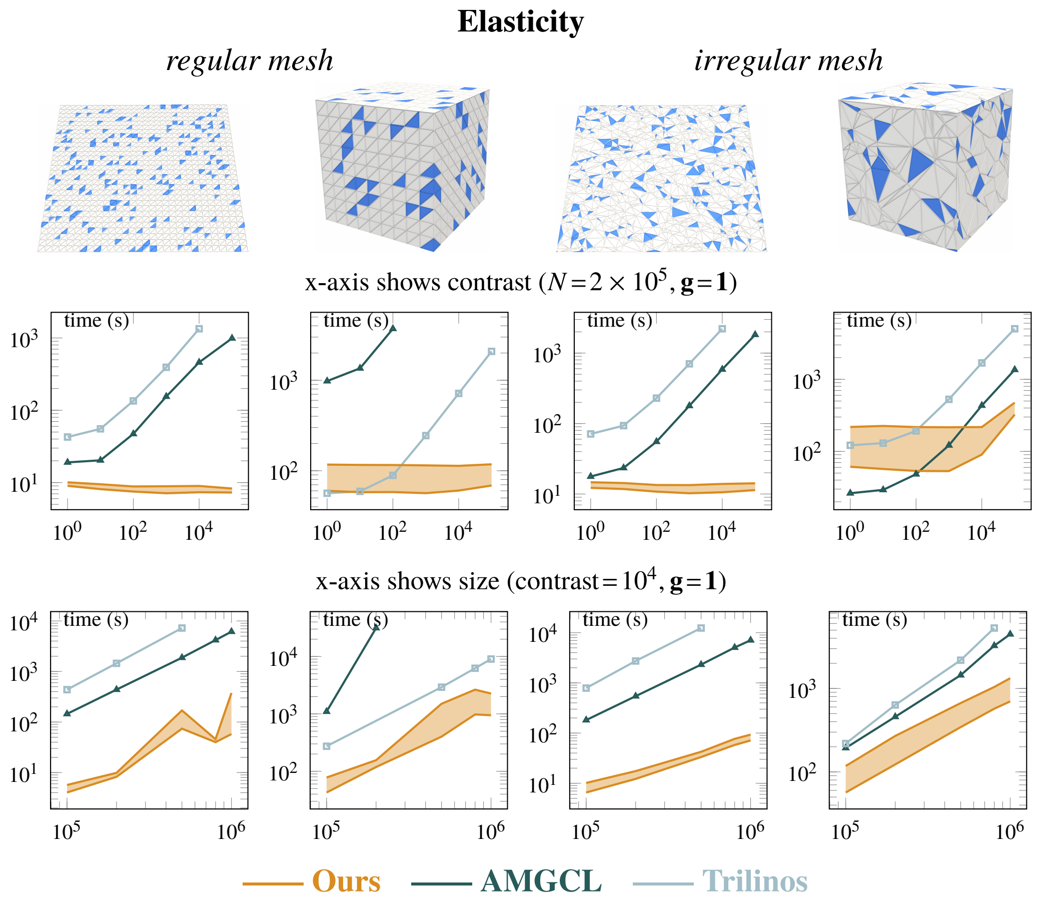


Figure 5.12: **Comparisons with AMG libraries.**. Figures indicate time costs (including factorization and PCG iteration times) as a function of material contrast. Our method is much less sensitive to contrast and problem size, and is particularly efficient when the size N becomes large and/or for bad condition numbers. For our method, timings are within the orange region depending on the actual value of ρ, for which we used the range $[6.5, 8.5]$ in 2D and $[2.5, 4.0]$ in 3D. All meshes are generated by Delaunay triangulation.

definiteness [234]. The linear solves of the Newton descent exhibit a 2.5× speedup on average compared to AMGCL, with larger speed-ups for smaller error tolerances.

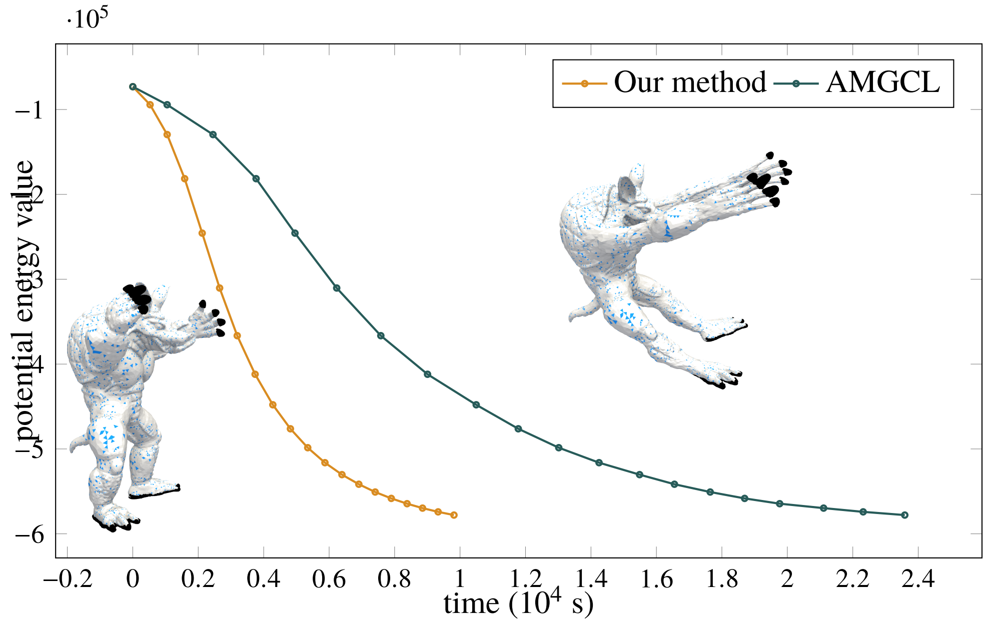


Figure 5.13: **Nonlinear quasi-statics**. An armadillo is stretched via lateral gravity with a few nodes (marked in black) fixed at their initial position. We use a trust region nonlinear optimization algorithm involving the solution of a linear system using our IC preconditioner or AMGCL at each step; timing of the first 20 iterations are plotted.

Chapter 6

CHOLESKY FACTORIZATION BY KULLBACK-LEIBLER MINIMIZATION

6.1 Overview

6.1.1 Factorizing Θ^{-1} from entries of Θ

In this chapter, we propose to compute a sparse approximate inverse Cholesky factor L of Θ, by minimizing with respect to L and subject to a sparsity constraint, the Kullback-Leibler (KL) divergence between two centered multivariate normal distributions with covariance matrices Θ and $(LL^{\top})^{-1}$. Surprisingly, this minimization problem has a closed-form solution, enabling the efficient computation of optimally accurate Cholesky factors for any specified sparsity pattern.

The resulting approximation can be shown to be equivalent to the Vecchia approximation of Gaussian processes [241], which has become very popular for the analysis of geospatial data (e.g., [62, 104, 135, 136, 226, 233]); to the best of our knowledge, rigorous convergence rates and error bounds were previously unavailable for Vecchia approximations, and this work is the first one presenting such results. An equivalent approximation has also been proposed by [133] and [141] in the literature on factorized sparse approximate inverse (FSAI) preconditioners of (typically) sparse matrices (see e.g., [33] for a review and comparison, [51] for an application to dense kernel matrices); however, its KL-divergence optimality has not been observed before. KL-minimization has also been used to obtain sparse lower-triangular transport maps by [166]; while this literature is mostly concerned with the efficient sampling of non-Gaussian probability measures, the present work shows that an analogous approach can be used to obtain fast algorithms for numerical linear algebra if the sparsity pattern is chosen appropriately.

6.1.2 State-of-the-art computational complexity

The computational complexity and approximation accuracy of our approach depend on the choice of elimination ordering and sparsity pattern. When Θ is the covariance function of a Gaussian process that is subject to the screening effect, we propose to use the reverse maximin ordering and sparsity pattern proposed in Definitions 6 and 7. By using a grouping algorithm similar to the supernodes of Chapter 5 and the heuristics proposed by [83, 104, 226], we can show that the approximate inverse

Cholesky factor can be computed in computational complexity $\mathcal{O}(N\rho^{2d})$ in time and $\mathcal{O}(N\rho^{d})$ in space, using only $\mathcal{O}(N\rho^{d})$ entries of the original kernel matrix Θ, where ρ is a tuning parameter trading accuracy for computational efficiency.

In settings where Theorem 10 on the exponential decay of Θ^{-1} holds, it allows us to prove that the approximation error decays exponentially in ρ. In this setting, we can thus compute an ϵ-approximation of Θ in complexity $\mathcal{O}\left(N \log^{2d}(N/\epsilon)\right)$ by choosing $\rho \approx \log(N/\epsilon)$. This is the best known trade-off between computational complexity and accuracy for inverses of Green's matrices of general elliptic boundary value problems.

6.1.3 Practical advantages

Our method has important *practical* advantages complementing its theoretical and asymptotic properties. In many GP regression applications, large values of ρ are computationally intractable with present-day resources. By incorporating prediction points in the computation of KL-optimal inverse-Cholesky factors, we obtain a GP regression algorithm that is accurate even for small (≈ 3) values of ρ, including in settings where truncation of the *true* Cholesky factor of Θ^{-1} to the same sparsity pattern fails completely.

For other hierarchy-based methods, the computational complexity depends exponentially on the dimension d of the dataset. In contrast, because the construction of the ordering and sparsity pattern only uses pairwise distances between points, our algorithms automatically adapt to low-dimensional structure in the data and operate in complexities identified by replacing d with the *intrinsic dimension* $\tilde{d} \leq d$ of the dataset.

We have seen in section 5.5.3 that the screening deteriorates for independent sums of two GPs, such as when combining a GP with additive Gaussian white noise. As noted there, this problem could be overcome if we had access to the *inverse* of the covariance matrices. Analogs of the Cholesky factorization that compute this inverse by recursive Schur complementation exist, but they are too unstable to be useful in practice. The inherent stability of the KL minimization allows us to realize this approach and thus compute both cheaply and accurately with sums of independent Gaussian processes arising, for instance, from modeling measurement noise. To the best of our knowledge, this is the first time this has been achieved by a method based on the screening effect.

Finally, our algorithm is intrinsically parallel because it allows each column of the sparse factor to be computed independently (as in the setting of the Vecchia approximation, factorized sparse approximate inverses, and lower-triangular transport maps). Furthermore, we show that in the context of GP regression, the log-likelihood, the posterior mean, and the posterior variance can be computed in $O(N+\rho^d)$ space complexity. In a parallel setting, we require $O(\rho^d)$ communication between the different workers for every $O(\rho^{3d})$ floating-point operations, resulting in a total communication complexity of $O(N)$. Here, most of the floating-point operations arise from calls to highly optimized BLAS and LAPACK routines.

6.2 Cholesky factorization by KL-minimization

The Kullback-Leibler divergence between two probability measures P and Q is defined as $\mathbb{D}_{\mathrm{KL}}(P \parallel Q) = \int \log(\mathrm{d}P/\mathrm{d}Q)\,\mathrm{d}P$. If Q is an approximation of P, then the KL divergence is the expected difference between the associated true and approximate log-densities, and so its minimization is directly relevant for accurate approximations of GP inference, including GP prediction and likelihood-based inference on hyperparameters. By virtue of its connection to the likelihood ratio test [71], the KL divergence can also be interpreted as the strength of the evidence that samples from P were not instead obtained from Q. If P and Q are both N-variate centered normal distributions, the KL divergence is equivalent to a popular loss function for covariance-matrix estimation [129], and it can be written as

$$2\,\mathbb{D}_{\mathrm{KL}}(\mathcal{N}(0,\Theta_1) \parallel \mathcal{N}(0,\Theta_2)) = \operatorname{trace}(\Theta_2^{-1}\Theta_1) + \operatorname{logdet}(\Theta_2) - \operatorname{logdet}(\Theta_1) - N. \tag{6.1}$$

Let Θ be a positive-definite matrix of size $N \times N$. Given a lower-triangular sparsity set $S \subset I \times I$, where $I = \{1,\dots,N\}$, we want to use

$$L := \operatorname*{arg\,min}_{\hat{L}\in\mathcal{S}} \mathbb{D}_{\mathrm{KL}}\Big(\mathcal{N}(0,\Theta) \parallel \mathcal{N}(0,(\hat{L}\hat{L}^\top)^{-1})\Big) \tag{6.2}$$

as approximate Cholesky factor for Θ^{-1}, for $\mathcal{S} := \left\{A \in \mathbb{R}^{N\times N} : A_{ij} \neq 0 \Rightarrow (i,j) \in S\right\}$. While solving the non-quadratic program (6.2) might seem challenging, it turns out that it has a closed-form solution that can be computed efficiently:

Theorem 20. *The nonzero entries of the i-th column of L as defined in Equation* (6.2) *are given by*

$$L_{s_i,i} = \frac{\Theta^{-1}_{s_i,s_i}\mathbf{e}_1}{\sqrt{\mathbf{e}_1^\top \Theta^{-1}_{s_i,s_i}\mathbf{e}_1}}, \tag{6.3}$$

where $s_i := \{j : (i,j) \in S\}$, $\Theta^{-1}_{s_i,s_i} := (\Theta_{s_i,s_i})^{-1}$, Θ_{s_i,s_i} *is the restriction of* Θ *to the set of indices* s_i, *as illustrated in Figure* 6.1 *and* $\mathbf{e}_1 \in \mathbb{R}^{\#s_i\times 1}$ *is the vector with the*

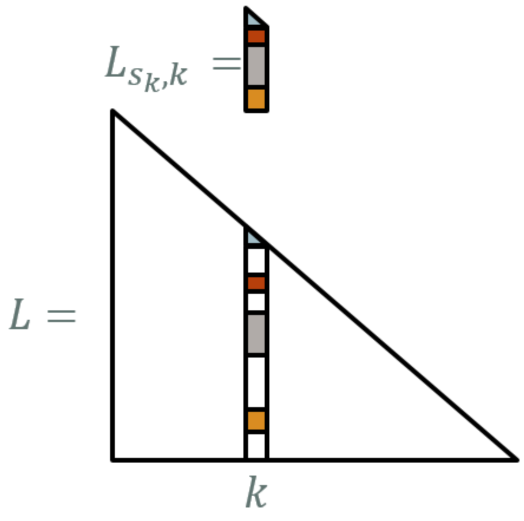


Figure 6.1: **The nonzero-vector of a sparse column.** The set s_k is the vector of row-indices contained in the k-th column of the sparsity pattern. The vector $L_{s_k,s}$ denotes the vector of nonzero entries of L with these row indices.

first entry equal to one and all other entries equal to zero. Using this formula, L can be computed in computational complexity $O\left(\#S + (\max_{1 \le i \le N} \#s_i)^2\right)$ *in space and* $O\left(\sum_{i=1}^{N} (\#s_i)^3 \right)$ *in time.*

Proof. By using the formula for the KL-divergence of two Gaussian random variables in (6.1), we obtain

$$L = \underset{\hat{L} \in \mathcal{S}}{\arg\min} \left(\operatorname{trace}(\hat{L}\hat{L}^\top \Theta) - \operatorname{logdet}(\hat{L}\hat{L}^\top) - \operatorname{logdet}(\Theta) - N\right) \tag{6.4}$$

$$= \underset{\hat{L} \in \mathcal{S}}{\arg\min} \left(\operatorname{trace}(\hat{L}^\top \Theta \hat{L}) - \operatorname{logdet}(\hat{L}\hat{L}^\top)\right) \tag{6.5}$$

$$= \underset{\hat{L} \in \mathcal{S}}{\arg\min} \sum_{k=1}^{N} \left(\hat{L}_{s_k,k}^\top \Theta_{s_k,s_k} \hat{L}_{s_k,k} - 2\log(\hat{L}_{k,k}) \right). \tag{6.6}$$

The k-th summand depends only on the k-th column of $\hat{L}$. Thus, taking the derivative with respect to the k-the column of L and setting it to zero, we obtain $\Theta_{s_k,s_k} \hat{L}_{s_k,k} = \frac{\mathbf{e}_1}{\hat{L}_{k,k}} \Leftrightarrow \hat{L}_{s_k,k} = \frac{\Theta_{s_k,s_k}^{-1} \mathbf{e}_1}{\hat{L}_{k,k}}$. Therefore, $\hat{L}_{s_k,k}$ can be written as $\lambda \Theta_{s_k,s_k}^{-1} \mathbf{e}_1$ for a $\lambda \in \mathbb{R}$. By plugging this ansatz into the equation, we obtain $\lambda = \sqrt{\left(\Theta_{s_k,s_k}^{-1} \mathbf{e}_1\right)_1} = \sqrt{\mathbf{e}_1^\top \Theta_{s_k,s_k}^{-1} \mathbf{e}_1}$ and hence Equation (6.3). By using dense Cholesky factorization to invert the Θ_{s_k,s_k}, the right-hand side of Equation (6.3) can be computed in computational complexity $O\left(\# (s_k)^2\right)$ in space and $O\left(\# (s_k)^3\right)$ in time, from which follows the result. □

Compared to ordinary sparse Cholesky factorization (see Algorithm 3), the algorithm implied by Theorem 20 has the advantage of giving the *best* possible Cholesky factor (as measured by KL) for a given sparsity pattern. Furthermore, it is embarrassingly parallel — all evaluations of Equation (6.3) can be performed independently for different i. While the computational complexity is slightly worse than the one of in-place incomplete Cholesky factorization, we will show in Theorem 21 that for important choices of S, the time complexity can be reduced to $O\left(\sum_{k=1}^{N} (\# s_k)^2 \right)$, matching the computational complexity of incomplete Cholesky factorization.

The formula in Equation (6.3) can be shown to be equivalent to the formula that has been used to compute the Vecchia approximation [241] in spatial statistics, without explicit awareness of the KL-optimality of the resulting L. In the literature on factorized sparse approximate inverses, the above formula was derived for minimizers of $\|\mathrm{Id} - L \operatorname{chol}(\Theta)\|_{\mathrm{Fro}}$ subject to the constraints $L \in \mathcal{S}$ and $\operatorname{diag}(L\Theta L^\top) = 1$ [141], and for minimizers of the Kaporin condition number $(\operatorname{trace}(\Theta L L^\top)/N)^N / \det(\Theta(LL^\top))$ subject to the constraint $L \in \mathcal{S}$ [133]. The KL-divergence, as opposed to $\|\mathrm{Id} - L \operatorname{chol}(\Theta)\|_{\mathrm{Fro}}$, strongly penalizes zero eigenvalues of $\Theta L L^\top$, which explains the observation of [75] that adding the constraint $\operatorname{diag}(L\Theta L^\top) = 1$ tends to improve the spectral condition number of the resulting preconditioner, despite increasing the size of the fidelity term $\|\mathrm{Id} - L \operatorname{chol}(\Theta)\|_{\mathrm{Fro}}$. [166] showed that the embarrassingly parallel nature of KL-minimization is even preserved when replacing the Cholesky factors with nonlinear transport maps with Knothe-Rosenblatt structure. As part of work on the sample complexity of the estimation of transport maps, [27] discovered representations very similar to Equation (6.3), independently of the present work.

We propose the following procedure to approximate a positive-definite matrix Θ:

1. Order the degrees of freedom (i.e., rows and columns of Θ) according to some ordering $\prec$.

2. Pick a sparsity set $S \subset I \times I$.

3. Use Formula (6.3) to compute the lower-triangular matrix L with nonzero entries contained in S that minimizes $\mathbb{D}_{\mathrm{KL}}\left(\mathcal{N}(0, \Theta) \;\|\; \mathcal{N}(0, (LL^\top)^{-1})\right)$.

In the next section, we will describe how to implement all three steps of this procedure in the more concrete setting of positive-definite matrices obtained from the evaluation of a finitely smooth covariance function at pairs of points in $\mathbb{R}^d$.

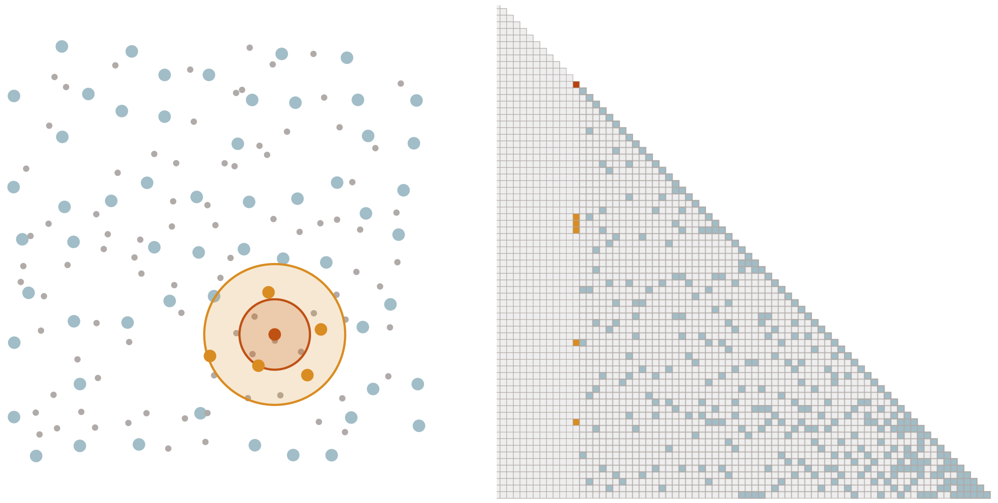


Figure 6.2: **The reverse maximin ordering.** To obtain the reverse maximin ordering, for $k = N-1, N-2, \ldots, 1$, we successively select the point x_{i_k} that has the largest distance ℓ_{i_k} to those points $x_{i_{k+1}}, \ldots, x_{i_N}$ selected previously (shown as enlarged). All previously selected points within distance $\rho \ell_i$ of x_{i_k} (here, $\rho = 2$) form the k-th column of the sparsity pattern.

6.3 Ordering and sparsity pattern motivated by the screening effect

The quality of the approximation given by Equation (6.2) depends on the ordering of the variables and the sparsity pattern S. For kernel matrices satisfying the screening effect, we propose to use reverse maximin ordering and sparsity pattern introduced in Chapter 3. For the convenience of the reader, we briefly recapitulate its definition and provide some notation that will be useful later on.

6.3.1 The reverse maximin ordering and sparsity pattern

Assume that $\mathcal{G}$ is the covariance function of a Gaussian process that is conditioned to be zero on (the possibly empty set) $\partial\Omega$, and the kernel matrix $\Theta \in \mathbb{R}^{I \times I}$ is obtained as $\Theta_{ij} := \mathcal{G}(x_i, x_j)$ for a set of locations $\{x_i\}_{i \in I} \subset \Omega$.

The *reverse maximum-minimum distance (reverse maximin) ordering* of $\{x_i\}_{i \in I}$ is achieved by selecting the last index as

$$i_N := \underset{i \in I}{\arg\max}\, \operatorname{dist}(x_i, \partial\Omega) \tag{6.7}$$

(or arbitrarily for $\partial\Omega = \emptyset$), and then choosing sequentially for $k = N-1, N-2, \ldots, 1$ the index that is furthest away from $\partial\Omega$ and those indices that were already picked:

$$i_k := \underset{i \in I \setminus \{i_{k+1}, \ldots, i_N\}}{\arg\max} \operatorname{dist}\left(x_i, \left\{x_{i_{k+1}}, \ldots, x_{i_N}\right\} \cup \partial\Omega\right). \tag{6.8}$$

Algorithm 12 Without aggregation

```
Input: $\mathcal{G}$, $\{x_i\}_{i \in I}$, $\prec$, $S_{\prec,l,\rho}$
Output: $L \in \mathbb{R}^{N \times N}$ l. triang. in $\prec$

1: for $k \in I$ do
2:     for $i, j \in s_k$ do
3:         $\left(\Theta_{s_k,s_k}\right)_{ij} \leftarrow \mathcal{G}(x_i, x_j)$
4:     end for
5:     $L_{s_k,k} \leftarrow \Theta_{s_k,s_k}^{-1}\mathbf{e}_k$
6:     $L_{s_k,k} \leftarrow L_{s_k,k}/\sqrt{L_{k,k}}$
7: end for
8: return $L$
```

Algorithm 13 With aggregation

```
Input: $\mathcal{G}$, $\{x_i\}_{i \in I}$, $\prec$, $S_{\prec,l,\rho,\lambda}$
output: $L \in \mathbb{R}^{N \times N}$ l. triang. in $\prec$

1: for $\tilde{k} \in \tilde{I}$ do
2:     for $i, j \in s_{\tilde{k}}$ do
3:         $\left(\Theta_{s_{\tilde{k}},s_{\tilde{k}}}\right)_{ij} \leftarrow \mathcal{G}(x_i, x_j)$
4:     end for
5:     $U \leftarrow P^{\updownarrow}\operatorname{chol}(P^{\updownarrow}\Theta_{s_{\tilde{k}},s_{\tilde{k}}}P^{\updownarrow})P^{\updownarrow}$
6:     for $k \rightsquigarrow \tilde{k}$ do
7:         $L_{s_k,k} \leftarrow U^{-\top}\mathbf{e}_k$
8:     end for
9: end for
10: return $L$
```

Figure 6.3: **KL-minimizing Cholesky factorization.** KL-minimization with and without using aggregation. For notational convenience, all matrices are assumed to have row and column ordering according to $\prec$. $P^{\updownarrow}$ denotes the order-reversing permutation matrix, and $\mathbf{e}_k$ is the vector with 1 in the k-th component and zero elsewhere.

Write $\ell_{i_k} = \operatorname{dist}\left(x_{i_k}, \left\{x_{i_{k+1}}, \ldots, x_{i_N}\right\} \cup \partial\Omega\right)$, and write $i \prec j$ if i precedes j in the reverse maximin ordering. We collect the $\{\ell_i\}_{i \in I}$ into a vector denoted by ℓ.

For a tuning parameter $\rho \in \mathbb{R}^{+}$, we select the sparsity set $S_{\prec,\ell,\rho} \subset I \times I$ as

$$S_{\prec,\ell,\rho} := \left\{(i,j) : i \succeq j, \operatorname{dist}(x_i, x_j) \leq \rho\ell_j\right\}. \tag{6.9}$$

The reverse maximin ordering and sparsity pattern is illustrated in Figure 6.2.

We can use a minor modification of Algorithm 11 to construct the reverse maximin ordering and sparsity pattern in computational complexity $O(N\log^2(N)\rho^{\tilde{d}})$ in time and $O(N\rho^{\tilde{d}})$ in space, where $\tilde{d} \leq d$ is the intrinsic dimension of the dataset, as will be defined in Condition 5. The inverse Cholesky factors L can then be computed using Equation (6.3), as in Algorithm 12.

6.3.2 Aggregated sparsity pattern

It was already observed by [83] in the context of sparse approximate inverses, and by [104, 226] in the context of the Vecchia approximation that a suitable grouping of the degrees of freedom makes it possible to *reuse* Cholesky factorizations of the

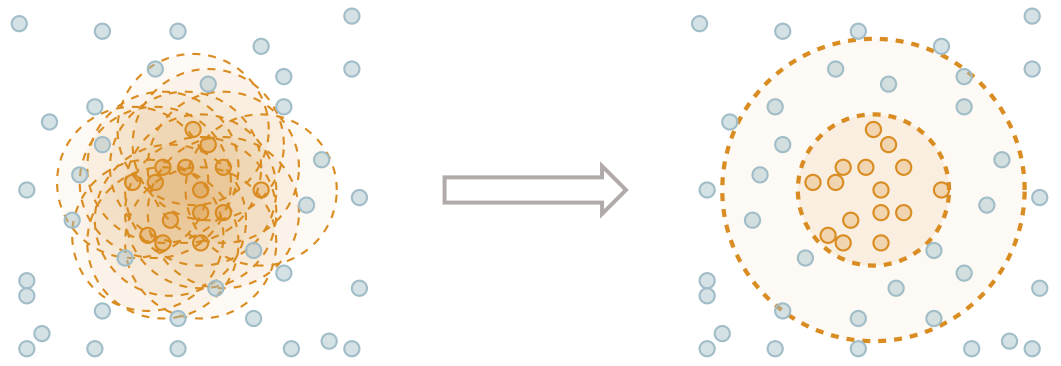


Figure 6.4: **Geometric aggregation.** The left figure illustrates the original pattern $S_{\prec,\ell,\rho}$. For each orange point, we need to keep track of its interactions with all points within a circle of radius $\approx \rho$. In the right figure, the points have been collected into a supernode, which can be represented by a list of *parents* (the orange points within an inner sphere of radius $\approx \rho$) and *children* (all points within a radius $\approx 2\rho$).

matrices Θ_{s_i,s_i} in Equation (6.3) to update multiple columns at once. The authors of [83, 104] propose grouping heuristics based on the sparsity graph of L and show empirically that they lead to improved performance. In contrast, we propose a grouping procedure based on geometric information and prove that it allows us to reach the best asymptotic complexity in the literature in a more concrete setting.

Assume that we have already computed the reverse maximin ordering $\prec$ and sparsity pattern $S_{\prec,\ell,\rho}$, and that we have access to the ℓ_i as defined above. We will now aggregate the points into groups called *supernodes*, consisting of points that are close in both location and ordering. To do so, we pick at each step the first (w.r.t. $\prec$) index $i \in I$ that has not been aggregated into a supernode yet and then we aggregate into a common supernode the indices in $\{j : (i,j) \in S_{\prec,\ell,\rho}, \ell_j \leq \lambda \ell_i\}$ for some $\lambda > 1$ ($\lambda \approx 1.5$ is typically a good choice) that have not been aggregated yet. We proceed with this procedure until every node has been aggregated into a supernode. We write $\tilde{I}$ for the set of all supernodes; for $i \in I, \tilde{i} \in \tilde{I}$, we write $i \rightsquigarrow \tilde{i}$ if $\tilde{i}$ is the supernode to which i has been aggregated. We furthermore define $s_{\tilde{i}} := \left\{j : \exists i \rightsquigarrow \tilde{i}, j \in s_i\right\}$ and introduce the aggregated sparsity pattern $\tilde{S}_{\prec,\ell,\rho,\lambda} := \bigcup_{k \rightsquigarrow \tilde{k}} \left\{(i,k) : k \preceq i \in s_{\tilde{k}}\right\}$. This sparsity pattern, while larger than $S_{\prec,\ell,\rho}$, can be represented efficiently by keeping track of the set of *parents* (the $k \in I$ such that $k \rightsquigarrow s_{\tilde{k}}$) and *children* (the $i \in s_{\tilde{k}}$) of each supernode, rather than the individual entries (see Figure 6.4 for an illustration). For well-behaved (cf. Theorem 21) sets of points, we obtain $O(N\rho^{-d})$ supernodes, each with $O(\rho^d)$ parents and children, thus improving the cost of storing the sparsity pattern from $O(N\rho^d)$ to $O(N)$.

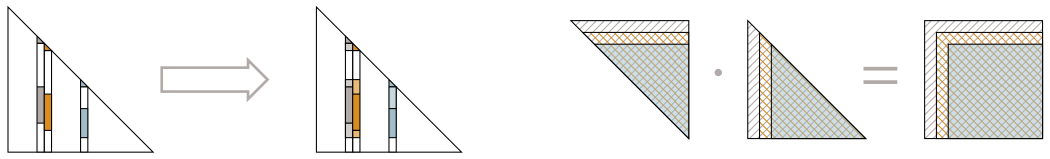


Figure 6.5: **Reusing Cholesky factors.** (Left:) By adding a few nonzero entries to the sparsity pattern, the sparsity patterns of columns in $s_{\tilde{k}}$ become subsets of one another. (Right:) Therefore, the matrices $\{\Theta_{s_k,s_k}\}_{k \rightsquigarrow \tilde{k}}$, which need to be inverted to compute the columns $L_{:,k}$ for $k \rightsquigarrow \tilde{k}$, become submatrices of one another. Thus, submatrices of the Cholesky factors of $\Theta_{s_{\tilde{k}},s_{\tilde{k}}}$ can be used as factors of Θ_{s_k,s_k} for any $k \rightsquigarrow \tilde{k}$.

While the above aggregation procedure can be performed efficiently once $\prec$ and $S_{\prec,\ell,\rho}$ are computed, it is possible to directly compute $\prec$ and an outer approximation $\bar{S}_{\prec,\ell,\rho,\lambda} \supset \tilde{S}_{\prec,\ell,\rho,\lambda}$ in computational complexity $\mathcal{O}(N)$ in space and $\mathcal{O}(N\log(N))$ in time. $\bar{S}_{\prec,\ell,\rho,\lambda}$ can either be used directly, or it can be used to compute $\tilde{S}_{\prec,\ell,\rho,\lambda}$ in $\mathcal{O}(N)$ in space and $\mathcal{O}(N\log(N)\rho^d)$ in time, using a simple and embarrassingly parallel algorithm. Details are given in Section .3.4.

In addition to reducing the memory cost, the aggregated ordering and sparsity pattern allows us to compute the Cholesky factors (in reverse ordering) $\Theta_{s_{\tilde{k}},s_{\tilde{k}}} = UU^\top$ once for each supernode and then use it to compute the $L_{s_k,k}$ for all $k \rightsquigarrow \tilde{k}$ as in Algoritm 13 (see Figure 6.5 for an illustration).

As we show in the next section, this allows us to reduce the computational complexity from $\mathcal{O}(N\rho^{3d})$ to $\mathcal{O}(N\rho^{2d})$ for sufficiently well-behaved sets of points.

6.3.3 Theoretical guarantees

We now present our rigorous theoretical result bounding the computational complexity and approximation error of our method. Proofs and additional details are deferred to Section .3.2.

Remark 1. *As detailed in Section .3.2, the results below apply to more general* reverse r-maximin *orderings, which can be computed in complexity $\mathcal{O}(N\log(N))$, improving over reverse maximin orderings by a factor of* $\log(N)$.

Computational complexity

We can derive the following bounds on the computational complexity depending on ρ and N.

Theorem 21 (Informal)**.** *Under mild assumptions on $\{x_i\}_{i \in I} \subset \mathbb{R}^d$, the KL-minimizer L is computed in complexity $CN\rho^d$ in space and $CN\rho^{3d}$ in time when using Algorithm 12 with $S_{<,\ell,\rho}$ and in complexity $CN\rho^d$ in space and $C_{\lambda,\ell}CN\rho^{2d}$ in time when using Algorithm 13 with $\tilde{S}_{<,\ell,\rho,\lambda}$. Here, the constant C depends only on d, λ, and the cost of evaluating entries of Θ.*

A more formal statement and a proof of Theorem 21 can be found in Section .3.2.

As can be seen from Theorem 21, using the aggregation scheme decreases the computational cost by a factor ρ^d. This is because each supernode has $\approx \rho^d$ members that can all be updated by reusing the same Cholesky factorization.

Remark 2. *As described in Section .3.2, the computational complexity only depends on the intrinsic dimension of the dataset (as opposed to the potentially much larger ambient dimension d). This means that the algorithm automatically exploits low-dimensional structure in the data to decrease the computational complexity.*

Approximation error

The rigorous bounds on the approximation error are derived from Theorem 10 and therefore hold under the same assumptions. We assume for the purpose of this section that Ω is a bounded domain of $\mathbb{R}^d$ with Lipschitz boundary, and for an integer $s > d/2$, we write $H_0^s(\Omega)$ for usual Sobolev the space of functions with zero Dirichlet boundary values and order s derivatives in L^2, and $H_0^{-s}(\Omega)$ for its dual. Let the operator

$$\mathcal{L} : H_0^s(\Omega) \mapsto H^{-s}(\Omega), \tag{6.10}$$

be linear, symmetric ($\int u\mathcal{L}v = \int v\mathcal{L}u$), positive ($\int u\mathcal{L}u \geq 0$), bijective, bounded (write $\|\mathcal{L}\| := \sup_u \|\mathcal{L}u\|_{H^{-s}(\Omega)}/\|u\|_{H_0^s(\Omega)}$ for its operator norm), and local in the sense that $\int u\mathcal{L}v \,\mathrm{d}x = 0$, for all $u, v \in H_0^s(\Omega)$ with disjoint support. By the Sobolev embedding theorem, we have $H_0^s(\Omega) \subset C_0(\Omega)$, and hence $\{\delta_x\}_{x\in\Omega} \subset H^{-s}(\Omega)$. We then define $\mathcal{G}$ as the Green's function of $\mathcal{L}$,

$$\mathcal{G}(x_1, x_2) := \int \delta_{x_1}\mathcal{L}^{-1}\delta_{x_2} \,\mathrm{d}x. \tag{6.11}$$

A simple example when $d = 1$ and $\Omega = (0, 1)$, is $\mathcal{L} = -\Delta$, and $\mathcal{G}(x, y) = \mathbb{1}_{x<y}\frac{1-y}{1-x} + \mathbb{1}_{y\leq x}\frac{y}{x}$. Let us define the following measure of *homogeneity* of the distribution of $\{x_i\}_{i\in I}$,

$$\delta := \frac{\min_{x_i, x_j \in I} \operatorname{dist}(x_i, \{x_j\} \cup \partial\Omega)}{\max_{x\in\Omega} \operatorname{dist}(x, \{x_i\}_{i\in I} \cup \partial\Omega)}. \tag{6.12}$$

Using the above definitions, we can rigorously quantify the increase in approximation accuracy as ρ increases.

Theorem 22. *There exists a constant C depending only on d, Ω, λ, s, $\|\mathcal{L}\|$, $\|\mathcal{L}^{-1}\|$, and δ, such that for $\rho \geq C \log(N/\epsilon)$, we have*

$$\mathbb{D}_{\mathrm{KL}}\big(\mathcal{N}(0,\Theta) \,\big\|\, \mathcal{N}(0,\big(L^{\rho}L^{\rho,\top}\big)^{-1})\big) + \big\|\Theta - (L^{\rho}L^{\rho,\top})^{-1}\big\|_{\mathrm{Fro}} \leq \epsilon. \tag{6.13}$$

Thus, Algorithm 12 computes an ϵ-accurate approximation of Θ in computational complexity $CN\log^{d}(N/\epsilon)$ in space and $CN\log^{3d}(N/\epsilon)$ in time, from $CN\log^{d}(N/\epsilon)$ entries of Θ. Similarly, Algorithm 13 computes an ϵ-accurate approximation of Θ in computational complexity $CN\log^{d}(N/\epsilon)$ in space and $CN\log^{2d}(N/\epsilon)$ in time, from $CN\log^{d}(N/\epsilon)$ entries of Θ.

To the best of our knowledge, the above result is the best known complexity/accuracy trade-off for kernel matrices based on Green's functions of elliptic boundary value problems. In particular, we improve upon the methods in Chapter 5, where we showed that the Cholesky factors of Θ (as opposed to those of Θ^{-1}) can be approximated in computational complexity $\mathcal{O}(N\log^{2}(N)\log^{2d}(N/\epsilon))$ in time and $\mathcal{O}(N\log(N)\log^{d}(N/\epsilon))$ in space using zero-fill-in incomplete Cholesky factorization (Algorithm 15) applied to Θ.

Screening in theory and practice

The theory described in the last section covers any self-adjoined operator $\mathcal{L}$ with an associated quadratic form

$$\mathcal{L}[u] \coloneqq \int_{\Omega} u\mathcal{L}u \,\mathrm{d}x = \sum_{k=0}^{s} \int \sigma^{(k)}(x)\|D^{(k)}u(x)\|^{2}\,\mathrm{d}x$$

and $\sigma^{(s)} \in L^{2}(\Omega)$ positive almost everywhere. That is, $\mathcal{L}[u]$ is a weighted average of the squared norms of derivatives of u and thus measures the roughness of u. We can formally think of a Gaussian process with covariance function given by $\mathcal{G}$ as having density $\sim \exp(-\mathcal{L}[u]/2)$ and therefore assigning an exponentially low probability to "rough" functions, making it a prototypical smoothness prior. Theorems 10 and 9 imply that these Gaussian processes are subject to an exponentially strong screening effect in the sense that, after conditioning a set of ℓ-dense points, the conditional covariance of a given point decays exponentially with rate $\sim \ell^{-1}$, as shown in the first panel of Figure 6.6. The most closely related model in common use is the

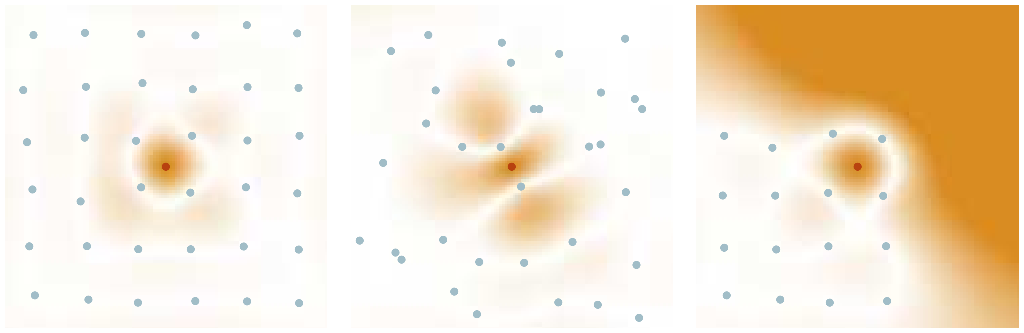


Figure 6.6: **Limitations of screening.** To illustrate the screening effect exploited by our methods, we plot the conditional correlation with the point in red conditional on the blue points. In the first panel, the points are evenly distributed, leading to a rapidly decreasing conditional correlation. In the second panel, the same number of points is irregularly distributed, slowing the decay. In the last panel, we are at the fringe of the set of observations, weakening the screening effect.

Matérn covariance function [167] that is the Green's function of an elliptic PDE of order s, when choosing the "smoothness parameter" ν as $\nu = s - d/2$. While our theory only covers $s \in \mathbb{N}$, we observed in Section 6.5 that Matérn kernels with non-integer values of s and even the "Cauchy class" [95] seem to be subject to similar behavior. In the second panel of Figure 6.6, we show that as the distribution of conditioning points becomes more irregular, the screening effect weakens. In our theoretical results, this is controlled by the upper bound on δ in (6.12). The screening effect is significantly weakened close to the boundary of the domain, as illustrated in the third panel of Figure 6.6 (cf. Figure 5.4). This is the reason why our theoretical results, different from the Matérn covariance, are restricted to Green's functions with zero Dirichlet boundary condition, which corresponds to conditioning the process to be zero on $\partial\Omega$. A final limitation is that the screening effect weakens as we take the order of smoothness to infinity, obtaining, for instance the Gaussian kernel. However, according to Theorem 2, this results in matrices that have efficient low-rank approximations, instead.

6.4 Extensions

We now present extensions of our method that improve its performance in practice. In Section 6.4.1, we show how to improve the approximation when Θ is replaced by $\Theta + R$, for R diagonal, as is frequently the case in statistical inference where R is the covariance matrix of additive, independent noise. In Section 6.4.2, we show how including the prediction points can improve the computational complexity (Section 6.4.2) or accuracy (Section 6.4.2) of the posterior mean and covariance. In

Section 6.4.3, we discuss memory savings and parallel computation for GP inference when we are only interested in computing the likelihood and the posterior mean and covariance (as opposed to, for example, sampling from $\mathcal{N}(0,\Theta)$ or computing products $v \mapsto \Theta v$).

We note that it is not possible to combine the variant in Section 6.4.1 with that in Section 6.4.3, and that the combination of the variants in Sections 6.4.1 and 6.4.2 might diminish accuracy gains from the latter. Furthermore, while Section 6.4.3 can be combined with Section 6.4.2 to compute the posterior mean, this combination cannot be used to compute the full posterior covariance matrix.

6.4.1 Additive noise

Assume that a diagonal noise term is added to Θ, so that $\Sigma = \Theta + R$, where R is diagonal. Extending the Vecchia approximation to this setting has been a major open problem in spatial statistics [62, 135, 136]. Applying our approximation directly to Σ would not work well because the noise term attenuates the exponential decay. Instead, given the approximation $\hat{\Theta}^{-1} = LL^{\top}$ obtained using our method, we can write, following Section 5.5.3:

$$\Sigma \approx \hat{\Theta} + R = \hat{\Theta}(R^{-1} + \hat{\Theta}^{-1})R.$$

Applying an incomplete Cholesky factorization with zero fill-in (Algorithm 15) to $R^{-1} + \hat{\Theta}^{-1} \approx \tilde{L}\tilde{L}^{\top}$, we have

$$\Sigma \approx (LL^{\top})^{-1}\tilde{L}\tilde{L}^{\top}R.$$

The resulting procedure, given in Algorithm 14, has asymptotic complexity $O(N\rho^{2d})$, because every column of the triangular factors has at most $O(\rho^{d})$ entries.

Following the intuition that Θ^{-1} is *essentially* an elliptic partial differential operator, $\Theta^{-1} + R^{-1}$ is *essentially* a partial differential operator with an added zero-order term, and its Cholesky factors can thus be expected to satisfy an exponential decay property just as those of Θ^{-1}. Indeed, as shown in Figure 5.5, the exponential decay of the Cholesky factors of $R^{-1} + \Theta^{-1}$ is as strong as for Θ^{-1}, even for large R. We suspect that this could be proved rigorously by adapting the proof of exponential decay in [190] to the discrete setting. We note that while independent noise is most commonly used, the above argument leads to an efficient algorithm whenever R^{-1} is approximately given by an elliptic PDE (possibly of order zero).

For small ρ, the additional error introduced by the incomplete Cholesky factorization can harm accuracy, which is why we recommend using the conjugate gradient

Algorithm 14 Including independent noise with covariance matrix R

Input: $\mathcal{G}$, $\{x_i\}_{i\in I}$, ρ, (λ,) and R
Output: $L, \tilde{L} \in \mathbb{R}^{N\times N}$ l. triang. in $\prec$

```
1: Comp. $\prec$ and $S \leftarrow S_{\prec,\ell,\rho}(S_{\prec,\ell,\rho,\lambda})$
2: Comp. $L$ using Alg. 12(13)
3: for $(i,j) \in S$ do
4:     $A_{ij} \leftarrow \langle L_{i,:}, L_{j,:}\rangle$
5: end for
6: $A \leftarrow A + R$
7: $\tilde{L} \leftarrow \mathtt{ichol}(A, S)$
8: return $L, \tilde{L}$
```

Algorithm 15 Zero fill-in incomplete Cholesky factorization (`ichol(A,S)`)

Input: $A \in \mathbb{R}^{N\times N}$, S
Output: $L \in \mathbb{R}^{N\times N}$ l. triang. in $\prec$

```
1: $L \leftarrow (0,\dots,0)(0,\dots,0)^\top$
2: for $j \in \{1,\dots,N\}$ do
3:     for $i \in \{j,\dots,N\} : (i,j) \in S$ do
4:         $L_{ij} \leftarrow A_{ij} - \langle L_{i,1:(j-1)}, L_{j,1:(j-1)}\rangle$
5:     end for
6:     $L_{:i} \leftarrow A_{:i}/\sqrt{A_{ii}}$
7: end for
8: return $L$
```

Figure 6.7: **Sums of independent processes.** Algorithms for approximating covariance matrices with added independent noise $\Theta + R$ (left), using the zero fill-in incomplete Cholesky factorization (right). Alternatively, the variants discussed in Section 5.3 could be used. See Section 6.4.1.

algorithm (CG) to invert $(R^{-1} + \hat{\Theta}^{-1})$ using $\tilde{L}$ as a preconditioner. In our experience, CG converges to single precision in a small number of iterations (~ 10).

Alternatively, higher accuracy can be achieved by using the sparsity pattern of $LL^\top$ (as opposed to that of L) to compute the incomplete Cholesky factorization of A in Algorithm 14; in fact, in our numerical experiments in Section 6.5.2, this approach was as accurate as using the exact Cholesky factorization of A over a wide range of ρ values and noise levels. The resulting algorithm still requires $O(N\rho^{2d})$ time, albeit with a larger constant. This is because for an entry (i,j) to be part of the sparsity pattern of $LL^\top$, there needs to exist a k such that both (i,k) and (j,k) are part of the sparsity pattern of L. By the triangle inequality, this implies that (i,j) is contained in the sparsity pattern of L obtained by doubling ρ. In conclusion, we believe that the above modifications allow us to compute an ϵ–accurate factorization in $O(N\log^{2d}(N/\epsilon))$ time and $O(N\log^{d}(N/\epsilon))$ space, just as in the noiseless case.

6.4.2 Including the prediction points

In GP regression, we are given N_{Tr} points of training data and want to compute predictions at N_{Pr} points of test data. We denote as $\Theta_{\mathrm{Tr},\mathrm{Tr}}$, $\Theta_{\mathrm{Pr},\mathrm{Pr}}$, $\Theta_{\mathrm{Tr},\mathrm{Pr}}$, $\Theta_{\mathrm{Pr},\mathrm{Tr}}$ the covariance matrix of the training data, the covariance matrix of the test data, and the covariance matrices of training and test data. Together, they form the joint covariance matrix $\begin{pmatrix} \Theta_{\mathrm{Tr},\mathrm{Tr}} & \Theta_{\mathrm{Tr},\mathrm{Pr}} \\ \Theta_{\mathrm{Pr},\mathrm{Tr}} & \Theta_{\mathrm{Pr},\mathrm{Pr}} \end{pmatrix}$ of training and test data. In GP regression with training data $y \in \mathbb{R}^{N_{\mathrm{Tr}}}$, we are interested in:

- Computation of the log-likelihood $\sim y^{\top} \Theta_{\mathrm{Tr},\mathrm{Tr}}^{-1} y + \operatorname{logdet} \Theta_{\mathrm{Tr},\mathrm{Tr}} + N \log(2\pi)$.
- Computation of the posterior mean $y^{\top} \Theta_{\mathrm{Tr},\mathrm{Tr}}^{-1} \Theta_{\mathrm{Tr},\mathrm{Pr}}$.
- Computation of the posterior covariance $\Theta_{\mathrm{Pr},\mathrm{Pr}} - \Theta_{\mathrm{Pr},\mathrm{Tr}} \Theta_{\mathrm{Tr},\mathrm{Tr}}^{-1} \Theta_{\mathrm{Tr},\mathrm{Pr}}$.

In the setting of Theorem 22, our method can be applied to accurately approximate the matrix $\Theta_{\mathrm{Tr},\mathrm{Tr}}$ in near-linear cost. The training covariance matrix can then be replaced by the resulting approximation for all downstream applications.

However, approximating instead the joint covariance matrix of training and prediction variables improves (1) stability and accuracy compared to computing the KL-optimal approximation of the training covariance alone, (2) computational complexity by circumventing the computation of most of the $N_{\mathrm{Tr}} N_{\mathrm{Pr}}$ entries of the off-diagonal part $\Theta_{\mathrm{Tr},\mathrm{Pr}}$ of the covariance matrix.

We can add the prediction points before or after the training points in the elimination ordering.

Ordering the prediction points first, for rapid interpolation

The computation of the mixed covariance matrix $\Theta_{\mathrm{Pr},\mathrm{Tr}}$ can be prohibitively expensive when interpolating with a large number of prediction points. This situation is common in spatial statistics when estimating a stochastic field throughout a large domain. In this regime, we propose to order the $\{x_i\}_{i \in I}$ by first computing the reverse maximin ordering $\prec_{\mathrm{Tr}}$ of only the training points as described in Section 6.3.1 using the original Ω, writing ℓ_{Tr} for the corresponding length scales. We then compute the reverse maximin ordering $\prec_{\mathrm{Pr}}$ of the prediction points using the modified $\tilde{\Omega} \coloneqq \Omega \cup \{x_i\}_{i \in I_{\mathrm{Tr}}}$, obtaining the length scales ℓ_{Pr}. Since $\tilde{\Omega}$ contains $\{x_i\}_{i \in I_{\mathrm{Tr}}}$, when computing the ordering of the prediction points, prediction points close to the training set will tend to have a smaller length-scale ℓ than in the naive application of the algorithm, and thus, the resulting sparsity pattern will have fewer nonzero entries. We then order the prediction points before the training points and compute $S_{(\prec_{\mathrm{Pr}}, \prec_{\mathrm{Tr}}),(\ell_{\mathrm{Pr}}, \ell_{\mathrm{Tr}}),\rho}$ or $S_{(\prec_{\mathrm{Pr}}, \prec_{\mathrm{Tr}}),(\ell_{\mathrm{Pr}}, \ell_{\mathrm{Tr}}),\rho,\lambda}$ following the same procedure as in Sections 6.3.1 and 6.3.2, respectively. The distance of each point in the prediction set to the training set can be computed in near-linear complexity using, for example, a minor variation of Algorithm 11. Writing L for the resulting Cholesky factor of the joint precision matrix, we can approximate $\Theta_{\mathrm{Pr},\mathrm{Pr}} \approx L_{\mathrm{Pr},\mathrm{Pr}}^{-\top} L_{\mathrm{Pr},\mathrm{Pr}}^{-1}$ and $\Theta_{\mathrm{Pr},\mathrm{Tr}} \approx L_{\mathrm{Pr},\mathrm{Pr}}^{-\top} L_{\mathrm{Tr},\mathrm{Pr}}^{\top}$ based on submatrices of L. See Sections .3.3 and 21 for additional details. We note that the idea of ordering the prediction points first (last, in their notation) has already been proposed by [136] in the context of the Vecchia approximation, although without providing an explicit algorithm.

If one does not use the method in Section 6.4.1 to treat additive noise, then the method described in this section amounts to making each prediction using only $O(\rho^d)$ nearby points. In the extreme case where we only have a single prediction point, this means that we are only using $O(\rho^d)$ training values for prediction. On the one hand, this can lead to improved robustness of the resulting estimator, but on the other hand, it can lead to some training data being missed entirely.

Ordering the prediction points last, for improved robustness

If we want to use the improved stability of including the prediction points, maintain near-linear complexity, and use all N_{Tr} training values for the prediction of even a single point, we have to include the prediction points *after* the training points in the elimination ordering. Naively, this would lead to a computational complexity of $O(N_{\mathrm{Tr}}(\rho^d + N_{\mathrm{Pr}})^2)$, which might be prohibitive for large values of N_{Pr}. If it is enough to compute the posterior covariance only among m_{Pr} small *batches* of up to n_{Pr} predictions each (often, it makes sense to choose $n_{\mathrm{Pr}} = 1$), we can avoid this increase of complexity by performing prediction on groups of only n_{Pr} at once, with the computation for each batch only having computational complexity $O(N_{\mathrm{Tr}}(\rho^d + n_{\mathrm{Pr}})^2)$. A naive implementation would still require us to perform this procedure m_{Pr} times, eliminating any gains due to the batched procedure. However, careful use of the Sherman-Morrison-Woodbury matrix identity allows to to reuse the biggest part of the computation for each of the batches, thus reducing the computational cost for prediction and computation of the covariance matrix to only $O(N_{\mathrm{Tr}}((\rho^d + n_{\mathrm{Pr}})^2 + (\rho^d + n_{\mathrm{Pr}})m_{\mathrm{Pr}}))$. This procedure is detailed in Section .3.3 and summarized in Section 23.

6.4.3 GP regression in $O(N + \rho^{2d})$ space complexity

When deploying direct methods for approximate inversion of kernel matrices, a major difficulty is the superlinear memory cost that they incur. This, in particular, poses difficulties in a distributed setting or on graphics processing units. In the following, $I = I_{\mathrm{Tr}}$ denotes the indices of the training data, and we write $\Theta := \Theta_{\mathrm{Tr},\mathrm{Tr}}$, while I_{Pr} denotes those of the test data. In order to compute the log-likelihood, we need to compute the matrix-vector product $L^{\rho,\top} y$, as well as the diagonal entries of L^{ρ}. This can be done by computing the columns $L^{\rho}_{:,k}$ of L^{ρ} individually using (6.3) and setting $(L^{\rho,\top} y)_k = (L^{\rho}_{:,k})^{\top} y$, $L^{\rho}_{kk} = (L^{\rho}_{:,k})_k$, without ever forming the matrix L^{ρ}. Similarly, in order to compute the posterior mean, we only need to compute $\Theta^{-1} y = L^{\rho,\top} L^{\rho} y$, which only requires us to compute each column of L^{ρ} twice, without ever forming the entire matrix. In order to compute the posterior covariance, we need to compute the matrix-matrix product $L^{\rho,\top}\Theta_{\mathrm{Tr},\mathrm{Pr}}$, which again can be performed by computing each column of L^{ρ} once without ever forming the entire matrix L^{ρ}. However, it does require us to know beforehand at which points we want to make predictions. The submatrices Θ_{s_i,s_i} for all i belonging to the supernode $\tilde{k}$ (i.e., $i \rightsquigarrow \tilde{k}$) can be formed from a list of the elements of $\tilde{s}_k$. Thus, the overall memory complexity of the resulting algorithm is $O(\sum_{k \in \tilde{I}} \# \tilde{s}_k) = O(N_{\mathrm{Tr}} + N_{\mathrm{Pr}} + \rho^{2d})$. The above described procedure is implemented in Algorithms 19 and 20 in Section .3.1. In a distributed setting with workers $W_1, W_2, \ldots$, this requires communicating only $O(\# \tilde{s}_k)$ floating-point numbers to worker W_k, which then performs $O((\# \tilde{s}_k)^3)$ floating-point operations; a naive implementation would require the communication of $O((\# \tilde{s})^2)$ floating-point numbers to perform the same number of floating-point operations.

6.5 Applications and numerical results

We conclude with numerical experiments studying the practical performance of our method. The Julia code can be found under `https://github.com/f-t-s/cholesky_by_KL_minimization`.

6.5.1 Gaussian process regression and aggregation

We begin our numerical experiments with two-dimensional ($d = 2$) synthetic data. We use circulant embeddings [98, 225], [`https://github.com/PieterjanRobbe/GaussianRandomFields.jl`] to create 10^3 samples of a Gaussian process with Matérn covariance function at 10^6 locations on a regular grid in $\Omega = [0, 1]^2$. From these 10^6 locations, we select 2×10^4 prediction points and use the remaining points as training data. As illustrated in Figure 6.8 (left panel), half of the prediction points form two elliptic regions devoid of any training points (called "region"), while the remaining prediction points are interspersed among the training points (called "scattered"). We then use the "prediction points first" approach of Section 6.4.2 and the

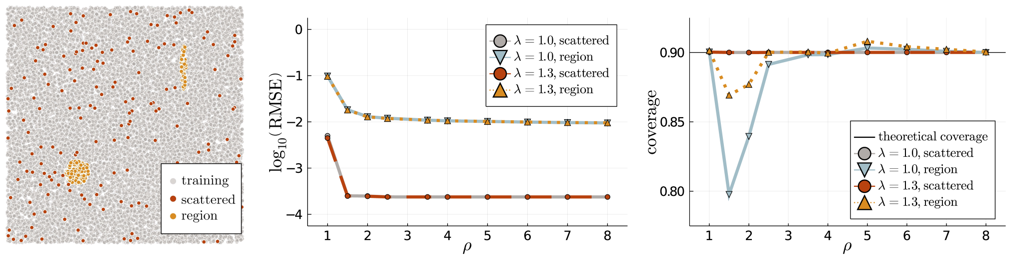


Figure 6.8: **Prediction and uncertainty quantification with Matérn covariance.** We show the accuracy of our approximation with and without aggregation for a Gaussian process with Matérn covariance ($\nu = 3/2$) on a grid of size 10^6 on the unit square. (Left) Randomly sampled 2 percent of the training and prediction points. (Middle) RMSE, averaged over prediction points and 1,000 realizations. (Right) Empirical coverage of 90% prediction intervals computed from the posterior covariance.

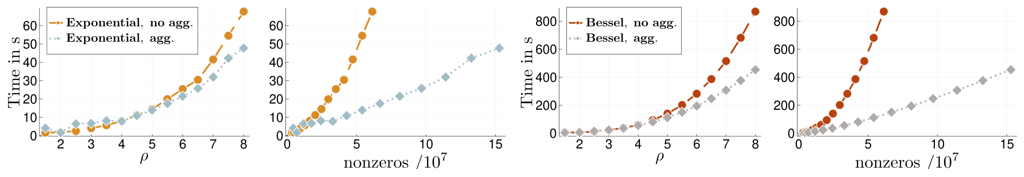


Figure 6.9: **Computational cost of factorization.** Time for computing the factor L^ρ with or without aggregation ($N = 10^6$), as a function of ρ and of the number of nonzero entries. For the first two panels, the Matérn covariance function was computed using a representation in terms of exponentials, while for the second two panels they were computed using (slower) Bessel function evaluations. Computations performed on an Intel®Core™i7-6400 CPU with 4.00GHz and 64 GB of RAM. The second and fourth panels show that aggregation leads to faster computation despite producing much denser Cholesky factors (and hence higher accuracy).

aggregated sparsity pattern $\tilde{S}_{<,\ell,\rho,\lambda}$ of Section 6.3.2 with $\lambda \in \{1.0, 1.3\}$, to compute the posterior distributions at the prediction points from the values at the training points. In Figure 6.8, we report the RMSE of the posterior means, as well as the empirical coverage of the 90% posterior intervals, averaged over all 10^3 realizations, for a range of different ρ. Note that while the RMSE between the aggregated ($\lambda = 1.3$) and non-aggregated ($\lambda = 1.0$) is almost the same, the coverage converges significantly faster to the correct value with $\lambda = 1.3$.

We further provide timing results for 10^6 training points uniformly distributed in $[0, 1]^2$ comparing the aggregated and non-aggregated version of the algorithm in Figure 6.9. As predicted by the theory, the aggregated variant scales better as we are increasing ρ. This holds true both when using `Intel® oneMKL Vector Mathematics functions` library to evaluate the exponential function, or when us-

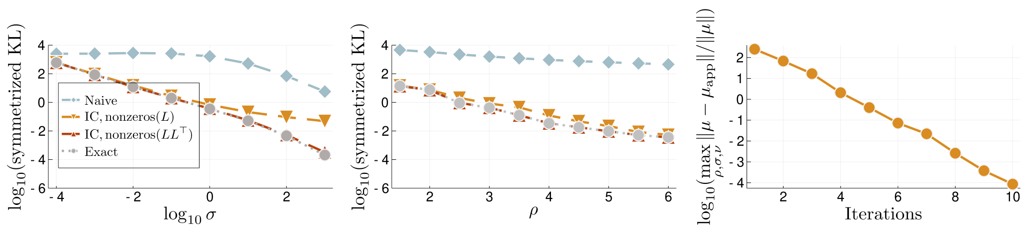


Figure 6.10: **Factorization with additive noise.** Comparison of the methods proposed in Section 6.4.1 for approximating $\Sigma = \Theta + R$, where Θ is based on a Matérn covariance with range parameter 0.5 and smoothness $\nu = 3/2$ at $N = 10^4$ uniformly sampled locations on the unit square, and $R = \sigma^2 I$ is additive noise. For each approximation, we compute the symmetrized KL divergence (the sum of the KL-divergences with either ordering of the two measures) to the true covariance. "Naive": Directly apply Algorithm 13 to Σ. "Exact": Apply Algorithm 13 to Θ, then compute $\tilde{L}$ as the exact Cholesky factorization of $A := R^{-1} + \hat{\Theta}^{-1}$. "IC": Apply Algorithm 13 to Θ, then compute $\tilde{L}$ using incomplete Cholesky factorization of A on the sparsity pattern of either L or $LL^\top$. (Left) Varying σ, fixed $\rho = 3.0$. (Middle) Varying ρ, fixed $\sigma = 1.0$. (Right) Maximal relative error (over the above σ, ρ, $\nu \in \{1/2, 3/2, 5/2\}$ and 10 random draws) of inverting A using up to 10 conjugate-gradient iterations (x-axis), with IC, nonzeros(L) as preconditioner.

ing `amos` to instead evaluate the modified Bessel function of the second kind. While the former is faster and emphasizes the improvement from $O(N\rho^{3d})$ to $O(N\rho^{2d})$ for the complexity of computing the factorization, the latter can be used to evaluate Matérn kernels with arbitrary smoothness. Due to being slower, using Bessel functions highlights the improvement from needing $O(N\rho^{2d})$ matrix evaluations without the aggregation to just $O(N\rho^d)$. By plotting the number of nonzeros used for the two approaches, we see that the aggregated version is faster to compute despite using many more entries of Θ than the non-aggregated version. Thus, aggregation is both faster and more accurate for the same value of ρ, which is why we recommend using it over the non-aggregated variant.

6.5.2 Adding noise

We now experimentally verify the claim that the methods described in Section 6.4.1 enable accurate approximation in the presence of independent noise while preserving the sparsity, and thus computational complexity, of our method. To this end, pick a set of $N = 10^4$ points uniformly at random in $\Omega = [0, 1]^2$, use a Matérn kernel with smoothness $\nu = 3/2$, and add I.I.D. noise with variance σ^2. We use an aggregation parameter $\lambda = 1.5$. As shown in Figure 6.10, our approximation stays accurate over a wide range of values of both ρ and σ, even for the most frugal version of our method.

The asymptotic complexity for both incomplete-Cholesky variants is $O(N\rho^{2d})$, with the variant using the sparsity pattern of $LL^{\top}$ being roughly equivalent to doubling ρ. Hence, to avoid additional memory overhead, we recommend using the sparsity pattern of L as a default choice; the accuracy of the resulting log-determinant of Σ should be sufficient for most settings, and the accuracy for solving systems of equations in Σ can easily be increased by adding a few iterations of the conjugate gradient algorithm.

6.5.3 Including prediction points

We continue by studying the effects of including the prediction points in the approximation, as described in Sections 6.4.2 and 6.4.2. We compare not including the predictions points in the approximation with including them either before or after training points in the approximation. We compare the accuracy of the approximation of the posterior mean and standard deviation over three different geometries and a range of different values for ρ. The results, displayed in Figure 6.11, show that including the prediction points can increase the accuracy by multiple orders of magnitude. The performance difference between the two schemes for including prediction points varies over different geometries, degrees of regularity, and values of ρ. If the number of prediction points is comparable to the number of training points, the only way to avoid quadratic scaling in the number of points is to order the prediction points first, making this approach the method of choice. If we only have few prediction points, ordering the prediction variables last can improve the accuracy for low orders of smoothness, especially in settings in which only a small part of the training data is used in the prediction-variables-first approach (e.g., second row in Figure 6.11).

6.5.4 Comparison to HSS matrices

As described in the introduction, there are many existing methods for the approximation and inversion of dense covariance matrices. Hierarchically semiseparable (HSS) matrices [251] are natural candidates for comparison with our method because they are amenable to a Cholesky factorization [150], implementations of which are available in existing software packages. They are also closely related to hierarchically off-diagonal low-rank (HODLR) matrices, which have been promoted as tools for Gaussian process regression [11]. We consider a regression problem with 50^3 training points on a randomly perturbed regular grid and 50 test points distributed uniformly at random in the unit cube. Using the Matérn covari-

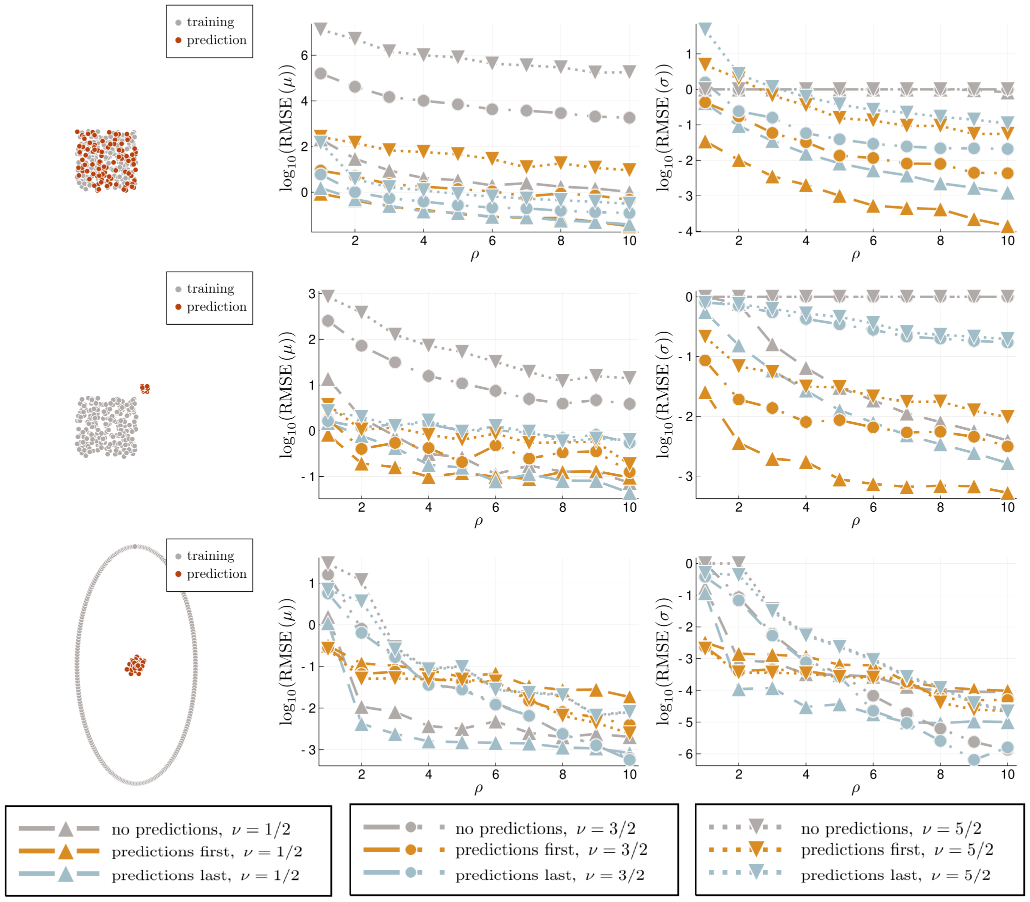


Figure 6.11: **Including prediction points.** To analyze the effects of including the prediction points into the approximation, we consider three datasets. Each consists of 5×10^4 training points and 10^2 test points, averaged over ten independent realizations of the Gaussian process. We use Matérn kernels with range parameter 0.5 and smoothness $\nu \in \{1/2, 3/2, 5/2\}$, with ρ ranging from 1.0 to 10.0. We do not use aggregation since it might lead to slightly different sparsity patterns for the three variants, possibly polluting the results. On the y-axis we plot the RMSE of the posterior mean and standard deviation, scaled in each point by the reciprocal of the true posterior standard deviation. In almost all cases, including the prediction points into the approximation improves the accuracy. The comparison between ordering the predictions first or last is complicated, but "predictions-last" seems to perform better for lower smoothness and "predictions-first" for higher smoothness.

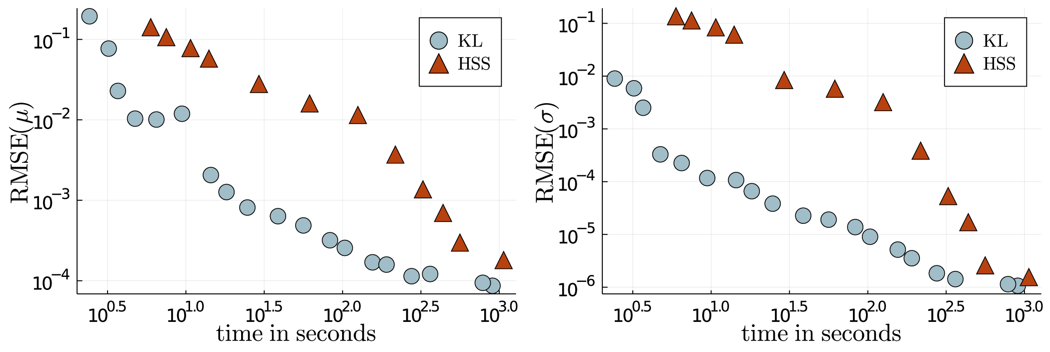


Figure 6.12: **Comparison to HSS matrices.** We compare the accuracy and computational time of our method described in Section 6.4.2 with the HSS implementation of `H2Pack` [123]. Each point corresponds to a different run with different parameters (ρ, tolerance, and diagonal shift). Throughout this experiment, we use the aggregation scheme described in Section 6.3.2 with $\lambda = 1.25$. The left plot shows the RMSE of the posterior mean and the right plot that of the posterior standard deviation. Our method is significantly faster for a wide range of target accuracies.

ance with $\nu = 3/2$ and length scale $l = 0.2$, we compute the posterior mean and standard deviation for 50 samples using the method described in Section 6.4.2 and the HSS implementation of `H2Pack` [123], both using eight threads on an Intel® Skylake ™CPU with 2.10GHz and 192 GB of RAM. In Figure 6.12, we report the computational time and accuracy for a wide range of tuning parameters (ρ for our method, error tolerance and diagonal shift for HSS). We ignore the setup cost for both methods, which includes the selection of the "*numerical proxy points*" for the HSS approach. Our experiments show that for a given target accuracy, our method is an order of magnitude faster than HSS, despite the highly optimized implementation of the latter. For very high accuracies, the gap between the methods closes, but the memory cost of HSS approaches that of the dense problem, preventing us from further increasing the target accuracy. We note that for three-dimensional problems, $\mathcal{H}^2$-matrices have better asymptotic complexity than HSS matrices, making them a possibly stronger competitor; however, the Cholesky factorization of $\mathcal{H}^2$-matrices is considerably more complicated and not implemented in `H2Pack`. Another possible approach is the inversion of an $\mathcal{H}^2$ approximation using conjugate gradient methods, using our method or HSS matrices ([252]) as a preconditioner. We defer a more comprehensive comparison to the various kinds of hierarchical matrices to future work.

6.5.5 Single-layer boundary element methods

We now provide an application to boundary element methods. For a domain $\Omega \in \mathbb{R}^d$ with boundary $\partial\Omega$, let us assume that we want to solve the Dirichlet boundary value problem

$$\begin{aligned} -\Delta u(x) &= 0, & \forall x \in \Omega \\ u(x) &= g(x), & \forall x \in \partial\Omega. \end{aligned}$$

For $d = 3$, the Green's function of the Laplace operator is given by the gravitational / electrostatic potential

$$\mathcal{G}_{\mathbb{R}^3}(x, y) = \frac{1}{4\pi |x - y|}.$$

Under mild regularity assumptions, one can verify that

$$u = \int\limits_{x \in \partial\Omega} \mathcal{G}_{\mathbb{R}^3}(x, \cdot) h(x) \,\mathrm{d}x, \quad \text{for } h \text{ the solution of} \quad g = \int\limits_{x \in \partial\Omega} \mathcal{G}_{\mathbb{R}^3}(x, \cdot) h(x) \,\mathrm{d}x.$$

Let us choose finite dimensional basis functions $\{\phi_i\}_{i \in I_{\mathrm{Pr}}}$ in the interior of Ω and $\{\phi_i\}_{i \in I_{\mathrm{Tr}}}$ on the boundary of Ω. We form the matrix $\Theta \in \mathbb{R}^{(I_{\mathrm{Tr}} \cup I_{\mathrm{Pr}}) \times (I_{\mathrm{Tr}} \cup I_{\mathrm{Pr}})}$ as

$$\Theta_{ij} := \int_{x \in \mathcal{D}_i} \int_{y \in \mathcal{D}_j} \phi_i(x) \mathcal{G}_{\mathbb{R}^3}(x, y) \, \phi_j(y) \,\mathrm{d}y \,\mathrm{d}x, \quad \text{where} \quad \mathcal{D}_p = \begin{cases} \partial\Omega, & \text{for } p \in I_{\mathrm{Tr}} \\ \Omega, & \text{for } p \in I_{\mathrm{Pr}} \end{cases} \tag{6.14}$$

and denote as $\Theta_{\mathrm{Tr},\mathrm{Tr}}, \Theta_{\mathrm{Tr},\mathrm{Pr}}, \Theta_{\mathrm{Pr},\mathrm{Tr}}, \Theta_{\mathrm{Pr},\mathrm{Pr}}$ its restrictions to the rows and columns indexed by I_{Tr} or I_{Pr}. Defining

$$\vec{g}_i := \int\limits_{x \in \partial\Omega} \phi_i(x) g(x) \,\mathrm{d}x, \quad \forall i \in I_{\mathrm{Tr}} \quad \text{and} \quad \vec{u}_i := \int\limits_{x \in \partial\Omega} \phi_i(x) u(x) \,\mathrm{d}x, \quad \forall i \in I_{\mathrm{Pr}},$$

we approximate $\vec{u}$ as

$$\vec{u} \approx \Theta_{I_{\mathrm{Pr}}, I_{\mathrm{Tr}}} \Theta_{I_{\mathrm{Tr}}, I_{\mathrm{Tr}}}^{-1} \vec{g}. \tag{6.15}$$

This is a classical technique for the solution of partial differential equations, known as single layer boundary element methods [214]. However, it can also be seen as Gaussian process regression with u being the conditional mean of a Gaussian process with covariance function $\mathcal{G}$, conditional on the values of the process on $\partial\Omega$. Similarly, it can be shown that the zero boundary value Green's function is given by the posterior covariance of the same process.

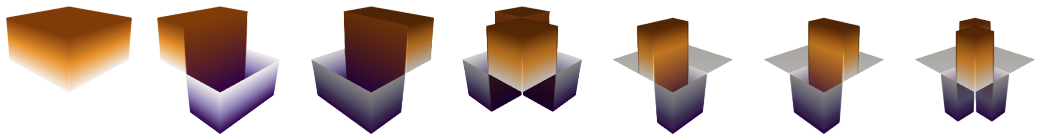


Figure 6.13: **Orthogonal basis from subdivision.** We recursively divide each panel of $\partial\Omega$. The basis functions on finer levels are constructed as linear combinations of indicator functions that are orthogonal to functions on coarser levels.

The Laplace operator in three dimensions does not satisfy $s > d/2$ (cf. Section 6.3.3). Therefore, the variance of pointwise evaluations at $x \in \mathbb{R}^3$ given by $\mathcal{G}_{\mathbb{R}^3}(x, x)$ is infinite and we cannot let $\{\phi_i\}_{i \in I_{\mathrm{Pr}}}$ be Dirac-functions as in other parts of this work.

Instead, we recursively subdivide the boundary $\partial\Omega$ and use Haar-type wavelets as in Example 2 for $\{\phi_i\}_{i \in I_{\mathrm{Tr}}}$. For our numerical experiments, we will consider $\Omega := [0,1]^3$ to be the three-dimensional unit cube. On each face of $\partial\Omega$, we then obtain a multiresolution basis by hierarchical subdivision, as shown in Figure 6.13. In this case, the equivalent of a maximin ordering is an ordering from coarser to finer levels, with an arbitrary ordering within each level. We construct our sparsity pattern as

$$S_{<,\ell_j,\rho} := \{ (i,j) : i \geq j, \operatorname{dist}(x_i, x_j) \leq \rho \ell_j + \sqrt{2}(\ell_i + \ell_j) \}, \tag{6.16}$$

where for $i \in I_{\mathrm{Tr}}$, x_i is defined as the center of the support of ϕ_i and ℓ_i as half of the side-length of the (quadratic) support of ϕ_i. The addition of $\sqrt{2}(\ell_i + \ell_j)$ to the right-hand side ensures that the entries corresponding to neighboring basis functions are always added to the sparsity pattern.

We construct a solution u of the Laplace equation in Ω as the sum over $N_c = 2000$ charges with random signs $\{s_i\}_{1 \leq i \leq N_c}$ located at points $\{c_i\}_{1 \leq i \leq N_c}$ We then pick a set of N_{Pr} points $\{x_i\}_{i \in I_{\mathrm{Pr}}}$ inside of Ω and try to predict the values $\{u(x_i)\}_{i \in I_{\mathrm{Pr}}}$ using Equation (6.15) and the method described in Section 6.4.2. We compare the computational time, the number of entries in the sparsity pattern, and the mean accuracy of the approximate method for $\rho \in \{1.0, 2.0, 3.0\}$, as well as the exact solution of the linear system. We use different levels of discretization $q \in \{3, \ldots, 8\}$, leading to a spatial resolution of up to 2^{-8}. As shown in Figure 6.14, even using $\rho = 1.0$ leads to near-optimal accuracy, at a greatly reduced computational cost.

There exists a rich literature on the numerical solution of boundary element equations [214], and we are not yet claiming improvement over the state-of-the-art. Presently,

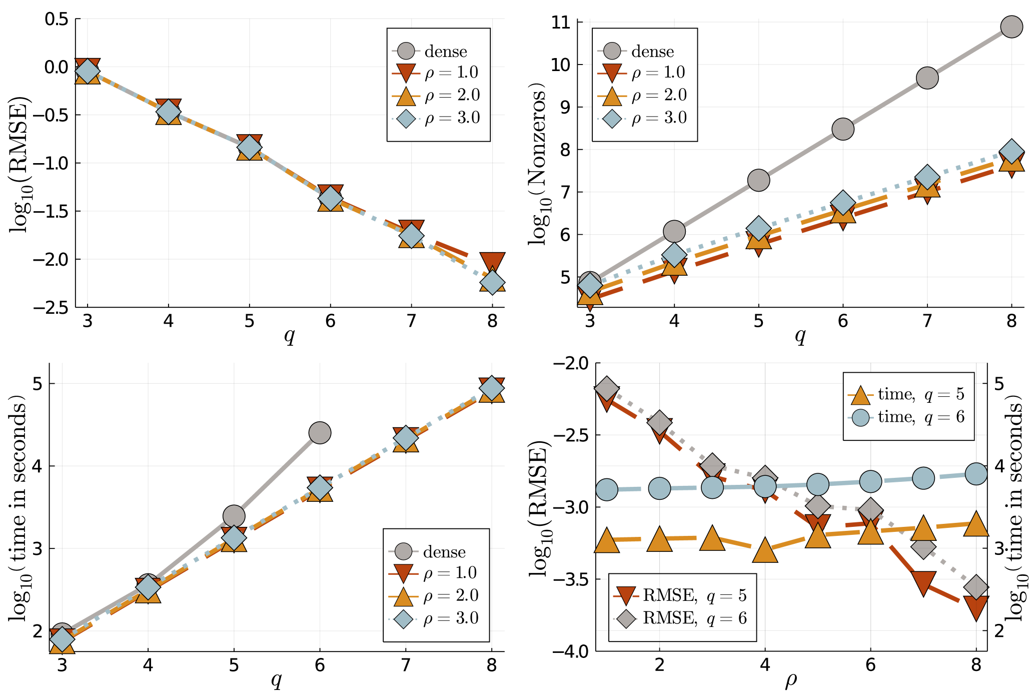


Figure 6.14: **Solving boundary value problems by KL-minimization.** Accuracy and computational complexity in boundary value problem. We compare the root mean square error, number of nonzeros of sparsity pattern, and the computational time for the exact boundary element method and using our approximation for $\rho \in \{1, 2, 3\}$. The dense solution is prohibitively expensive for $q > 6$, which is why accuracy and computational time for these cases are missing. The reason that the computational time is hardly affected by different choices of ρ is due to the fact that entries $\left(\Theta_{\text{Tr},\text{Tr}}\right)_{ij}$ for nearby ϕ_i, ϕ_j are significantly more expensive to compute than for distant ones when using an off-the-shelf adaptive quadrature rule. The computations were performed on 32 threads of an Intel® Skylake ™CPU with 2.10GHz and 192 GB of RAM. In the first figure, we plot the RMSE compared to the true solution of the PDE as a function of $q \approx \log(N)$. In the last figure, we compute the RMSE between *dense* computation and our method, as well as its computational time, as a function of ρ.

the majority of the computational time is spent computing the matrix entries of $\Theta_{\mathrm{Tr},\mathrm{Tr}}$. In order to compete with the state-of-the-art in terms of wall-clock times, we would need to implement more efficient quadrature rules, which is beyond the scope of this paper. Due to the embarrassing parallelism of our method, together with the high accuracy obtained even for small values of ρ, we hope that it will become a useful tool for solving boundary integral equations, but we defer a detailed study to future work.

6.6 Conclusions

In this work, we have shown that, surprisingly, the optimal (in KL-divergence) inverse Cholesky factor of a positive definite matrix, subject to a sparsity pattern, can be computed in closed form. In the special case of Green's matrices of elliptic boundary value problems in d dimensions, we show that by applying this method to the elimination orderings and sparsity patterns described in Chapters 3 and 4, one can compute the sparse inverse Cholesky factor with accuracy ϵ in computational complexity $O(N \log^{2d}(N/\epsilon))$ using only $O(N \log^{d}(N/\epsilon))$ entries of the dense Green's matrix. This improves upon the state-of-the-art in this classical problem. We also propose a variety of improvements, capitalizing on the improved stability, parallelism, and memory footprint of our method. Finally, we show how to extend our approximation to the setting with additive noise, resolving a major open problem in spatial statistics.

Chapter 7

COMPETITIVE GRADIENT DESCENT

7.1 Introduction

Competitive optimization. Whereas traditional optimization is concerned with a single agent trying to optimize a cost function, competitive optimization extends this problem to the setting of multiple agents, each trying to minimize their own cost function, which in general depends on the actions of all agents. For this thesis, we will concentrate on the case of two such agents:

$$\min_{x \in \mathbb{R}^m} f(x, y), \quad \min_{y \in \mathbb{R}^n} g(x, y) \tag{7.1}$$

for two functions $f, g : \mathbb{R}^m \times \mathbb{R}^n \longrightarrow \mathbb{R}$.
In single agent optimization, the solution of the problem consists of the minimizer of the cost function. In competitive optimization, the right definition of *solution* is less obvious, but often one is interested in computing Nash– or strategic equilibria: Pairs of strategies, such that no player can decrease their costs by unilaterally changing their strategies. If f and g are not convex, finding a global Nash equilibrium is typically impossible and the appropriate solution concept may depend on the application.

The benefits of competition. While competitive optimization problems arise naturally in mathematical economics and game/decision theory [184], they also provide a highly expressive and transparent language to formulate algorithms in a wide range of domains. In optimization [37] and statistics [124], it has long been observed that competitive optimization is a natural way to encode robustness requirements of algorithms. More recently, researchers in machine learning have been using multi-agent optimization to design highly flexible objective functions for reinforcement learning [157, 194, 196, 242, 245] and generative models [96]. We believe that this approach has a lot of untapped potential, the realization of which depends crucially on the development of efficient and reliable algorithms for the numerical solution of competitive optimization problems.

Gradient descent/ascent and the cycling problem. For differentiable objective functions, the most naive approach to solving (7.1) is gradient descent ascent (GDA),

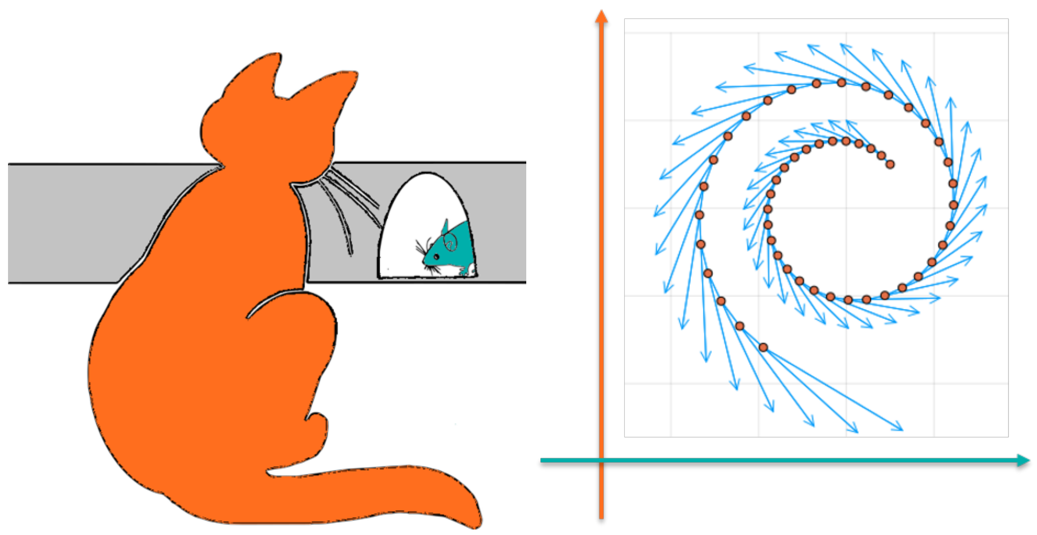


Figure 7.1: **The cycling problem.** The cycling problem of GDA arises, because each chooses the optimal action according to the *last* action of the other agent.

whereby both players independently change their strategy in the direction of steepest descent of their cost function. Unfortunately, this procedure features oscillatory or divergent behavior even in the simple case of a bilinear game ($f(x, y) = x^\top y = -g(x, y)$) (see Figure 7.4). In game-theoretic terms, GDA lets both players choose their new strategy optimally with respect to the last move of the other player (see Figure 7.1). Thus, the cycling behavior of GDA is not surprising: It is the analog of *"Rock! Paper! Scissors! Rock! Paper! Scissors! Rock! Paper!..."* in the eponymous hand game. While gradient descent is a reliable basic *workhorse* for single-agent optimization, GDA can not play the same role for competitive optimization. The lack of such a *workhorse* greatly hinders the broader adoption of methods based on competition.

Existing works. Most existing approaches to stabilizing GDA follow one of three lines of attack.
In the special case $f = -g$, the problem can be written as a minimization problem $\min_x F(x)$, where $F(x) := \max_y f(x, y)$. For certain structured problems, [93] use techniques from convex optimization [181] to minimize the implicitly defined F. For general problems, the two-scale update rules proposed in [96, 117, 174] can be seen as an attempt to approximate F and its gradients.
In GDA, players pick their next strategy based on the last strategy picked by the other players. Methods based on *follow the regularized leader* [101, 217], *fictitious play* [44], *predictive updates* [254], *opponent learning awareness* [86], and *opti-*

mism [61, 172, 201] propose more sophisticated heuristics that the players could use to predict each other's next move. Algorithmically, many of these methods can be considered variations of the *extragradient method* [143](see also [76, Chapter 12]). Finally, some methods directly modify the gradient dynamics, either by promoting convergence through gradient penalties [173], or by attempting to disentangle convergent *potential* parts from rotational *Hamiltonian* parts of the vector field [24, 89, 147].

Contribution of this chapter. Our main *conceptual* objection to most existing methods is that they lack a clear game-theoretic motivation, but instead rely on the ad-hoc introduction of additional assumptions, modifications, and model parameters.
Their main *practical* shortcoming is that to avoid divergence, the stepsize has to be chosen inversely proportional to the magnitude of the interaction of the two players (as measured by $D_{xy}^2 f$, $D_{xy}^2 g$).
On the one hand, the small stepsize results in slow convergence. On the other hand, a stepsize small enough to prevent divergence will not be known in advance in most problems. Instead, it has to be discovered through tedious trial and error, which is further aggravated by the lack of a good diagnostic for improvement in multi-agent optimization (which is given by the objective function in single-agent optimization).
We alleviate the above-mentioned problems by introducing a novel algorithm, *competitive gradient descent* (CGD) that is obtained as a natural extension of gradient descent to the competitive setting. Recall that in the single player setting, the gradient descent update is obtained as the optimal solution to a regularized linear approximation of the cost function. In the same spirit, the update of CGD is given by the Nash equilibrium of a regularized *bilinear* approximation of the underlying game. The use of a bilinear as opposed to linear approximation lets the local approximation preserve the competitive nature of the problem, significantly improving stability. We prove (local) convergence results of this algorithm in the case of (locally) convex-concave zero-sum games. We also show that stronger interactions between the two players only improve convergence without requiring an adaptation of the stepsize. In comparison, the existing methods need to reduce the stepsize to match the increase of the interactions to avoid divergence, which we illustrate on a series of polynomial test cases considered in previous works.
We begin our numerical experiments by trying to use a GAN on a bimodal Gaus-

sian mixture model. Even in this simple example, trying five different (constant) stepsizes under RMSProp, the existing methods diverge. The typical solution would be to decay the learning rate. However, even with a constant learning rate, CGD succeeds with all these stepsize choices to approximate the main features of the target distribution. In fact, throughout our experiments we *never* saw CGD diverge. In order to measure the convergence speed more quantitatively, we next consider a nonconvex matrix estimation problem, measuring computational complexity in terms of the number of gradient computations performed. We observe that all methods show improved speed of convergence for larger stepsizes, with CGD roughly matching the convergence speed of optimistic gradient descent [61], at the same stepsize. However, as we increase the stepsize, other methods quickly start diverging, whereas CGD continues to improve, thus being able to attain significantly better convergence rates (more than two times as fast as the other methods in the noiseless case, with the ratio increasing for larger and more difficult problems). For small stepsize or games with weak interactions, on the other hand, CGD automatically invests less computational time per update, thus gracefully transitioning to a cheap correction of GDA at minimal computational overhead. We furthermore present an application to constrained optimization problems arising in model-based reinforcement learning and refer to [198] for recent applications of CGD to multi-agent reinforcement learning.

7.2 Competitive gradient descent

We propose a novel algorithm, which we call *competitive gradient descent* (CGD), for the solution of competitive optimization problems $\min_{x \in \mathbb{R}^m} f(x, y)$, $\min_{y \in \mathbb{R}^n} g(x, y)$, where we have access to function evaluations, gradients, and Hessian-vector products of the objective functions.[1]

Algorithm 16 Competitive Gradient Descent (CGD)

```
**for** $0 \leq k \leq N - 1$ **do**
    $x_{k+1} = x_k - \eta \left(\mathrm{Id} - \eta^2 D^2_{xy} f D^2_{yx} g\right)^{-1} \left(\nabla_x f - \eta D^2_{xy} f \nabla_y g\right)$
    $y_{k+1} = y_k - \eta \left(\mathrm{Id} - \eta^2 D^2_{yx} g D^2_{xy} f\right)^{-1} \left(\nabla_y g - \eta D^2_{yx} g \nabla_x f\right)$
**end for**
**return** $(x_N, y_N)$
```

[1]Here and in the following, unless otherwise mentioned, all derivatives are evaluated in the point (x_k, y_k)

How to linearize a game. To motivate this algorithm, we remind ourselves that gradient descent with stepsize η applied to the function $f : \mathbb{R}^m \longrightarrow \mathbb{R}$ can be written as

$$x_{k+1} = \arg\min_{x \in \mathbb{R}^m} (x^\top - x_k^\top) \nabla_x f(x_k) + \frac{1}{2\eta} \|x - x_k\|^2. \tag{7.2}$$

This can be interpreted as a (single) player solving a local linear approximation of the (minimization) game, subject to a quadratic penalty that expresses her limited confidence in the global accuracy of the model. The natural generalization of this idea to the competitive case should then be given by the two players solving a local approximation of the true game, both subject to a quadratic penalty that expresses their limited confidence in the accuracy of the local approximation.
To implement this idea, we need to generalize the linear approximation in the single-agent setting to the competitive setting: *How to linearize a game?*

Linear or Multilinear. GDA answers the above question by choosing a linear approximation of $f, g : \mathbb{R}^m \times \mathbb{R}^n \longrightarrow \mathbb{R}$. This seemingly natural choice has the flaw that linear functions cannot express any interaction between the two players and are thus unable to capture the competitive nature of the underlying problem. From this point of view, it is not surprising that the convergent modifications of GDA are, implicitly or explicitly, based on higher-order approximations (see also [149]). An equally valid generalization of the linear approximation in the single-player setting is to use a *bilinear* approximation in the two-player setting. Since the bilinear approximation is the lowest order approximation that can capture some interaction between the two players, we argue that the natural generalization of gradient descent to competitive optimization is not GDA, but rather the update rule $(x_{k+1}, y_{k+1}) = (x_k, y_k) + (x, y)$, where (x, y) is a Nash equilibrium of the game

$$\begin{aligned} &\min_{x \in \mathbb{R}^m} x^\top \nabla_x f + x^\top D_{xy}^2 f y + y^\top \nabla_y f + \frac{1}{2\eta} x^\top x \\ &\min_{y \in \mathbb{R}^n} y^\top \nabla_y g + y^\top D_{yx}^2 g x + x^\top \nabla_x g + \frac{1}{2\eta} y^\top y. \end{aligned} \tag{7.3}$$

Indeed, the (unique) Nash equilibrium of (7.3) can be computed in closed form.

Theorem 23. *Among all (possibly randomized) strategies with finite first moment, the only Nash equilibrium of the Game (7.3) is given by*

$$x = -\eta \left(\mathrm{Id} - \eta^2 D_{xy}^2 f D_{yx}^2 g \right)^{-1} \left(\nabla_x f - \eta D_{xy}^2 f \nabla_y g \right) \tag{7.4}$$

$$y = -\eta \left(\mathrm{Id} - \eta^2 D_{yx}^2 g D_{xy}^2 f \right)^{-1} \left(\nabla_y g - \eta D_{yx}^2 g \nabla_x f \right), \tag{7.5}$$

given that the matrix inverses in the above expression exist. [2]

Proof. Let X, Y be the randomized strategies of the two agents. By subtracting and adding $\mathbb{E}[X]^2/(2\eta), \mathbb{E}[Y]^2/(2\eta)$, and taking expectations, we rewrite the game as

$$\min_{\mathbb{E}[X]\in\mathbb{R}^m} \mathbb{E}[X]^\top \nabla_x f + \mathbb{E}[X]^\top D_{xy}^2 f\, \mathbb{E}[Y] + \mathbb{E}[Y]^\top \nabla_y f + \frac{1}{2\eta}\mathbb{E}[X]^\top \mathbb{E}[X] + \frac{1}{2\eta}\mathbb{V}\mathrm{ar}[X] \tag{7.6}$$

$$\min_{\mathbb{E}[Y]\in\mathbb{R}^n} \mathbb{E}[Y]^\top \nabla_y g + \mathbb{E}[Y]^\top D_{yx}^2 g\, \mathbb{E}[X] + \mathbb{E}[X]^\top \nabla_x g + \frac{1}{2\eta}\mathbb{E}[Y]^\top \mathbb{E}[Y] + \frac{1}{2\eta}\mathbb{V}\mathrm{ar}[Y]. \tag{7.7}$$

Thus, the objective value for both players can always be improved by decreasing the variance while keeping the expectation the same, meaning that the optimal value will always (and only) be achieved by a deterministic strategy. We can then replace the $\mathbb{E}[X], \mathbb{E}[Y]$ with x, y, set the derivative of the first expression with respect to x and of the second expression with respect to y to zero, and solve the resulting system of two equations for the Nash equilibrium (x, y). □

According to Theorem 23, the Game (7.3) has exactly one optimal pair of strategies, which is deterministic. Thus, we can use these strategies as an update rule, generalizing the idea of local optimality from the single– to the multi-agent setting and obtaining Algorithm 16.

What I think that they think that I think ... that they do. Another game-theoretic interpretation of CGD follows from the observation that its update rule can be written as

$$\begin{pmatrix}\Delta x\\ \Delta y\end{pmatrix} = -\begin{pmatrix}\mathrm{Id} & \eta D_{xy}^2 f\\ \eta D_{yx}^2 g & \mathrm{Id}\end{pmatrix}^{-1}\begin{pmatrix}\nabla_x f\\ \nabla_y g\end{pmatrix}. \tag{7.8}$$

Applying the expansion $\lambda_{\max}(A) < 1 \Rightarrow (\mathrm{Id} - A)^{-1} = \lim_{N\to\infty}\sum_{k=0}^{N} A^k$ to the above equation, we observe that the first partial sum ($N = 0$) corresponds to the optimal strategy if the other player's strategy stays constant (GDA). The second partial sum ($N = 1$) corresponds to the optimal strategy if the other player thinks that the other player's strategy stays constant (LCGD, see Figure 7.3). The third partial sum ($N = 2$) corresponds to the optimal strategy if the other player thinks that the other player thinks that the other player's strategy stays constant, and so forth, until the Nash equilibrium is recovered in the limit (see Figure 7.2). For small

[2]We note that the matrix inverses exist for almost all values of η, and for all η in the case of a zero-sum game.

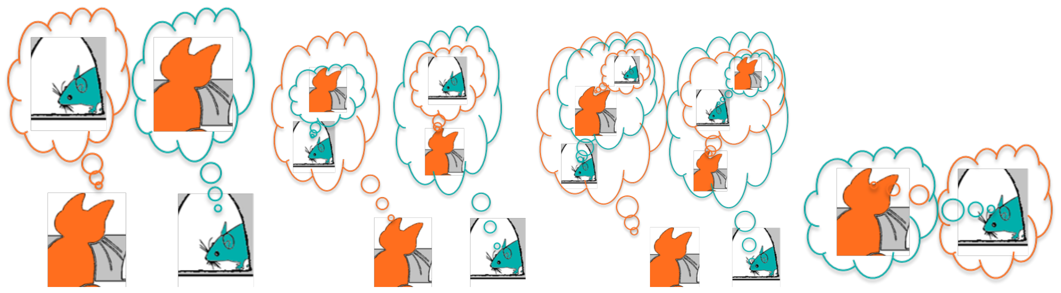


Figure 7.2: **What I think that they think that I think...** The partial sums of a Neumann-series representation of Equation (7.8) represent different orders of opponent-awareness, recovering the Nash-equilibrium in the limit.

enough η, we could use the above series expansion to solve for $(\Delta x, \Delta y)$, which is known as Richardson iteration and would recover high order lookahead [86]. However, expressing it as a matrix inverse will allow us to use optimal Krylov subspace methods to obtain far more accurate solutions with fewer gradient evaluations.

Rigorous results on convergence and local stability. We now present some basic convergence results for CGD, the proofs of which can be found in the appendix. Our results are restricted to the case of a zero-sum game ($f = -g$), but we expect that they can be extended to games that are dominated by competition. To simplify notation, we define

$$\bar{D} := (\mathrm{Id}+\eta^2 D^2_{xy} f D^2_{yx} f)^{-1} \eta^2 D^2_{xy} f D^2_{yx} f, \quad \tilde{D} := (\mathrm{Id}+\eta^2 D^2_{yx} f D^2_{xy} f)^{-1} \eta^2 D^2_{yx} f D^2_{xy} f. \tag{7.9}$$

We furthermore define the spectral function $\phi(\lambda) := 2\lambda - |\lambda|$.

Theorem 24. *If f is two times differentiable with L-Lipschitz continuous Hessian and the diagonal blocks of its Hessian are bounded as $\eta \|D^2_{xx} f\|, \eta \|D^2_{yy} f\| \leq 1/18$, we have*

$$\begin{aligned}
&\|\nabla_x f(x + x_k, y + y_k)\|^2 + \left\|\nabla_y f(x + x_k, y + y_k)\right\|^2 - \|\nabla_x f\|^2 - \|\nabla_y f\|^2 \\
\leq &- \nabla_x f^\top \left(2\eta h_\pm\left(D^2_{xx} f\right) + \frac{1}{3}\bar{D} - 32\eta^2 L\left(\|\nabla_x f\| + \|\nabla_y f\|\right) - 768\eta^4 L^2\right) \nabla_x f \\
&- \nabla_y f^\top \left(2\eta h_\pm\left(-D^2_{yy} f\right) + \frac{1}{3}\tilde{D} - 32\eta^2 L\left(\|\nabla_x f\| + \|\nabla_y f\|\right) - 768\eta^4 L^2\right) \nabla_y f.
\end{aligned}$$

Under suitable assumptions on the curvature of f, Theorem 24 implies results on the convergence of CGD.

Corollary 1. *Under the assumptions of Theorem 24, if for $\alpha > 0$*

$$\left(2\eta h_{\pm}\left(D_{xx}^2 f\right) + \frac{1}{3}\bar{D} - 32\eta^2 L\left(\|\nabla_x f(x_0, y_0)\| + \|\nabla_y f(x_0, y_0)\|\right) - 768\eta^4 L^2\right) \geq \alpha \mathrm{Id}$$
$$\left(2\eta h_{\pm}\left(-D_{yy}^2 f\right) + \frac{1}{3}\tilde{D} - 32\eta^2 L\left(\|\nabla_x f(x_0, y_0)\| + \|\nabla_y f(x_0, y_0)\|\right) - 768\eta^4 L^2\right) \geq \alpha \mathrm{Id}$$

for all $(x, y) \in \mathbb{R}^{m+n}$, then CGD started in (x_0, y_0) converges at exponential rate with exponent α to a critical point.

Furthermore, we can deduce the following local stability result.

Theorem 25. *Let (x^*, y^*) be a critical point ($(\nabla_x f, \nabla_y f) = (0, 0)$) and assume furthermore that*

$$\lambda_{\min} := \min\left(\lambda_{\min}\left(2\eta h_{\pm}\left(D_{xx}^2 f\right) + \frac{1}{3}\bar{D} - 768\eta^4 L^2\right), \lambda_{\min}\left(2\eta h_{\pm}\left(-D_{yy}^2 f\right) + \frac{1}{3}\tilde{D} - 768\eta^4 L^2\right)\right) > 0$$

and $f \in C^2(\mathbb{R}^{m+n})$ with Lipschitz continuous Hessian. Then there exists a neighborhood $\mathcal{U}$ of (x^*, y^*), such that CGD started in $(x_1, y_1) \in \mathcal{U}$ converges to a point in $\mathcal{U}$ at an exponential rate that depends only on $\lambda_{\min}$.

The results on local stability for existing modifications of GDA, including those of [61, 172, 173] (see also [154]) all require the stepsize to be chosen inversely proportional to an upper bound on $\sigma_{\max}(D_{xy}^2 f)$, and indeed we will see in our experiments that the existing methods are prone to divergence under strong interactions between the two players (large $\sigma_{\max}(D_{xy}^2 f)$). In contrast to these results, our convergence results *only improve* as the interaction between the players becomes stronger.

Why not use $D_{xx}^2 f$ and $D_{yy}^2 g$? The use of a bilinear approximation that contains some, but not all second order terms is unusual and begs the question why we do not include the diagonal blocks of the Hessian in Equation (7.8) resulting in the damped and regularized Newton's method

$$\begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = -\begin{pmatrix} \mathrm{Id} + \eta D_{xx}^2 f & \eta D_{xy}^2 f \\ \eta D_{yx}^2 g & \mathrm{Id} + \eta D_{yy}^2 g \end{pmatrix}^{-1} \begin{pmatrix} \nabla_x f \\ \nabla_y g \end{pmatrix}. \tag{7.10}$$

For the following reasons, we believe that the bilinear approximation is preferable both from a practical and conceptual point of view.

• *Conditioning of matrix inverse:* One advantage of competitive gradient descent is that in many cases, including all zero-sum games, the condition number of the matrix inverse in Algorithm 16 is bounded above by $\eta^2 \|D_{xy}\|^2$. If we include the diagonal blocks of the Hessian in a non-convex-concave problem, the matrix can even be singular as soon as $\eta \|D_{xx}^2 f\| \geq 1$ or $\eta \|D_{yy}^2 g\| \geq 1$.

• *Irrational updates:* We can only expect the update rule (7.10) to correspond to a local Nash equilibrium if the problem is convex-concave or $\eta \|D_{xx}^2 f\|, \eta \|D_{yy}^2 g\| < 1$. If these conditions are violated, it can instead correspond to the players playing their *worst* as opposed to best strategy based on the quadratic approximation, leading to behavior that contradicts the game-interpretation of the problem.

• *Lack of regularity:* For the inclusion of the diagonal blocks of the Hessian to be helpful at all, we need to make additional assumptions on the regularity of f, for example by bounding the Lipschitz constants of $D_{xx}^2 f$ and $D_{yy}^2 g$. Otherwise, their value at a given point can be totally uninformative about the global structure of the loss functions (consider as an example the minimization of $x \mapsto x^2 + \epsilon^{3/2} \sin(x/\epsilon)$ for $\epsilon \ll 1$). Many problems in competitive optimization, including GANs, have the form $f(x, y) = \Phi(\mathcal{G}(x), \mathcal{D}(y)), g(x, y) = \Theta(\mathcal{G}(x), \mathcal{D}(y))$, where Φ, Θ are *smooth* and *simple*, but $\mathcal{G}$ and $\mathcal{D}$ might only have first order regularity. In this setting, the bilinear approximation has the advantage of fully exploiting the first order information of $\mathcal{G}$ and $\mathcal{D}$, without assuming them to have higher order regularity. This is because the bilinear approximations of f and g then contains only the first derivatives of $\mathcal{G}$ and $\mathcal{D}$, while the quadratic approximation contains the second derivatives $D_{xx}^2 \mathcal{G}$ and $D_{yy}^2 \mathcal{D}$ and therefore needs stronger regularity assumptions on $\mathcal{G}$ and $\mathcal{D}$ to be effective.

• *No spurious symmetry:* One reason to favor full Taylor approximations of a certain order in single-player optimization is that they are invariant under changes of the coordinate system. For competitive optimization, a change of coordinates of $(x, y) \in \mathbb{R}^{m+n}$ can correspond, for instance, to taking a decision variable of one player and giving it to the other player. This changes the underlying game significantly and thus we do *not* want our approximation to be invariant under this transformation. Instead, we want our local approximation to only be invariant to coordinate changes of $x \in \mathbb{R}^m$ and $y \in \mathbb{R}^n$ *in separation*, that is to block-diagonal coordinate changes on $\mathbb{R}^{m+n}$. *Mixed* order approximations (bilinear, biquadratic, etc.) have exactly this invariance property and thus are the natural approximation for two-player games.

While we are convinced that the right notion of first order competitive optimization is given by quadratically regularized bilinear approximations, we believe that the right notion of second order competitive optimization is given by *cubically* regularized *biquadratic* approximations, in the spirit of [182].

7.3 Consensus, optimism, or competition?

We will now show that many of the convergent modifications of GDA correspond to different subsets of four common ingredients. *Consensus optimization* (ConOpt) [173], penalises the players for non-convergence by adding the squared norm of the gradient at the next location, $\gamma \|\nabla_x f(x_{k+1}, y_{k+1}), \nabla_x f(x_{k+1}, y_{k+1})\|^2$ to both player's loss function (here $\gamma \geq 0$ is a hyperparameter). As we see in Figure 7.3, the resulting gradient field has two additional Hessian corrections. [24, 147] observe that any game can be written as the sum of a *potential game* (that is easily solved by GDA), and a *Hamiltonian game* (that is easily solved by ConOpt). Based on this insight, they propose *symplectic gradient adjustment* that applies (in its simplest form) ConOpt only using the skew-symmetric part of the Hessian, thus alleviating the problematic tendency of ConOpt to converge to spurious solutions. The same algorithm was independently discovered by [89], who also provide a detailed analysis in the case of linear-quadratic GANs.
[61] proposed to modify GDA as

$$\Delta x = -\left(\nabla_x f(x_k, y_k) + (\nabla_x f(x_k, y_k) - \nabla_x f(x_{k-1}, y_{k-1}))\right) \tag{7.11}$$

$$\Delta y = -\left(\nabla_y g(x_k, y_k) + \left(\nabla_y g(x_k, y_k) - \nabla_y g(x_{k-1}, y_{k-1})\right)\right), \tag{7.12}$$

which we will refer to as optimistic gradient descent ascent (OGDA). By interpreting the differences appearing in the update rule as finite difference approximations to Hessian vector products, we see that (to leading order) OGDA corresponds to yet another second order correction of GDA (see Figure 7.3). It will also be instructive to compare the algorithms to linearized competitive gradient descent (LCGD), which is obtained by skipping the matrix inverse in CGD (which corresponds to taking only the leading order term in the limit $\eta D_{xy}^2 f \to 0$) and also coincides with first order lookahead [86]. As illustrated in Figure 7.3, these six algorithms amount to different subsets of the following four terms.

1. The *gradient term* $-\nabla_x f$, $\nabla_y f$ which corresponds to the most immediate way in which the players can improve their cost.

$$
\begin{array}{lllllll}
\text{GDA:} & \Delta x = & & -\nabla_x f & & & \\
\text{LCGD:} & \Delta x = & & -\nabla_x f & -\eta D_{xy}^2 f \nabla_y f & & \\
\text{SGA:} & \Delta x = & & -\nabla_x f & -\gamma D_{xy}^2 f \nabla_y f & & \\
\text{ConOpt:} & \Delta x = & & -\nabla_x f & -\gamma D_{xy}^2 f \nabla_y f & -\gamma D_{xx}^2 f \nabla_x f & \\
\text{OGDA:} & \Delta x \approx & & -\nabla_x f & -\eta D_{xy}^2 f \nabla_y f & +\eta D_{xx}^2 f \nabla_x f & \\
\text{CGD:} & \Delta x = & \left(\mathrm{Id} + \eta^2 D_{xy}^2 f D_{yx}^2 f\right)^{-1} & \big(-\nabla_x f & -\eta D_{xy}^2 f \nabla_y f & & \big)
\end{array}
$$

Figure 7.3: **Comparison.** The update rules of the first player for (from top to bottom) GDA, LCGD, ConOpt, OGDA, and CGD, in a zero-sum game ($f = -g$).

2. The *competitive term* $-D_{xy} f \nabla_y f$, $D_{yx} f \nabla_x f$ which can be interpreted either as anticipating the other player to use the naive (GDA) strategy, or as decreasing the other players influence (by decreasing their gradient).

3. The *consensus term* $\pm D_{xx}^2 \nabla_x f$, $\mp D_{yy}^2 \nabla_y f$ that determines whether the players prefer to decrease their gradient ($\pm = +$) or to increase it ($\pm = -$). The former corresponds the players seeking consensus, whereas the latter can be seen as the opposite of consensus. It also corresponds to an approximate Newton's method.[3]

4. The *equilibrium term* $(\mathrm{Id} + \eta^2 D_{xy}^2 D_{yx}^2 f)^{-1}$, $(\mathrm{Id} + \eta^2 D_{yx}^2 D_{xy}^2 f)^{-1}$, which arises from the players solving for the Nash equilibrium. This term lets each player prefer strategies that are less vulnerable to the actions of the other player.

Each of these is responsible for a different feature of the corresponding algorithm, which we can illustrate by applying the algorithms to three prototypical test cases considered in previous works.

• We first consider the bilinear problem $f(x, y) = \alpha xy$ (see Figure 7.4). It is well known that GDA will fail on this problem, for any value of η. For $\alpha = 1.0$, all the other methods converge exponentially towards the equilibrium, with ConOpt and SGA converging at a faster rate due to the stronger gradient correction ($\gamma > \eta$). If we choose $\alpha = 3.0$, OGDA, ConOpt, and SGA fail. The former diverges, while the latter two oscillate widely. If we choose $\alpha = 6.0$, all methods but CGD diverge.

[3]Applying a damped and regularized Newton's method to the problem of Player 1 would amount to choosing $x_{k+1} = x_k - \eta(\mathrm{Id} + \eta D_{xx}^2)^{-1} f \nabla_x f \approx x_k - \eta(\nabla_x f - \eta D_{xx}^2 f \nabla_x f)$, for $\|\eta D_{xx}^2 f\| \ll 1$.

- In order to explore the effect of the consensus Term 3, we now consider the convex-concave problem $f(x, y) = \alpha(x^2 - y^2)$ (see Figure 7.5). For $\alpha = 1.0$, all algorithms converge at an exponential rate, with ConOpt converging the fastest, and OGDA the slowest. The consensus promoting term of ConOpt accelerates convergence, while the competition promoting term of OGDA slows down the convergence. As we increase α to $\alpha = 3.0$, the OGDA and ConOpt start failing (diverge), while the remaining algorithms still converge at an exponential rate. Upon increasing α further to $\alpha = 6.0$, all algorithms diverge.

- We further investigate the effect of the consensus Term 3 by considering the concave-convex problem $f(x, y) = \alpha(-x^2 + y^2)$ (see Figure 7.5). The critical point $(0, 0)$ does not correspond to a Nash-equilibrium, since both players are playing their *worst possible strategy*. Thus it is highly undesirable for an algorithm to converge to this critical point. However for $\alpha = 1.0$, ConOpt does converge to $(0, 0)$ which provides an example of the consensus regularization introducing spurious solutions. The other algorithms, instead, diverge away towards infinity, as would be expected. In particular, we see that SGA is correcting the problematic behavior of ConOpt, while maintaining its better convergence rate in the first example. As we increase α to $\alpha \in \{3.0, 6.0\}$, the radius of attraction of $(0, 0)$ under ConOpt decreases and thus ConOpt diverges from the starting point $(0.5, 0.5)$, as well.

The first experiment shows that the inclusion of the competitive Term 2 is enough to solve the cycling problem in the bilinear case. However, as discussed after Theorem 24, the convergence results of existing methods in the literature are not break down as the interactions between the players becomes too strong (for the given η). The first experiment illustrates that this is not just a lack of theory, but corresponds to an actual failure mode of the existing algorithms. The experimental results in Figure 7.7 further show that for input dimensions $m, n > 1$, the advantages of CGD can not be recovered by simply changing the stepsize η used by the other methods.
While introducing the competitive term is enough to fix the cycling behaviour of GDA, OGDA and ConOpt (for small enough η) add the additional consensus term to the update rule, with opposite signs.
In the second experiment (where convergence is desired), OGDA converges in a smaller parameter range than GDA and SGA, while only diverging slightly faster in the third experiment (where divergence is desired).
ConOpt, on the other hand, converges faster than GDA in the second experiment, for

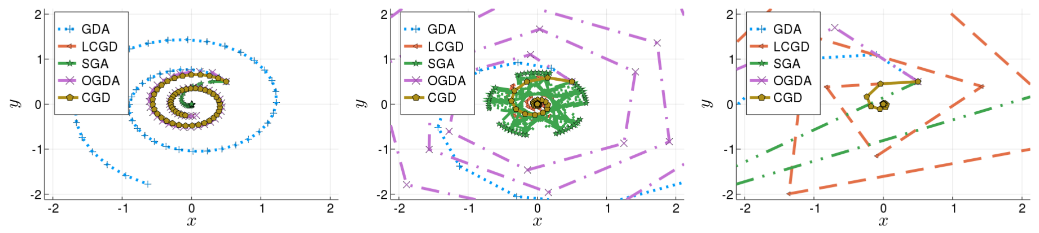


Figure 7.4: **The bilinear problem.** The first 50 iterations of GDA, LCGD, ConOpt, OGDA, and CGD with parameters $\eta = 0.2$ and $\gamma = 1.0$. The objective function is $f(x,y) = \alpha x^{\top} y$ for, from left to right, $\alpha \in \{1.0, 3.0, 6.0\}$. (Note that ConOpt and SGA coincide on a bilinear problem)

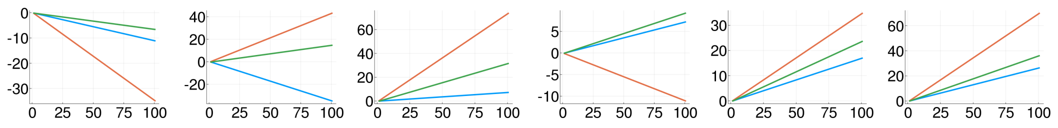


Figure 7.5: **A separable problem.** We measure the (non-)convergence to equilibrium in the separable convex-concave ($f(x,y) = \alpha(x^2 - y^2)$, left three plots) and concave-convex problem ($f(x,y) = \alpha(-x^2 + y^2)$, right three plots), for $\alpha \in \{1.0, 3.0, 6.0\}$. (Color coding given by GDA, SGA, LCGD, CGD, ConOpt, OGDA, the y-axis measures $\log_{10}(\|(x_k, y_k)\|)$ and the x-axis the number of iterations k. Note that convergence is desired for the first problem, while *divergence* is desired for the second problem.

$\alpha = 1.0$ however, it diverges faster for the remaining values of α and, what is more problematic, it converges to a spurious solution in the third experiment for $\alpha = 1.0$. Based on these findings, the consensus term with either sign does not seem to systematically improve the performance of the algorithm, which is why we suggest to only use the competitive term (that is, use lookahead/LCGD, or CGD, or SGA).

7.4 Implementation and numerical results

We briefly discuss the implementation of CGD.

Computing Hessian vector products. First, our algorithm requires products of the mixed Hessian $v \mapsto D_{xy} f v$, $v \mapsto D_{yx} g v$, which we want to compute using automatic differentiation. As was already observed by [195], Hessian vector products can be computed at minimal overhead over the cost of computing gradients, by combining forward– and reverse mode automatic differentiation. To this end, a function $x \mapsto \nabla_y f(x,y)$ is defined using reverse mode automatic differentiation. The Hessian vector product can then be evaluated as $D^2_{xy} f v = \frac{\partial}{\partial h} \nabla_y f(x + hv, y)\big|_{h=0}$, using forward mode automatic differentiation. Many AD frameworks, like Autograd (`https://github.com/HIPS/autograd`) and ForwardDiff(`https://github.com/JuliaDiff/ForwardDiff.jl`, [204]) together with ReverseDiff(`https://github.com/JuliaDiff/ReverseDiff.jl`) support this procedure. In settings where we are only given access to gradient evaluations but cannot use automatic differentiation to compute Hessian vector products, we can instead approximate them using finite differences.

Matrix inversion for the equilibrium term. Similar to a *truncated Newton's method* [185], we propose to use iterative methods to approximate the inverse-matrix vector products arising in the equilibrium term 4. We will focus on zero-sum games, where the matrix is always symmetric positive definite, making the conjugate gradient (CG) algorithm the method of choice. For nonzero sum games we recommend using the GMRES or BCGSTAB (see for example [210] for details). We suggest terminating the iterative solver after a given relative decrease of the residual is achieved ($\|Mx - y\| \leq \epsilon \|x\|$ for a small parameter ϵ, when solving the system $Mx = y$). In our experiments we choose $\epsilon = 10^{-6}$. Given the strategy Δx of one player, Δy is the optimal counter strategy which can be found without solving another system of equations. Thus, we recommend in each update to only solve for the strategy of one of the two players using Equation (7.4), and then use the optimal counter strategy for the other player. The computational cost can be further improved by using the last round's optimal strategy as a a warm start of the inner CG solve. An appealing feature of the above algorithm is that the number of iterations of CG adapts to the difficulty of solving the equilibrium term 4. If it is easy, we converge rapidly and CGD thus *gracefully reduces to LCGD*, at only a small overhead. If it is difficult, we might need many iterations, but correspondingly the problem would be very hard without the preconditioning provided by the equilibrium term.

Experiment: Fitting a bimodal distribution. We use a simple GAN to fit a Gaussian mixture model with two modes, in two dimensions (see supplement for details). We apply SGA, ConOpt ($\gamma = 1.0$), OGDA, and CGD for stepsize $\eta \in \{0.4, 0.1, 0.025, 0.005\}$ together with RMSProp ($\rho = 0.9$). In each case, CGD produces an reasonable approximation of the input distribution without any mode collapse. In contrast, all other methods diverge after some initial cycling behavior! Reducing the steplength to $\eta = 0.001$, did not seem to help, either. While we believe that the other methods can be made work with more hyperparameter tuning or a change of the GAN loss function (we use the original, zero-sum GAN loss

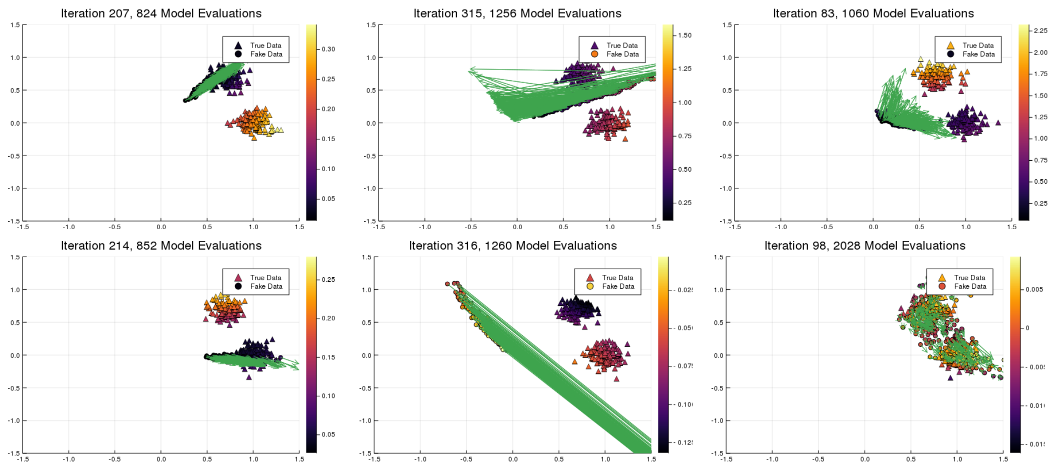


Figure 7.6: **Fitting a bimodal distribution.** For all methods, initially the players cycle between the two modes (first column). For all methods but CGD, the dynamics eventually become unstable (middle column). Under CGD, the mass eventually distributes evenly among the two modes (right column). (The arrows show the update of the generator and the colormap encodes the logit output by the discriminator.)

proposed by [96]), this result substantiates our claim that CGD is significantly more robust than existing methods for competitive optimization. For more details and visualizations of the whole trajectories, consult the supplementary material.

Experiment: Estimating a covariance matrix. To show that CGD is also competitive in terms of computational complexity we consider the noiseless case of the covariance estimation example used by [61][Appendix C], We study the tradeoff between the number of evaluations of the forward model (thus accounting for the inner loop of CGD) and the residual and observe that for comparable stepsize, the convergence rate of CGD is similar to the other methods. However, due to CGD being convergent for larger stepsize it can beat the other methods by more than a factor two (see supplement for details).

Experiment: Model-Based reinforcement learning. Finally, we apply CGD to a constrained optimization problem arising in imitation learning under the linear quadratic assumption [14]. We minimize the Euclidean distance between the actions of an optimal LQR controller derived from the discrete time algebraic Riccati equation and the demonstrated actions. Our constrained optimization problem is therefore:

$$\begin{aligned}&\text{minimize} && \mathbb{E}\left[\|a_i - \pi_X(s_i)\|_2\right]\\ &\text{subject to} && A^\top X A - \left(A^\top X B\right)\left(R + B^\top X B\right)^{-1}(BXA) + Q = 0\end{aligned}$$

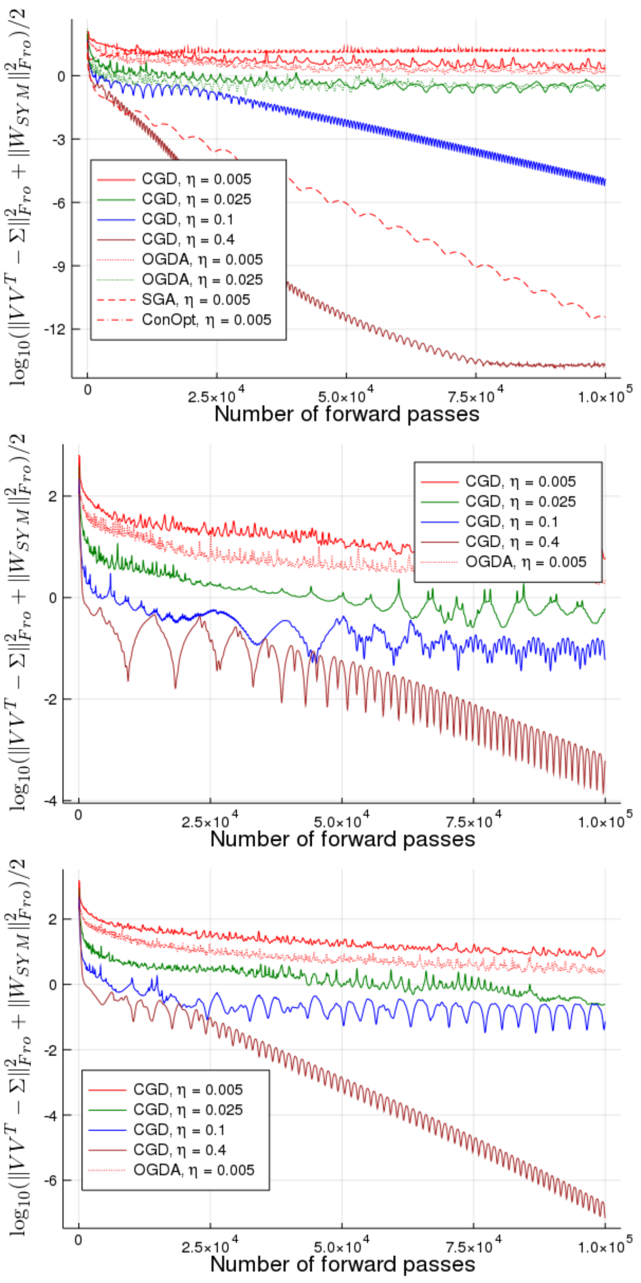


Figure 7.7: **Comparison of convergence speed.** We plot the decay of the residual after a given number of model evaluations, for increasing problem sizes and $\eta \in \{0.005, 0.025, 0.1, 0.4\}$. Experiments that are not plotted diverged.

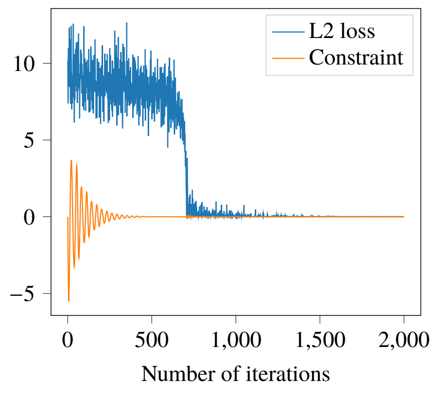


Figure 7.8: **A training run** (1 **seed).**

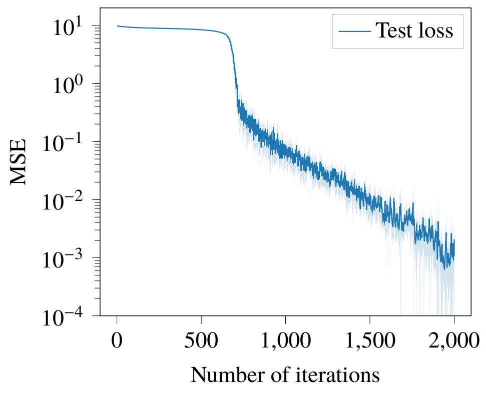


Figure 7.9: **Generalization** (40 **seeds).**

Figure 7.10: **Imitation learning experiment in the cartpole domain.** On the left, we show a single training run, and on the right, we plot the averaged loss over 40 random seeds.

where $\pi_X(s_i) := -(R + B^\top XB)^{-1}(BXA)s_i$ and the expectation is taken with respect to a distribution over demonstrations. We estimate the expectation by querying an optimal LQR policy at 1000 random states sampled around the equilibrium. By introducing Lagrange multipliers, we turn this problem into an unconstrained competitive optimization problem and solve it using CGD. For more details, we refer to [22], where this experiment is taken from. Figure 7.8 shows the joint evolution of the player behind the imitation loss and its opponent trying to satisfy the constraint. We attribute the initial plateauing of the "imitation loss player" to the need of first roughly satisfying the constraint before the loss can be systematically improved. Once the feasible set has been approached, the loss drops quickly as the direction of improvement becomes easier to identify. Figure 7.9 measures the generalization loss over an independent dataset of 500 states sampled at random around the equilibrium. We report the test performance across 40 random seeds and compute 99% confidence intervals. By viewing the generalization plot on a log scale, we see that our competitive differentiation converges linearly to a solution once it overcomes the initial plateau.

Chapter 8

COMPETITIVE MIRROR DESCENT

Constrained competitive optimization. In Chapter 7, we have introduced competitive gradient descent (CGD) as a general purpose algorithm for unconstrained competitive optimization. However, many appplications in machine learning and beyond require the use of constraints, for instance to encode physical laws [13, 53, 200, 229], safety requirements or optimality conditions in reinforcement learning [2, 22, 175], or fairness requirements [54, 55, 178], leading to *constrained competitive optimization* problems of the general form

$$\min_{\substack{x \in \bar{C}, \\ \tilde{f}(x,y) \in \tilde{C}}} \bar{f}(x,y), \qquad \min_{\substack{y \in \bar{\mathcal{K}}, \\ \tilde{g}(x,y) \in \tilde{\mathcal{K}}}} \bar{g}(x,y), \tag{8.1}$$

where $\bar{C}, \bar{\mathcal{K}}$ are convex sets and $\tilde{\mathcal{K}}, \tilde{C}$ are convex cones while $\tilde{f}, tg, \bar{f}, \bar{g}$ are possibly nonconvex functions. The goal of this work is to extend CGD to a general-purpose algorithm for solving constrained competitive problems as a counterpart of gradient descent in unconstrained single-agent optimization.

Constraints and competition. At a closer look, constraints and competition are closely related. For (in)equality constrained problems, we can use Lagrange multiplier λ, μ to turn a constrained minimization problem into an unconstrained minmax problem

$$\min_{x\colon\, g(x)=0, h(x)\geq 0} f(x) \quad \Leftrightarrow \quad \min_{x} \max_{\lambda,\mu} f(x) + \lambda g(x) + \mu h(x).$$

By simultaneously optimizing over x and λ, we obtain an unconstrained competitive optimization problem that approximates the original constrained problem and, under certain conditions, recovers it. We can use an analogue procedure to turn Problem 8.1 by the simplified problem

$$\min_{x \in C} f(x,y), \quad \min_{y \in \mathcal{G}} g(x,y) \tag{8.2}$$

for suitably chosen convex sets C and $\mathcal{G}$, and (in general) nonconvex functions f and g.

Competitive gradient descent (CGD). The simplest approach to solving a min-max problem is simultaneous gradient descent (SimGD), where both players perform a step of gradient descent (ascent for the dual player) at each iteration. However, this algorithm has poor performance in practice and diverges even on simple problems. We argued in Chapter 7 that the poor performance of SimGD comes from the fact that the updates of the two players are computed without taking the interactive nature of the game into account. They propose to instead compute both players' updates as the Nash equilibrium of a local bilinear approximation of the original game, regularized with a quadratic penalty that models the limited confidence of both players in the approximation's accuracy. The resulting algorithm, *competitive gradient descent (CGD)*, is a promising candidate for solving constrained competitive optimization problems. The simplest approach for extending CGD constrained problems is to interleave its updates with projections onto the constraint set, obtaining projected gradient descent (PCGD). However, this algorithm converges to suboptimal points even in simple convex examples due to a phenomenon that we call *empty threats*.

Mirror descent. The fundamental reason for the *empty threats*-phenomenon observed in projected CGD is that the local update rule of CGD does not include any information about the global structure of the constraint set. The *mirror descent* framework [180] uses a *Bregman potential* ψ to obtain a local update rule that takes the global geometry of the constraint set into account. It computes the next iterate x_{k+1} as the minimizer of a linear approximation of the objective, regularized with the *Bregman divergence* $\mathbb{D}_{\psi}(x_{k+1} \parallel x_k)$ associated to ψ. For problems on the positive orthant $\mathbb{R}^m_+$, we can ensure feasible iterates by choosing ψ as the Shannon entropy, obtaining an algorithm known as *entropic mirror descent*, *exponentiated gradient descent*, or *multiplicative weights algorithm*. A naive extension of mirror descent to the competitive setting simply replaces the quadratic regularization in SimGD or CGD with a Bregman divergence. However, the former inherits the cycling problem of SimGD and the latter requires solving a nonlinear problem *at each iteration*.

Our contributions. In this chapter, we propose competitive mirror descent (CMD), a novel algorithm that combines the ideas of CGD and mirror descent in a computationally efficient way. In the special case where the Bregman potential is the Shannon entropy, our algorithm computes the Nash-equilibrium of a regularized bilinear approximation and uses it for a *multiplicative* update. The resulting *competitive multiplicative weights (CMW, Algorithm 17)* is a competitive extension of

Algorithm 17 Competitive multiplicative weights (CMW)[1]

```
1: for $0 \leq k < N$ do
2:     $\Delta x = -\left(\operatorname{diag}_{x_k} - \left[D^2_{xy} f\right] \operatorname{diag}^{-1}_{y_k} \left[D^2_{yx} g\right]\right)^{-1} \left(\left[\nabla_x f\right] - \left[D^2_{xy} f\right] \operatorname{diag}^{-1}_{y_k} \left[\nabla_y g\right]\right)$
3:     $\Delta y = -\left(\operatorname{diag}_{y_k} - \left[D^2_{yx} g\right] \operatorname{diag}^{-1}_{x_k} \left[D^2_{xy} f\right]\right)^{-1} \left(\left[\nabla_x g\right] - \left[D^2_{yx} g\right] \operatorname{diag}^{-1}_{x_k} \left[\nabla_x f\right]\right)$
4:     $x_{k+1} = x_k \exp(\Delta x)$
5:     $y_{k+1} = y_k \exp(\Delta y)$
6: end for
7: return $x_N, y_N$
```

Algorithm 18 Competitive mirror descent (CMD) for general Bregman potentials ψ and ϕ.

```
1: for $0 \leq k < N$ do
2:     $\Delta x = -\left(\left[D^2 \psi\right] - \left[D^2_{xy} f\right] \left[D^2 \phi\right]^{-1} \left[D^2_{yx} g\right]\right)^{-1} \left(\left[\nabla_x f\right] - \left[D^2_{xy} f\right] \left[D^2 \phi\right]^{-1} \left[\nabla_y g\right]\right)$
3:     $\Delta y = -\left(\left[D^2 \phi\right] - \left[D^2_{yx} g\right] \left[D^2 \psi\right]^{-1} \left[D^2_{xy} f\right]\right)^{-1} \left(\left[\nabla_x g\right] - \left[D^2_{yx} g\right] \left[D^2 \psi\right]^{-1} \left[\nabla_x f\right]\right)$
4:     $x_{k+1} = (\nabla \psi)^{-1} \left(\left[\nabla \psi(x)\right] + \Delta x\right)$
5:     $y_{k+1} = (\nabla \phi)^{-1} \left(\left[\nabla \phi(y)\right] + \Delta y\right)$
6: end for
7: return $x_N, y_N$
```

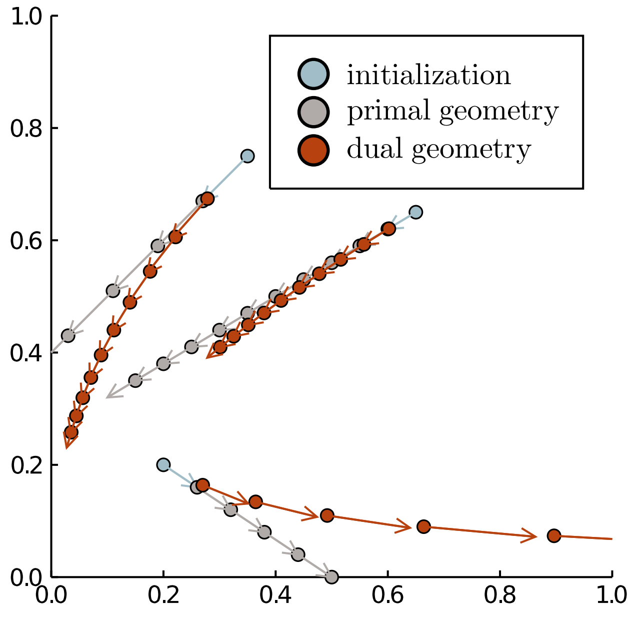


Figure 8.1: **Dual geometry.** The dual notion of straight line induced by the Shannon entropy guarantees feasible iterates on $\mathbb{R}^m_+$.

the multiplicative weights update rule that accounts for the interaction between the two players.

More generally, our method is based on the geometric interpretation of mirror descent proposed by [202]. From this point of view, mirror descent solves a quadratic local problem in order to obtain a *direction* of movement. The next iterate is then obtained by moving into this direction for a unit time interval. Crucially, this notion of *moving into a direction* is not derived from the Euclidean structure of the constraint set, but from the dual geometry defined by the Bregman potential. In the case of the Shannon entropy, this amounts to moving on straight lines in logarithmic coordinates, resulting in multiplicative updates (see Figure 8.1). This formulation is extended to CGD by letting both agents choose a direction of movement according to a local bilinear approximation of the original problem and then using the dual geometry of the Bregman potential to derive the next iterate. The resulting *competitive mirror descent (CMD, Algorithm 18)* combines the computational efficiency of CGD with the ability to use general Bregman potentials to account for the nonlinear structure of the constraints.

By presenting a series of examples from game theory, statistical estimation, and robust reinforcement learning, we establish the practical performance of CGM and show that it inherits the benefits of both mirror descent and competitive gradient descent.

8.1 Simplifying constraints by duality

Constrained competitive optimization. The most general class of problems that we are concerned with is of the form of (8.1) where $\bar{\mathcal{C}} \subset \mathbb{R}^m, \bar{\mathcal{K}} \in \mathbb{R}^n$ are convex sets, $\tilde{\mathcal{C}} \subset \mathbb{R}^{\tilde{n}}, \tilde{\mathcal{K}} \subset \mathbb{R}^{\tilde{m}}$ are closed convex cones, and $\bar{f} : \bar{\mathcal{C}} \longrightarrow \mathbb{R}, \bar{g} : \bar{\mathcal{K}} \longrightarrow \mathbb{R}, \tilde{f} : \tilde{\mathcal{C}} \longrightarrow \mathbb{R}^{\tilde{n}}$, and $\tilde{g} : \mathcal{G} \longrightarrow \mathbb{R}^{\tilde{m}}$ are continuous and piecewise differentiable multivariate functions of the two agents' decision variables x and y. This framework is highly general given suitable functions $\tilde{f}, \tilde{g}$, and convex cones $\tilde{\mathcal{C}}, \tilde{\mathcal{K}}$, it can implement a variety of nonlinear equality, inequality, and positive-definiteness constraints. While there are many ways in which a problem can be cast into the above form, we are interested in the case where the $f, \tilde{f}, g, \tilde{g}$ are allowed to be *complicated*, for instance, given in terms of neural networks, while the $\bar{\mathcal{C}}, \bar{\mathcal{K}}, \tilde{\mathcal{C}}, \tilde{\mathcal{K}}$ are simple and well-understood. For convex constraints and objectives the canonical solution

[1]Here, exp is applied element-wise and diag_z denotes the diagonal matrix with entries given by the vector z. $\nabla_x f$, $[D^2_{xy} f]$, $\nabla_y g$, $[D^2_{yx} g]$ denote the gradients and mixed Hessians of f and g, evaluated in (x_k, y_k).

concept is a *Nash equilibrium*, a pair of feasible strategies $(\bar{x}, \bar{y})$ such that $\bar{x}$ $(\bar{y})$ is the optimal strategy for x (y) given $y = \bar{y}$ $(x = \bar{x})$. In the non-convex case, it is less clear what should constitute a solution. In Chapter 9, we will argue that meaningful solutions need not even be local Nash equilibria.

Lagrange multipliers lead to linear constraints. Using the classical technique of Lagrangian duality, the complicated parameterization $f, \tilde{f}, g, \tilde{g}$ and the simple constraints given by the $\tilde{\mathcal{C}}, \tilde{\mathcal{K}}$ can be further decoupled. The polar of a convex cone $\mathcal{G}$ is defined as $\mathcal{G}^{\circ} \coloneqq \left\{y : \sup_{x \in \mathcal{G}} x^{\top} y \leq 0\right\}$. Using this definition, we can rewrite Problem 8.1 as

$$\min_{\substack{x \in \bar{\mathcal{C}}, \\ \mu \in \tilde{\mathcal{K}}^{\circ}}} \bar{f}(x, y) + \max_{\nu \in \tilde{\mathcal{C}}^{\circ}} \nu^{\top} \tilde{f}(x), \quad \min_{\substack{y \in \bar{\mathcal{K}}, \\ \nu \in \tilde{\mathcal{C}}^{\circ}}} g(x, y) + \max_{\mu \in \tilde{\mathcal{K}}^{\circ}} \mu^{\top} \tilde{g}(y). \tag{8.3}$$

Here we used the fact that the maxima are infinity if any constraint is violated and zero, otherwise.

Watchmen watching watchmen. We can now attempt to simplify the problem by making μ_j (ν_i) decision variables of the y (x) player and adding a zero sum objective to the game that incentivizes both players to enforce each other's compliance with the constraints, resulting in

$$\min_{\substack{x \in \mathcal{C}, \\ \mu \in \tilde{\mathcal{K}}^{\circ}}} f(x, y) + \nu^{\top} \tilde{f}(x) - \mu^{\top} \tilde{g}(y), \quad \min_{\substack{y \in \mathcal{G}, \\ \nu \in \mathcal{C}^{\circ}}} g(x, y) + \mu^{\top} \tilde{g}(y) - \nu^{\top} \tilde{f}(x). \tag{8.4}$$

If Problem 8.1 is convex and strictly feasible (Slater's condition), its Nash equilibria are equal to those of Problem 8.4 (see Section .5.1 in the Appendix for details).

A simplified problem. While this is not true in general, we propose to use Problem 8.4 as a more tractable approximation of Problem 8.1. In the following, we assume that the nonlinear constraints of the problem have already been eliminated, leaving us (for possible different $\mathcal{C}$ and $\mathcal{G}$) with

$$\min_{x \in \mathcal{C}} f(x, y), \quad \min_{y \in \mathcal{G}} g(x, y). \tag{8.5}$$

8.2 Projected CGD suffers from empty threats

Competitive gradient descent (CGD). In Chapter 7, we proposed to solve unconstrained competitive optimization problems by choosing iterates (x_{k+1}, y_{k+1}) as

Nash equilibria of a quadratically regularized bilinear approximation

$$
\begin{aligned}
x_{k+1} &= x_k + \operatorname*{arg\,min}_{x \in \mathbb{R}^m} [D_x f]\, x + y^\top \left[D_{yx} f\right] x + \left[D_y f\right] y + \frac{x^\top x}{2\eta} \\
y_{k+1} &= y_k + \operatorname*{arg\,min}_{y \in \mathbb{R}^n} [D_x g]\, x + x^\top \left[D_{xy} g\right] y + \left[D_y g\right] y + \frac{y^\top y}{2\eta},
\end{aligned}
\tag{8.6}
$$

where $f, g : \mathbb{R}^m \times \mathbb{R}^n \longrightarrow \mathbb{R}$ are the loss functions of the two agents and $[D_x f]$, $[D_x g]$, $\left[D_y f\right]$, $\left[D_y g\right]$, $\left[D_{yx} f\right]$, and $\left[D_{xy} g\right]$ their (mixed) derivatives evaluated in the last iterate (x_k, y_k).

A simple minmax game. We will use the following simple example to illustrate the difficulties when dealing with inequality constraints.

$$
\min_{x \in \mathbb{R}_+} 2xy - (1-y)^2 \quad \min_{y \in \mathbb{R}+} -2xy + (1-y)^2. \tag{8.7}
$$

How do we incorporate the positivity constraints into CGD? The simplest approach is to combine CGD with projections onto the constraint set, which we call *projected competitive gradient descent (PCGD)*. Here, we compute the updates of CGD as per usual, but after each update we project back onto the constraint set through $(x, y) \mapsto (\max(0, x), \max(0, y))$. This generalizes projected gradient descent, a popular method in constrained optimization with proven convergence properties [126].

PCGD and empty threats. Applying PCGD (with $\eta = 0.25$) to our example, we see that it converges instead to $(x, y) = (0, 2/3)$, which is suboptimal for the y-player! Since Problem 8.7 is a convex game, any sensible algorithm should converge to the unique Nash-equilibrium $(x, y) = (0, 1)$, making this a failure case for PCGD. The explanation of this behavior lies in the conflicting rules of the unconstrained local game 8.6 and the constrained global game 8.7. Under the local game of Equation (8.6), x can decrease its loss by turning negative. This is an *empty threat* in that it violates the constraints and will therefore be undone by the projection step, but this information is not included in the local game. Therefore, the outlook of x becoming negative deters y from moving towards the optimal strategy of $y = 1$ (c.f. Figure 8.2). This problem affects the projected versions of most algorithms featuring *competitive terms* involving the mixed Hessian. Since in Chapter 7 we identified these terms as crucial for convergence, this is a major obstacle for constrained competitive optimization.

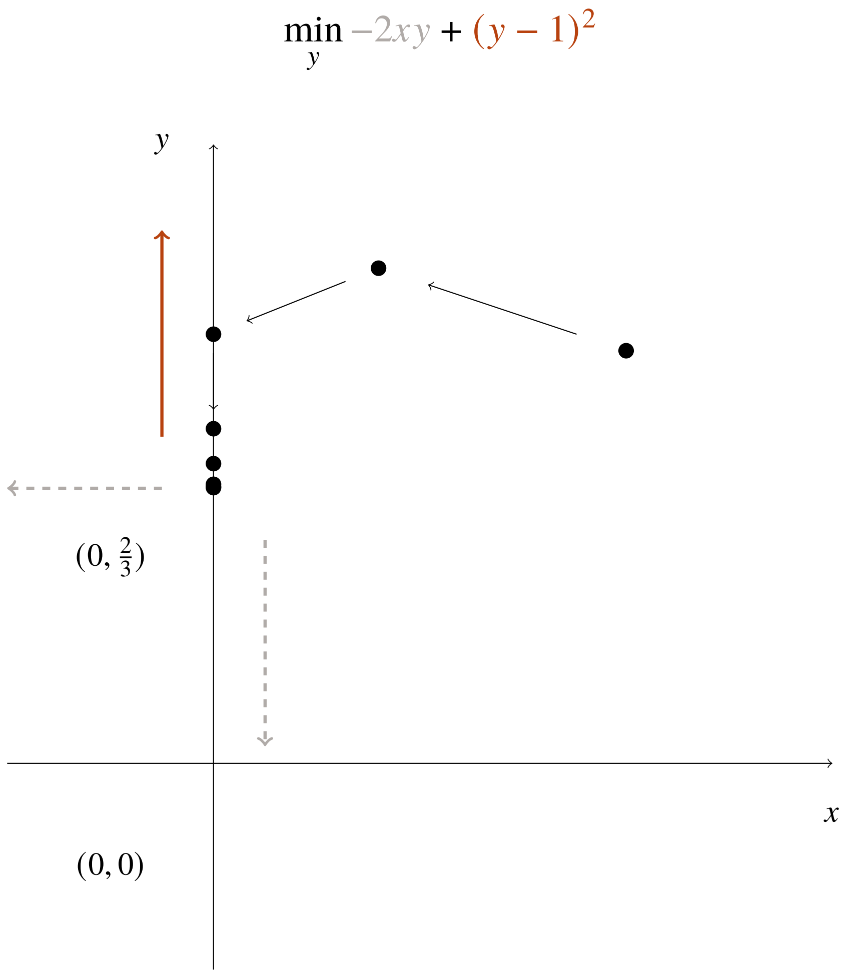


Figure 8.2: **Empty threats.** Since $x \geq 0$, the bilinear term should only lead to y picking a larger value. But CGD is oblivious to the constraint and y decreases in anticipation of x turning negative.

8.3 Mirror descent and Bregman potentials

Bregman potentials. The underlying reason for the *empty threats-phenomenon* described in the last section is that a local method such as CGD has no way of knowing how close it is to the boundary of the constraint set. In single-agent optimization, this problem has been addressed by the use of *Bregman potentials* or *Barrier functions*.

Definition 13. *We call a strictly convex and two-times differentiable function ψ : $C \longrightarrow \mathbb{R}$ a* Bregman potential *on the convex domain C.*

Possibly the most widely used Bregman potentials are the squared distance $x^\top x$ on $\mathbb{R}^n$ and the negative Shannon entropy $\sum_i x_i \log(x_i)$ on the positive orthant $\mathbb{R}^n_+$. For the purposes of this work, a Bregman potential ψ on C is best understood as quantifying how close a point $y \in C$ is from $\arg\min \psi$, which we think of as the center of C. Importantly, the notion of distance induced by ψ is anisotropic and usually increases rapidly as y approaches the boundary of the domain. In particular, derivative information of ψ at a point p allows us to infer its position relative to the boundary of C, allowing a local algorithm to take into account the global structure of the constraint set.

Bregman divergences. Associated with a Bregman potential $\psi(\cdot)$ on a domain C is the Bregman divergence $\mathbb{D}_{\psi}(\cdot \parallel \cdot)$ that allows us to extend the notion of distance between y and $\arg\min \psi$ given by ψ to a notion of distance between arbitrary elements of C.

Definition 14. *The Bregman divergence associated to the potential ψ is defined as*

$$\mathbb{D}_{\psi}(p \parallel q) \coloneqq \psi(p) - \psi(q) - [\nabla \psi(q)](p - q). \tag{8.8}$$

Just like a squared distance, $\mathbb{D}_{\psi}(p \parallel q)$ is convex, positive, and achieves its minimum of zero if and only if $p = q$. However, it is not symmetric in the sense that $\mathbb{D}_{\psi}(p \parallel q) \neq \mathbb{D}_{\psi}(q \parallel p)$, in general. Crucially, a Bregman divergence will in general not be translation invariant and can therefore take the position relative to the boundary of C into account.

Mirror Descent. *Mirror descent* [180] uses a Bregman to improve gradient descent by the following update rule.

$$x_{k+1} = \arg\min_{x} \left[D f\left(x_k\right) \right] \left(x - x_k\right) + \mathbb{D}_{\psi}(x \parallel x_k). \tag{8.9}$$

By solving for the first order optimality conditions, the update can be computed in closed form as

$$x_{k+1} = \left(D\psi\right)^{-1} \left(\left[D\psi\left(x_k\right) \right] - \left[D f\left(x_k\right) \right] \right), \tag{8.10}$$

where $\left(D\psi\right)^{-1}(y)$ is defined as the point $x \in C$ such that $[D\psi(x)] = y$. By strict convexity of ψ, if $\left(D\psi\right)^{-1}(y)$ exists, it is unique. In this case, $\left(D\psi\right)^{-1}(y)$ is contained in C which ensures that mirror descent satisfies the constraints without an additional projection step.

Definition 15. *A Bregman potential ψ is* complete *on C if the map $D\psi : C \longrightarrow \mathbb{R}^{1 \times m}$ is surjective.*

If ψ is complete, the mirror descent update is well-defined for all gradients. From this point of view, the necessity for a projection step in inequality constrained gradient descent arises from the potential $\psi(x) = x^\top x/2$ not being complete on the constraint set C. A popular choice of potential that is complete on the positive orthant $\mathbb{R}^m_+$ is the Shannon entropy $\psi(x) \sum_i x_i \log(x_i) - x_i$ resulting in *entropic mirror descent*

$$x_{k+1} = \exp\left(\log(x_k) - [Df(x_k)]^\top\right) = x_k \exp\left([Df(x_k)]^\top\right). \tag{8.11}$$

Here, multiplication, exponentiation, and logarithms are defined component-wise.

Naive competitive mirror descent. A possible generalization of mirror descent to Problem 8.5 is to obtain x_{k+1} and y_{k+1} as solutions to the game

$$\begin{aligned}
&\underset{x \in \mathbb{R}^m}{\arg\min}\, [D_x f](x - x_k) + (y - y_k)^\top \left[D_{yx} f\right](x - x_k) + \left[D_y f\right](y - y_k) + \mathbb{D}_\psi(x \parallel x_k) \\
&\underset{y \in \mathbb{R}^n}{\arg\min}\, [D_x g](x - x_k) + (x - x_k)^\top \left[D_{xy} g\right](y - y_k) + \left[D_y g\right](y - y_k) + \mathbb{D}_\phi(y \parallel y_k),
\end{aligned}$$

where ψ and ϕ are complete Bregman potentials on C and $\mathcal{G}$, respectively. However, the local game in this update rule does not have a closed form solution as in mirror descent or competitive gradient descent. Instead, it requires us to solve a nonlinear competitive optimization problem at each step. While it is possible to alternatingly solve for the two players' strategies until convergence, this will be significantly slower than the Krylov subspace methods employed to solve system of linear equations in the CGD. In order to avoid this overhead, we will now develop an alternative competitive generalization of mirror descent rooted in its information-geometric interpretation.

8.4 The information geometry of Bregman divergences

Reminder on differential geometry. To present the material in this chapter, we briefly review the following basic notions of differential geometry.

Definition 16 (Submanifold of $\mathbb{R}^m$)**.** *$\mathcal{M} \subset \mathbb{R}^m$ is a k-dimensional smooth submanifold of $\mathbb{R}^m$ if for every point $p \in \mathcal{M}$ there exists an open ball $B(p)$ centered at p and a smooth function $G_p : B(p) \longrightarrow \mathbb{R}^{m-k}$, such that the rank of the Jacobian of G_p is k everywhere and $\mathcal{M} \cap B_p = G^{-1}(0)$.*

We can think of a smooth submanifold as a (possible curved) surface in $\mathbb{R}^m$.

Definition 17 (Tangent space)**.** *The tangent space $\mathcal{T}_p\mathcal{M}$ of a k-dimensional submanifold $\mathcal{M}$ of $\mathbb{R}^m$ in p is given by the null space of the Jacobian of G_p in p. Here, G_p is chosen as in Definition 16.*

To a creature living on $\mathcal{M}$, the elements of the tangent space in $p \in \mathcal{M}$ correspond to the velocities with which it could depart from p.

Definition 18 (Riemannian metric)**.** *A Riemannian metric is a map* $p \mapsto g_p(\cdot,\cdot)$ *that assigns to each* $p \in \mathcal{M}$ *an inner product on* $\mathcal{T}_p\mathcal{M}$.

If the elements $x \in \mathcal{T}_p\mathcal{M}$ of the tangent space are velocities, $g_p(x, x)$ is their (squared) speed. It allows us to compare the magnitude or significance of moving in different velocities.

Definition 19 (Exponential map)**.** *We call a collection* $\left\{\mathrm{Exp}_p : \mathcal{T}_p\mathcal{M} \longrightarrow \mathcal{M}\right\}_{p\in\mathcal{M}}$ *an* exponential map *if it satisfies*

$$\mathrm{Exp}_p\left((t+s)\,x\right) = \mathrm{Exp}_q\left(sy\right), \quad \text{for } q = \mathrm{Exp}_p(tx), \quad y = \frac{\mathrm{d}}{\mathrm{d}r}\,\mathrm{Exp}_p\left(rx\right)\big|_{r=t}.$$

The exponential map $\mathrm{Exp}_p(x)$ returns the destination reached when *walking straight* in the velocity x for a unit time interval, starting in p.

The geometry of Bregman potentials. We will now explain how a Bregman potential equips its domain C with a geometric structure (see [8, 9] for details). Our manifold $\mathcal{M}$ will simply be the interior of $C \subset \mathbb{R}^m$ with $\mathcal{T}_p\mathcal{M}$ given by $\mathbb{R}^m$ for all $p \in \mathcal{M}$ (without loss of generality, we assume that $\operatorname{int} C$ has full dimension). The Riemannian metric g^ψ associated with the potential ψ is given by

$$g^\psi_p\left(x, x\right) = \frac{x^\top\left[D^2_{xx}\psi\left(p\right)\right]x}{2} = \frac{\mathrm{d}^2}{2\,\mathrm{d}r^2}\mathbb{D}_\psi(p + rx \parallel p).$$

Following the interpretation of the Bregman divergence as a notion of squared distance, the metric g^ψ_p measures how quickly a given velocity $x \in \mathcal{T}_p\mathcal{M}$ will take us away from p, according to this notion of distance.

The dual exponential map. A key feature of Bregman potentials is that they induce a new exponential map on the manifold C. $\mathcal{M}$ is an open subset of $\mathbb{R}^m$ and thus a restriction of the Euclidean exponential map $\mathrm{Exp}_p\left(x\right) := p + x$ to $p \in \mathcal{M}$ is an exponential map on $\mathcal{M}$, which we call the *primal* exponential map. However, a Bregman potential ψ also induces a *dual* exponential map on $\mathcal{M}$, given by

$$\mathrm{Exp}^\psi_p\left(x\right) = \left(D\psi\right)\left(\left(D\psi\right)^{-1} + \left[D^2\psi\left(p\right)\right]x\right).$$

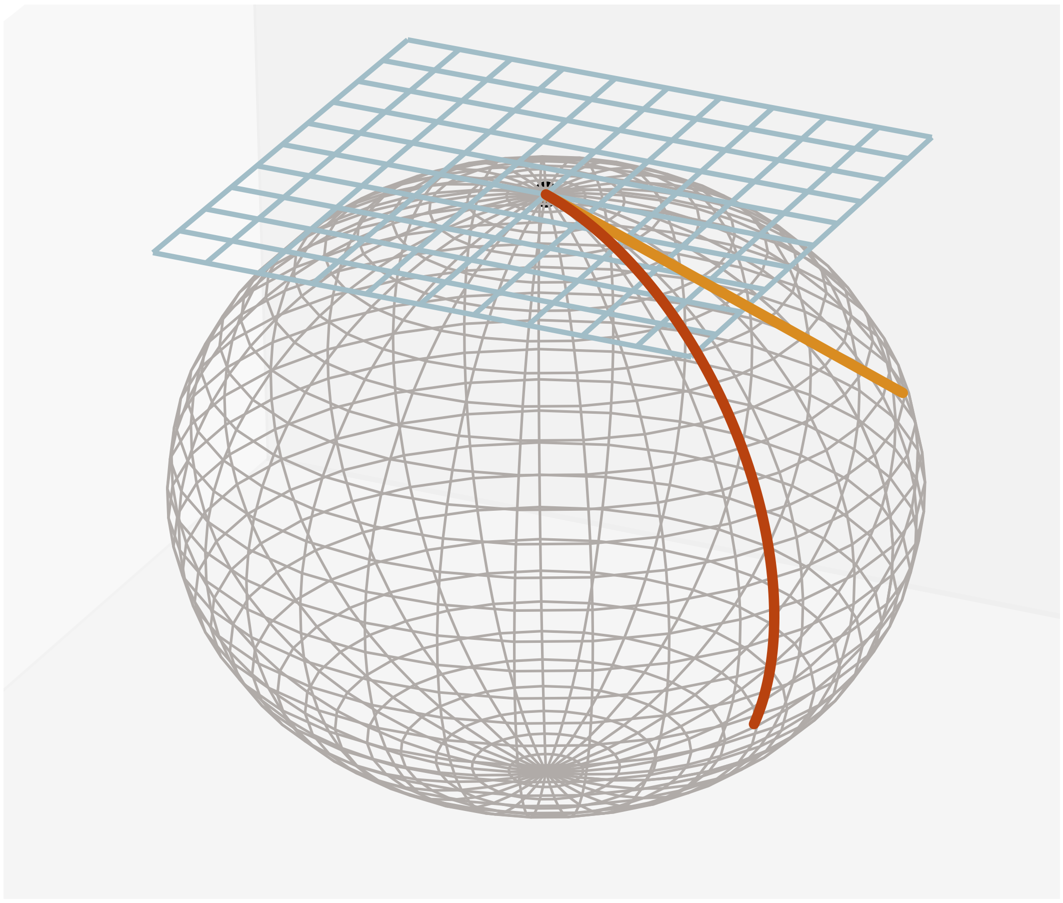


Figure 8.3: **Basic objects of differential geometry.** A manifold $\mathcal{M}$, tangent space $\mathcal{T}_p\mathcal{M}$, tangent vector $x \in \mathcal{T}_p\mathcal{M}$, and the path described by the exponential map $\{q : q = \mathrm{Exp}_p(tx), \text{for } t \in [0,1]\}$.

In the special case of $C = \mathbb{R}^m_+$ and $\psi(x) = \sum_i x_i \log(x_i) - x_i$ given by the Shannon entropy, the dual exponential map is given by

$$\left(\mathrm{Exp}^{\psi}_p(x)\right)_i = \exp\left(\log(p_i) + \frac{x_i}{p_i}\right) = p_i \exp\left(\frac{x_i}{p_i}\right).$$

Thus, while the *primal straight lines* $t \mapsto \mathrm{Exp}_x(tx)$ have a constant additive rate of change, *dual straight lines* $t \mapsto \mathrm{Exp}^{\psi}_x(tx)$ have constant relative rate of change.

An important property of the dual exponential map is that it inherits the completeness property of ψ.

Lemma 17. *If the potential ψ is complete with respect to $\mathcal{M}$, the dual exponential map* Exp^{ψ} *is complete in the sense that*

$$\forall p \in \mathcal{M}, \forall x \in \mathbb{R}^m : \mathrm{Exp}^{\psi}_p(x) \in \mathcal{M}. \tag{8.12}$$

Thus, a complete potential ψ allows us to follow a direction $x \in \mathcal{T}_p\mathcal{M}$ in $\mathcal{M} = \operatorname{int} C$ in such a way that we never accidentally leave the feasible set. We will now recast mirror descent in terms of the dual geometry induced by ψ.

8.5 Competitive mirror descent

In [202], it is observed that mirror descent has a natural formulation in terms of the dual geometry induced by ψ. Namely, its updates can be expressed as:

1. Choose a direction of movement that minimizes a linear local approximation, regularized with the metric induced by ψ.

$$\Delta x = \operatorname*{arg\,min}_{x \in \mathcal{T}_{x_k}\mathcal{M}} [D_x f] x + \frac{1}{2} x^T \left[D^2 \psi(x_k)\right] x$$

2. Compute x_{k+1} by moving into this direction, according to the *dual* geometry of ψ.

$$x_{k+1} = \operatorname{Exp}_{x_k}^{\psi}(\Delta x)$$

For ψ and ϕ complete Bregman divergences on C and $\mathcal{G}$, this form of mirror descent can be readily extended to Problem (8.5), resulting in *competitive mirror descent (CMD)*:

1. Solve for a local Nash equilibrium where both players try to minimize the bilinear local approximation of their objective, regularized with the metrics induced by ψ and ϕ.

$$\Delta x = \operatorname*{arg\,min}_{x \in \mathcal{T}_{x_k} C} [D_x f] x + x^\top \left[D_{xy}^2 f\right] y + \left[D_y f\right] y + \frac{1}{2} x^\top \left[D^2 \psi\right] x$$

$$\Delta y = \operatorname*{arg\,min}_{y \in \mathcal{T}_{y_k} \mathcal{G}} [D_x g] x + x^\top \left[D_{xy}^2 g\right] y + \left[D_y g\right] y + \frac{1}{2} y^\top \left[D^2 \phi\right] y$$

2. Compute x_{k+1} (y_{k+1}) by moving into this direction according to the dual geometries of ψ (ϕ).

$$x_{k+1} = \operatorname{Exp}_{x_k}^{\psi_C}(\Delta x), \quad y_{k+1} = \operatorname{Exp}_{y_k}^{\phi}(\Delta y)$$

Since ψ and ϕ are complete, each iterate is guaranteed to be feasible, while the local game (8.6) is quadratic and can be solved in closed form, resulting in Algorithm 18.

8.6 Numerical comparison

As discussed in Section 7.3, the *competitive term* involving the mixed Hessian $D_{xy}^2 f, D_{yx}^2 g$ is crucial to ensure convergence for bilinear problems. Most methods that include a competitive term such as [24, 89, 147, 173] encounter the empty threats-phenomenon described in Section 8.2 when combined with a projection. A notable exception is the projected extragradient method (PXGD) [143] (see also [76, Chapter 12]), and the extra mirror descent (XMD) proposed by [171], which we therefore see as the main alternatives to CMD. We also compare to an implementation of *naive* CMD (with update rule given by (8.3)) using XMD to solve the nonlinear local problem.

We begin with competitive optimization problems derived by applying Section 8.1 applying from classical matrix games. *Matching pennies* results in formulated as

$$\min_{p \in \mathbb{R}_+^2, \mu \in \mathbb{R}} \max_{q \in \mathbb{R}_+^2, \nu \in \mathbb{R}} \left(p_{\text{head}} - p_{\text{tail}}\right)\left(q_{\text{head}} - q_{\text{tail}}\right) + \mu q_{\text{total}} - \nu q_{\text{total}}, \tag{8.13}$$

where $p_{\text{total}}, q_{\text{total}}$ denote the total mass of the probability vectors p and q. While all methods considered here are able to find the equilibrium $(p, \mu) = (q, \nu) = (2^{-1}, 2^{-1}, 0)$, just as in the case of CGD in Chapter 7, we observe that by increasing the step size, we can speed up the convergence of CMD, while XMD or PXGD diverge eventually.

The Nash equilibrium for matching pennies lies in the interior of the constraint set. Therefore even PCGD converges to the correct equilibrium. When using Lagrange multipliers to normalization, the nonzero sum game *prisoner's dilemma* has the form.

$$\begin{aligned} &\min_{p \in \mathbb{R}_+^2, \mu \in \mathbb{R}} p_{\text{silent}} q_{\text{silent}} + 2 p_{\text{betrays}} q_{\text{betrays}} + 3 p_{\text{silent}} q_{\text{betrays}} + \mu q_{\text{total}} - \nu q_{\text{total}} \\ &\min_{q \in \mathbb{R}_+^2, \nu \in \mathbb{R}} p_{\text{silent}} q_{\text{silent}} + 2 p_{\text{betrays}} q_{\text{betrays}} + 3 p_{\text{betrays}} q_{\text{silent}} - \mu q_{\text{total}} + \nu q_{\text{total}}. \end{aligned} \tag{8.14}$$

When applying CMD, XMD, PCGD, and PXGD over a wide range of step sizes we observe that for all methods $p_{\text{silent}}, q_{\text{silent}}$ converge to the correct solution: one. (c.f. Section .5.3). However, under PCGD, $p_{\text{betray}}, q_{\text{betray}}$ converge to values larger than one, due to *empty threats*. Under CMD, $p_{\text{silent}}, q_{\text{silent}}$ converge to one as soon as the step size of the Lagrange multipliers is large enough, while XMD and PXGD need much smaller step sizes on the p and q, or else they oscillate or diverge.

We now seek answers to the following questions: **(1:)** Theoretical analysis of MD such as [30] suggests that the advantage of mirror descent over projected gradient

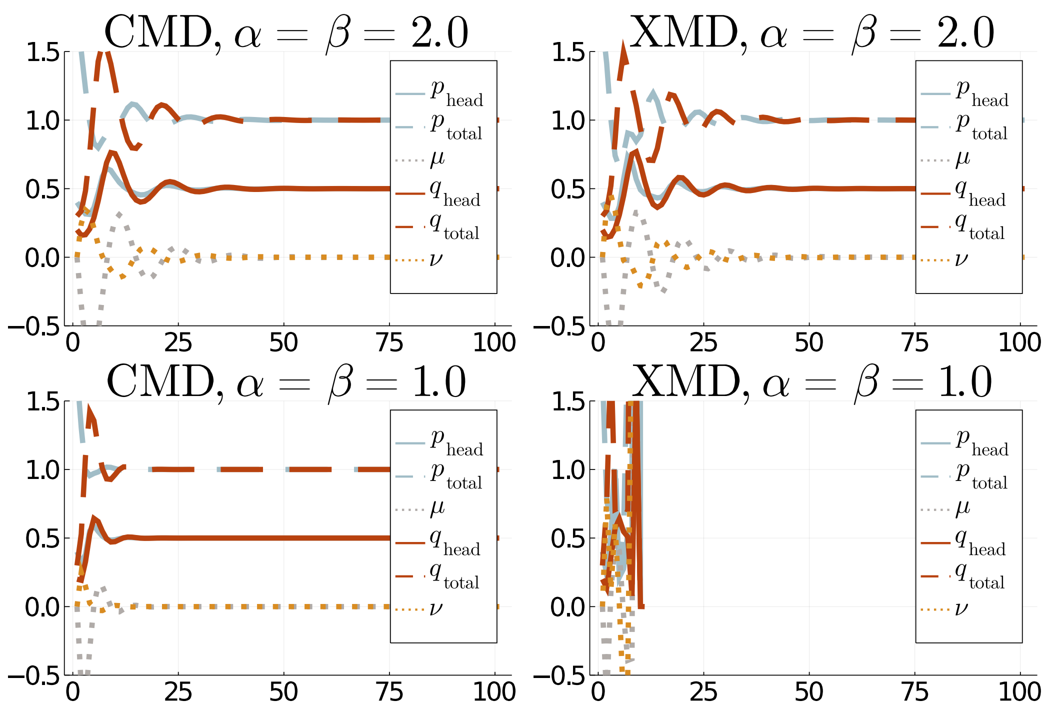


Figure 8.4: **Penny matching.** When applying CMD and XMD to penny matching ((8.13)), both methods converge. But as we increase the step sizes α^{-1}, β^{-1}, CMD converges faster and XMD diverges.

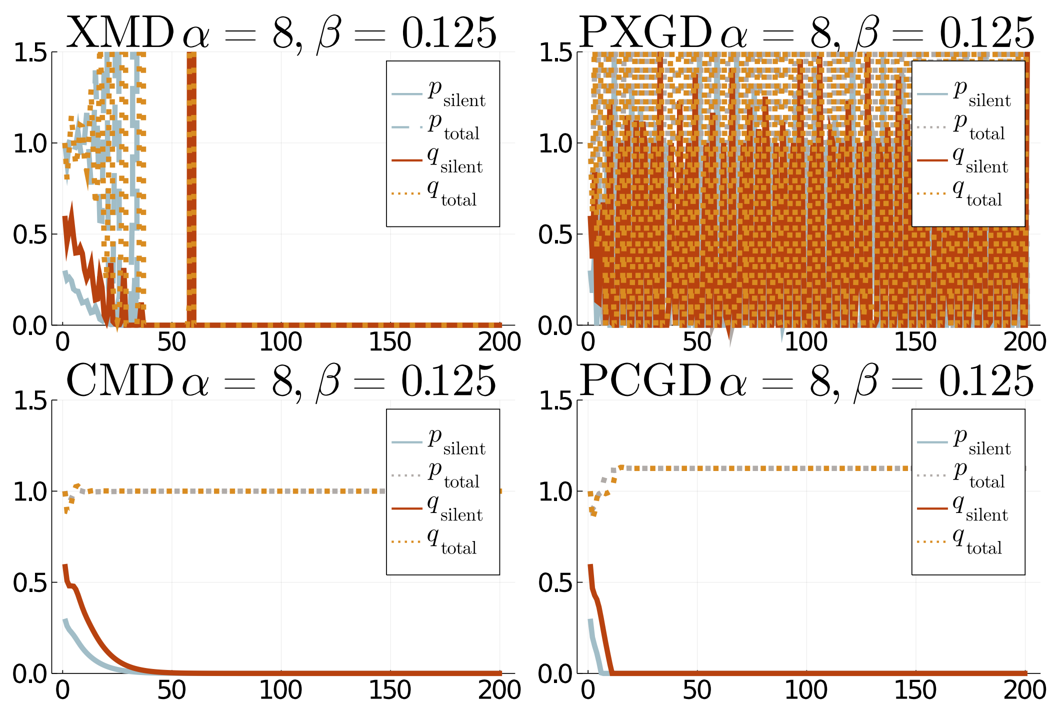


Figure 8.5: **Prisoner's dilemma.** Applied to *prisoner's dilemma* (8.14), CMD converges for step sizes α^{-1}, β^{-1} for which XMD and PXGD diverge. PCGD converges to the wrong solution, due to empty threats.

descent is most significant in high dimensions. Is CMD able to translate this advantage to competitive optimization in high dimensions? **(2:)** For large problems, the quadratic local problem of CMD needs to be solved by iterative methods. Can it beat XMD and PXGD even when accounting fairly for the complexity of the inner loop? **(3:)** Does CMD beat naive CMD (NCGD, (8.3)) in terms of computational complexity when accounting for the inner loop?

To this end, we move to a high-dimensional robust regression problem on the probability simplex given by

$$\min_{x:\, x \geq 0,\, \mathbf{1}^\top x = 1} \|Ax - b\|^2 \tag{8.15}$$

for $A \in \mathbb{R}^{50 \times 5000}$, $A_{ij} \sim \mathcal{N}(0, \mathrm{Id})$ i. i. d., $\epsilon \sim \mathcal{N}(0, \mathrm{Id})$, and $b = (A_{:,1} + A_{:,2})/2 + \epsilon$. In order to enforce the normalization constraint $\mathbf{1}^\top x = 1$, we introduce a Lagrange multiplier $y \in \mathbb{R}$ and solve the competitive optimization problem given as

$$\min_{x \in \mathbb{R}^{5000}_{+}} \|Ax - b\|^2 + y(\mathbf{1}^\top x - 1), \quad \min_{y \in \mathbb{R}} -y(\mathbf{1}^\top x - 1). \tag{8.16}$$

We use different inverse step sizes $\alpha \in \{100, 1000\}$ for x and $\beta \in \{1, 10, 100, 1000\}$ for y and solve the competitive optimization problem using CMW and PX. We then plot the loss incurred when using $x/\mathbf{1}^\top x$ as a function of the number of iterations. The extragradient method stalled for all step sizes that we tried, which is why we introduce extramirror (PXM), a mirror descent version of extragradient that at each step uses the gradient computed in the next iteration of mirror descent to perform a mirror descent update in the present iteration. As shown in Figure 8.6 CMW is the only algorithm that converges over the entire range of α and β. The projected extragradient method always stalls, while the extramirror algorithm diverges and produces NAN values for the largest step size. Generally speaking, CMW converges faster for larger step sizes, while PXM converges faster for smaller step sizes, as shown in Figure 8.6. Of course, to compare apples to apples, we need to account for the complexity of solving the matrix inverse in CMW. To this end, we show in Figure 8.7 the objective value compared to the number of gradient computations and Hessian-vector products. Therefore, each outer iteration of the extragradient and extramirror methods amounts to an x-axis value of 4, while an outer iteration of CMW, where the conjugate gradient solver requires k steps, amounts to an x-axis value of $4 + 2k$.

Similar to Chapter 7, we observe that even when fairly accounting for the complexity of the matrix inverse, CMW is competitive with PXM, beating it for larger stepsizes

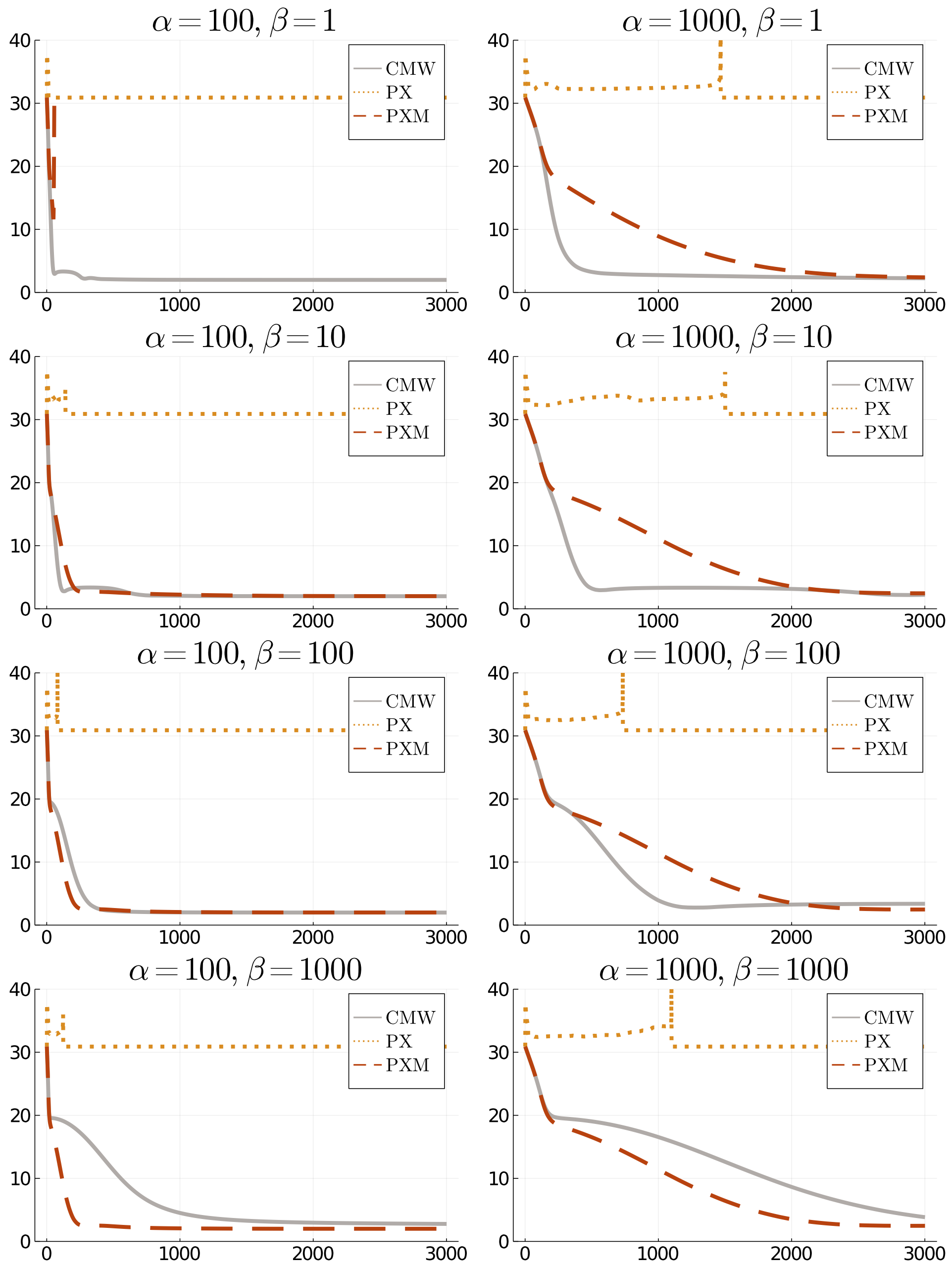


Figure 8.6: **Convergence against outer iterations.** We plot the objective value in Equation 8.15 (after normalization of x) compared to outer iterations. In the first panel, PXM diverges and produces NAN values, which is why the plot is incomplete.

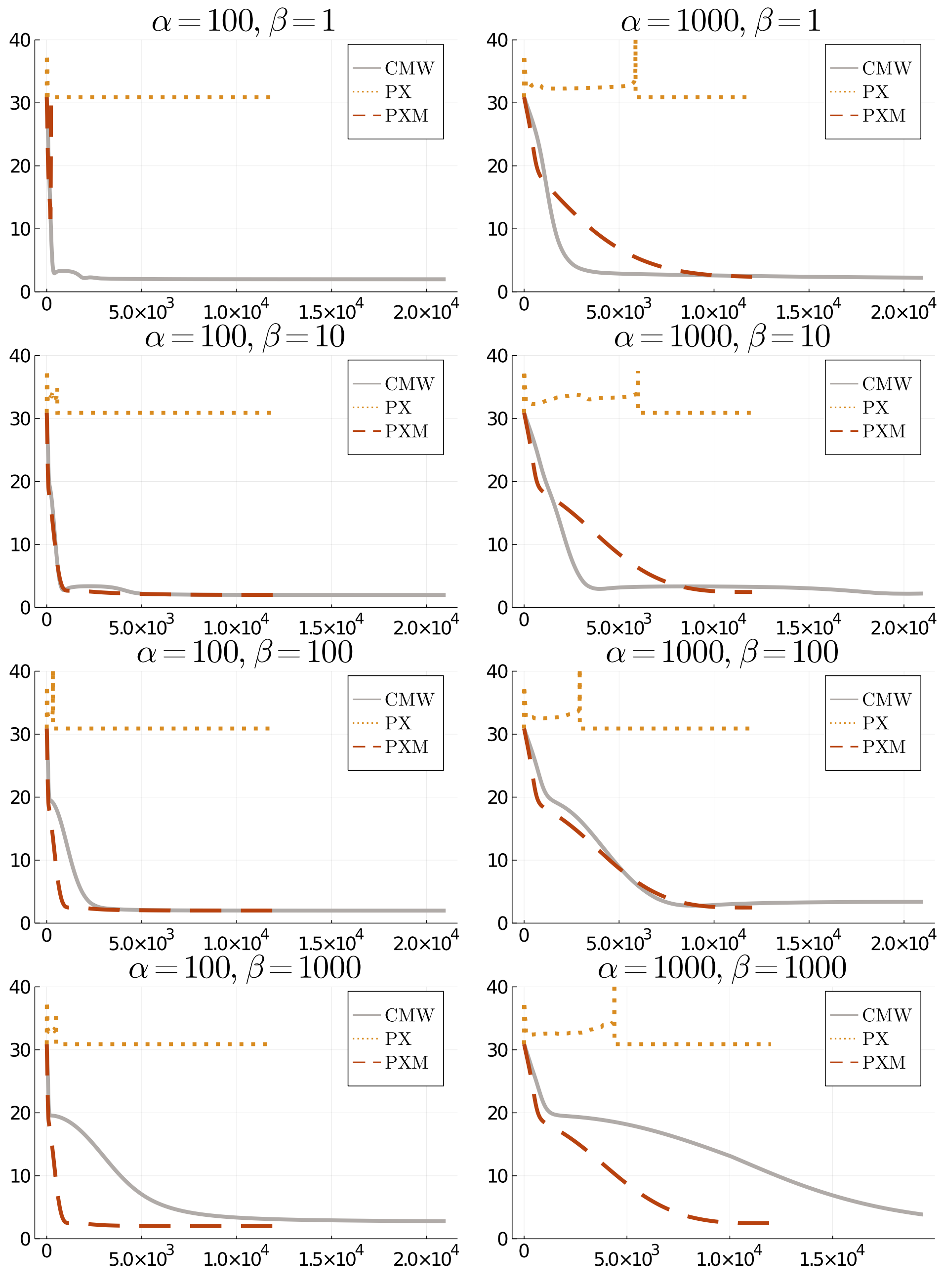


Figure 8.7: **Convergence against backprops.** We plot the objective value in Equation 8.15 (after normalization of x) compared to the number of gradient computations and Hessian-vector products, accounting for the inner loop of CMW.

while being a bit slower for smaller step sizes. Importantly, it is significantly more robust and converges even in settings where the competing methods diverge.

We conclude with an application of CMD to robust reinforcement learning, taken from [256]. We consider an algorithm with minimax linear quadratic (LQ) game where a *protagonist* player choses an action $u_t \in \mathbb{R}^m$, while an *antagonist* environmental agent chooses a disturbance $w_t \in \mathbb{R}^p$:

$$x_{t+1} = Ax_t + Bu_t + Cw_t, \tag{8.17}$$

with $x_t \in \mathbb{R}^n$ as the state vector. In LQR, a linear RL agent *minimizes* an infinite-horizon quadratic cost: $\mathbb{E}_{x_0 \sim \mathcal{P}} \left[\sum_{t=0}^{\infty} (x_t^T Q x_t + u_t^T R^u u_t)\right]$ where $Q \in \mathbb{R}^{n \times n}$ and $R^u \in \mathbb{R}^{m \times m}$ are positive definite. The two-player dynamics in (8.17) adds an environmental agent to the LQR formulation where the environmental agent adversarially chooses disturbance w_t that affect the states x_{t+1} through C. LQ games consider the following minimax objective:

$$\inf_{u_{t,t \geq 0}} \sup_{w_{t,t \geq 0}} \mathbb{E}_{x_0 \sim \mathcal{P}} \left[\sum_{t=0}^{\infty} (x_t^T Q x_t + u_t^T R^u u_t - w_t^T R^w w_t) \right]. \tag{8.18}$$

Zhang et al. [259] shows that the solution to the zero-sum LQ game (8.18) subject to (8.17) corresponds to the solution to a mixed $\mathcal{H}_2/\mathcal{H}_\infty$ problem [59]. Therefore, the LQ game can be interpreted both as finding a *robust* LQR policy against dynamic disturbances and as an optimal controller for a mixed $\mathcal{H}_2/\mathcal{H}_\infty$ problem.

Let the policy for the RL agent (who seeks to *minimize* the cost) and the environmental agent (who seeks to maximize the cost) take the form of $u_t = Kx_t$ and $w_t = Lx_t$ with $K \in \mathbb{R}^{m \times n}$ and $L \in \mathbb{R}^{p \times n}$. We denote the objective in (8.18) to be $C(K, L)$ to emphasize its dependence on the policy parameters. Under the condition $L \in \Omega := \{L \mid Q - L^T R^w L > 0\}$ and other technical assumptions [259], a globally unique and obtainable Nash equilibrium of the LQ game exists. We can pose an equivalent constrained minimax game for (8.18) as:

$$\min_{K} \max_{L \in \Omega} C(K, L). \tag{8.19}$$

In this game, the RL agent finds the best linear policy that is robust against the worst dynamic environmental disturbances. The solution to the LQ game can be obtained via linear matrix inequality [59]. However, when the system matrices and the objective function are unknown, our proposed framework for robust policy via CMD can be applied.

The Lagrangian of (8.19) with Lagrangian multiplier $\Lambda \in \mathcal{S}^n_{++}$ is given by:

$$\mathcal{L}(\Lambda, K, L) = C(K, L) - \left\langle \Lambda, L^T R^w L - Q \right\rangle, \tag{8.20}$$

where $\langle X, Y \rangle := \mathrm{tr}(X^T Y)$ denotes the matrix inner product. We augment the RL agent (K) with the Lagrangian multiplier Λ that penalizes the adversary (L) if it does not satisfy the constraint $L \in \Omega$. Using the Lagrangian as the augmented objective function, we arrive at the following constrained minimax game:

$$\min_{K, \Lambda \in \mathcal{S}^n_{++}} \max_{L} \ \mathcal{L}(\Lambda, K, L). \tag{8.21}$$

Note that the Lagrangian multiplier that enforces the constraint on the *maximizing* player is assigned to the *minimizing* player. It is straightforward to verify that the solution to (8.19) is the solution to (8.21) and vice versa because of the complementary slackness at KKT points. Intuitively, (8.21) means that whenever the constraints on the *maximizing* player are not satisfied, the *minimizing* player can improve its objective by increasing the Lagrange multiplier.

We choose the log-determinant function as the Bregman divergence associated to the Lagrange multiplier $\Lambda \in \mathcal{S}^n_{++}$. For the unconstrained variables in (8.21), we use the squared Frobenius norm. Following the notation of Section 8.3, the Bregman divergences for both players are

$$\psi(K, \Lambda) = -\mathrm{logdet}(\Lambda) + \frac{1}{2}\|K\|_F^2 \quad , \quad \phi(L) = \frac{1}{2}\|L\|_F^2.$$

Consider a double integrator LQ game with $A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$, $B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, and $C = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}$ with $Q = I$, $R^u = 1$, $R^w = 20$. We randomly generate a stabilizing K_0 and initialize the environmental agent's parameter L_0 in the interior of the constraint set. We sample trajectories of length $T = 15$ as the finite-horizon approximation to the infinite-horizon LQ cost and compute the corresponding gradient and Hessian estimations.

In Figure 8.8, we compare CMD to projected nested gradient descent (PNGD) proposed in [259] and projected gradient descent ascent (PGDA) as a baseline. For each of the methods, we test a variety of stepsizes for both minimizing player and maximizing player varying from 10^{-3} to 10^{-5}. We observe that for stepsizes larger than 10^{-5}, both PNGD with 50 inner loop iteration and PGDA diverges. On the other hand, Figure 8.9 shows that CMD is stable for larger stepsizes than the other methods.

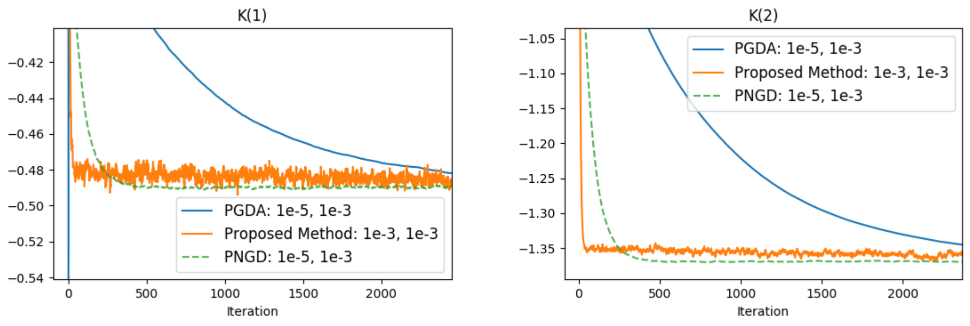


Figure 8.8: **Comparison of CMD to PNGD and PGDA.** We tested step sizes varying from 10^{-3} to 10^{-5} for the proposed algorithm (CMD), PNGD with inner loop iteration number set to 10, and PGDA. For each method, we plot the fastest converging trajectory against the number of *outer* iterations. The two step sizes are specified for minimizing player and maximizing player, respectively. Optimal closed-form solution is $K^* = [-0.4913, -1.3599]^T$.

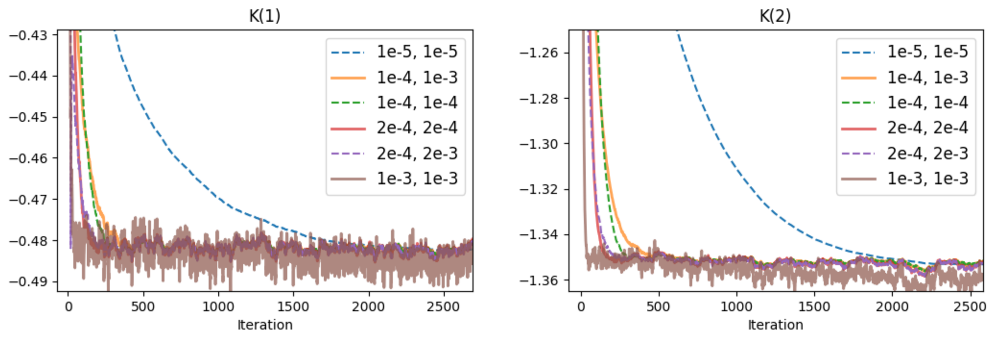


Figure 8.9: **Robustness of CMD to choice of step size.** The two panels show the iteration trajectory for the coordinates of parameter K. The two step sizes are specified for minimizing player and maximizing player, respectively. Optimal closed-form solution is $K^* = [-0.4913, -1.3599]^T$.

Chapter 9

IMPLICIT COMPETITIVE REGULARIZATION

9.1 Introduction

Generative adversarial networks (GANs) [96] are a class of generative models based on a competitive game between a *generator* that tries to generate realistic new data and a *discriminator* that tries to distinguish generated from real data. In practice, both players are parameterized by neural networks that are trained simultaneously by a variant of stochastic gradient descent.

The minimax interpretation. Presently, the success of GANs is mostly attributed to properties of the divergence or metric obtained under an optimal discriminator. For instance, an optimal discriminator in the original GAN leads to a generator loss equal to the Jensen-Shannon divergence between real and generated distribution. Optimization over the generator is then seen as approximately minimizing this divergence. We refer to this point of view as the *minimax interpretation*. The minimax interpretation has led to the development of numerous GAN variants that aim to use divergences or metrics with better theoretical properties.

The GAN-dilemma. However, every attempt to explain GAN performance with the minimax interpretation faces one of the two following problems:

1. Without regularity constraints, the discriminator can always be perfect. This is because it can selectively assign a high score to the finite amount of real data points while assigning a low score on the remaining support of the generator distribution, as illustrated in Figure 9.1. Therefore, the Jensen-Shannon divergence between a continuous and a discrete distribution always achieves its maximal value, a property that is shared by all divergences that do not impose regularity constraints on the discriminator. Thus, these divergences cannot meaningfully compare the quality of different generators.

2. Imposing regularity constraints needs a measure of similarity of images. Imposing regularity on the discriminator amounts to forcing it to map similar images to similar results. To do so, we would require a notion of similarity between images that is congruent with human perception. This is a longstanding unsolved problem

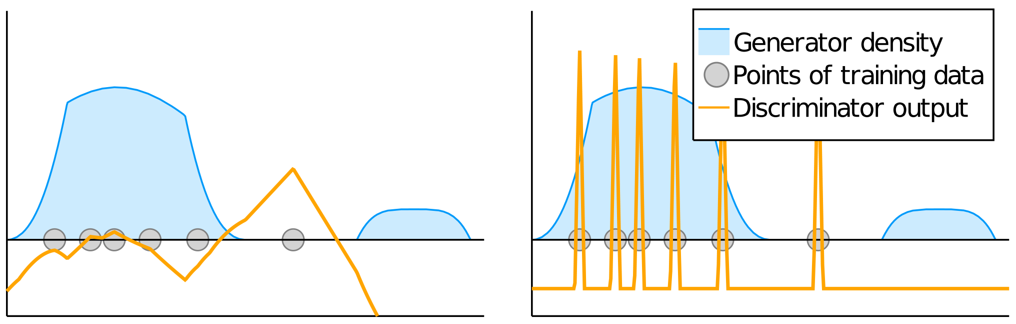


Figure 9.1: **The discriminator can always improve.** We want the discriminator confidence to reflect the relative abundance of true and fake data (left). But by picking out individual data points, the discriminator can almost always achieve arbitrarily low loss on any finite data set (right). Even in the limit of infinite data, the slightest misalignment of the supports of generated and real data can be exploited in a similar way.

in computer vision. Commonly used gradient penalties use the Euclidean norm, which is known to poorly capture visual similarity, as illustrated in Figure 9.2.

We believe that the different divergences underlying the various GAN formulations have little to do with their ability to produce realistic images. This is supported by the large-scale studies of [159] that did not find systematic differences in the performance of GANs associated with different divergence measures. Understanding GAN performance is crucial in order to improve training stability and reduce the required amount of hyperparameter tuning.

A way out? Due to the GAN-dilemma, attempts at explaining the performance of GANs need to go beyond the minimax interpretation and consider the *dynamics* of the training process. In this chapter, we argue that an implicit regularization due to the simultaneous[1] training of generator and discriminator allows GANs to use the inductive biases of neural networks for the generation of realistic images.

Implicit competitive regularization. We define *implicit competitive regularization* (ICR) as the introduction of stable points or regions due to the simultaneous training of generator and discriminator that do not exist when only training generator (or discriminator) with gradient descent while keeping the discriminator (or generator) fixed.

[1]Here and in the following, when talking about simultaneous training, we include variants such as alternating gradient descent.

It has been previously observed that performing simultaneous gradient descent (SimGD) on both players leads to stable points that are not present when performing gradient descent with respect to either player while keeping the other player fixed [168]. These stable points are not local Nash equilibria, meaning that they are not locally optimal for both players. This phenomenon is commonly seen as a shortcoming of SimGD and modifications that promote convergence only to local Nash equilibria were proposed by [24, 169]. In contrast to this view, we believe that ICR is crucial to overcoming the GAN-dilemma and hence to explaining GAN performance in practice by allowing the inductive biases of the discriminator network to inform the generative model.

Summary of contributions. In this chapter, we point out that a fundamental dilemma prevents the common minimax interpretation of GANs from explaining their successes. We then show that implicit competitive regularization (ICR), which so far was believed to be a *flaw* of SimGD, is key to overcoming this dilemma. Based on simple examples and numerical experiments on real GANs, we illustrate how it allows using the inductive biases of neural networks for generative modeling, resulting in the spectacular performance of GANs.
We then use this understanding to improve GAN performance in practice. Interpreting ICR from a game-theoretic perspective, we reason that strategic behavior and opponent-awareness of generator and discriminator during the training procedure can strengthen ICR. These elements are present in competitive gradient descent introduced in Chapter 7, which is based on the two players solving for a local Nash equilibrium at each step of training. Accordingly, we observe that CGD greatly strengthens the effects of ICR. In comprehensive experiments on CIFAR 10, competitive gradient descent stabilizes previously unstable GAN formulations and achieves higher inception score than a wide range of explicit regularizers, using both WGAN loss and the original saturating GAN loss of [96]. In particular, taking an existing WGAN-GP implementation, dropping the gradient penalty, and training with CGD leads to the highest inception score in our experiments. We interpret this as additional evidence that ICR, as opposed to explicit regularization, is the key mechanism behind GAN performance.

9.2 The GAN-dilemma

In this section, we study in more detail the fundamental dilemma that prevents the common minimax interpretation from explaining the successes of GANs. In

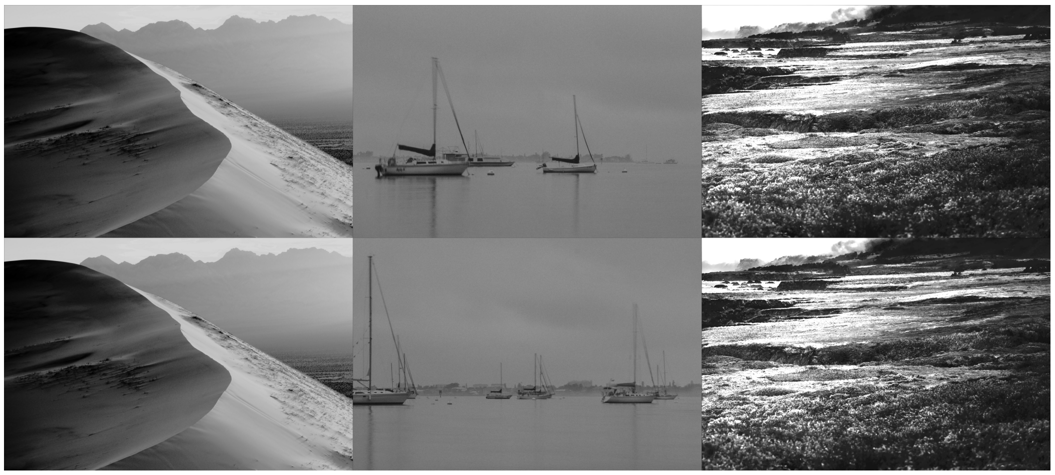


Figure 9.2: **The Euclidean distance is not perceptual.** We would like to challenge the reader to rank the above three pairs of images according to the Euclidean distance of their representation as vectors of pixel-intensities.[2]

particular, we show how the existing GAN variants fall into one or the other side of the GAN-dilemma.

Metric-agnostic GANs. In the original formulation due to [96], the two players are playing a zero-sum game with the loss function of the generator given by the binary cross-entropy

$$\min_{\mathcal{G}} \max_{\mathcal{D}} \frac{1}{2}\,\mathbb{E}_{x\sim P_{\text{data}}}\left[\log \mathcal{D}(x)\right] + \frac{1}{2}\,\mathbb{E}_{x\sim P_{\mathcal{G}}}\left[\log\left(1-\mathcal{D}(x)\right)\right]. \tag{9.1}$$

Here, $\mathcal{G}$ is the probability distribution generated by the generator, $\mathcal{D}$ is the classifier provided by the discriminator, and P_{data} is the target measure, for example the empirical distribution of the training data. A key feature of the original GAN is that it depends on the discriminator only through its output when evaluated on samples. This property is shared, for instance, by the more general class of f-divergence GANs [186]. We call GAN formulations with this property *metric-agnostic*.

Metric-informed GANs. To address instabilities observed in the original GAN, [16] introduced WGAN, with loss function given by

$$\min_{\mathcal{G}} \max_{\mathcal{D}} \mathbb{E}_{x\sim P_{\text{data}}}\left[\mathcal{D}(x)\right] - \mathbb{E}_{x\sim P_{\mathcal{G}}}\left[\mathcal{D}(x)\right] + \mathcal{F}\left(\nabla \mathcal{D}\right), \tag{9.2}$$

[2]The pairs of images are ordered from left to right, in increasing order of distance. The first pair is identical, while the third pair differs by a tiny warping.

where $\mathcal{F}(\nabla\mathcal{D})$ is infinity if $\sup_x \|\nabla\mathcal{D}(x)\| > 1$, and zero, otherwise. [106] propose WGAN-GP, where this inequality constraint is relaxed by replacing $\mathcal{F}$ with a penalty, for instance $\mathcal{F}(\nabla\mathcal{D}) = \mathbb{E}\left[(\|\nabla_x\mathcal{D}\| - 1)^2\right]$. These GAN formulations are fundamentally different from metric-agnostic GANs in that they depend explicitly on the gradient of the discriminator. In particular, they depend on the choice of metric used to measure the size of $\nabla\mathcal{D}$. Subsequent to WGAN(-GP), which uses the Euclidean norm, other variants such as Sobolev-GAN [177], Banach-GAN [6], or Besov-GAN [239] have been proposed that use different metrics to measure gradient size. We refer to these types of GAN formulations as *metric-informed* GANs.

The problem with metric-agnostic GANs. GANs are able to generate highly realistic images, but they suffer from unstable training and mode collapse that often necessitates extensive hyperparameter tuning. Beginning with [15], these problems of the original GAN have been explained with the fact that the supports of the generator distribution and the training data are almost never perfectly aligned. For any fixed generator, the discriminator can take advantage of this fact to achieve arbitrarily low loss, as illustrated in Figure 9.1. In the case of the Formulation 9.1, this corresponds to the well-known fact that the Jensen-Shannon divergence between mutually singular measures is always maximal. This result extends to *all* metric-agnostic divergences simply because they have no way of accessing the degree of similarity between data points on disjoint supports.

[17, 122] emphasize that the discriminator is restricted to a function class parameterized by a neural network. However, the experiments of [15], as well as our own in Figure 9.4 clearly show the tendency of the discriminator to diverge as it achieves near-perfect accuracy. This is not surprising since [257] observed that modern neural networks are able to fit even *random* data perfectly. [15] also show that as the discriminator improves its classification loss, the generator achieves less and less useful gradient information. This is again not surprising since confidence scores of deep neural networks are known to be poorly calibrated [108]. Therefore, the outputs of a near-perfect discriminator can not be expected to provide a useful assessment of the quality of the generated samples.

Since GAN optimization is highly non-convex it is natural to ask if GANs find *locally* optimal points in the form of local Nash or Stackelberg equilibria. This local minmax interpretation has been emphasized by [84, 130], but the experiments of [35] as well as our own in Figure 9.4 suggest that good GAN solutions for metric-agnostic

GANs are typically not locally optimal for both players. It seems plausible that the discriminator, being highly overparameterized, can find a direction of improvement against most generators.

The problem with metric-informed GANs. The above observation has motivated the introduction of metric-informed GANs that restrict the size of the gradient of the discriminator (as a function mapping images to real numbers). This limits the discriminator's ability to capitalize on small misalignments between $\mathcal{D}$ and P_{data} and thus makes for a meaningful minimax interpretation even if the two measures have fully disjoint support. However, the Achilles heel of this approach is that it needs to choose a metric to quantify the magnitude of the discriminator's gradients. Most of the early work on metric-informed GANs chose to measure the size of $\nabla\mathcal{D}$ using the Euclidean norm [15, 16, 106, 139, 176, 208]. However, since the discriminator maps images to real numbers, this corresponds to quantifying the similarity of images at least locally by the Euclidean distance of vectors containing the intensity values of each pixel. As illustrated in Figure 9.2, this notion of similarity is poorly aligned with visual similarity even locally. From this point of view, it is not surprising that the generative model of [48], based on a differentiable optimal transport solver, produced samples of lower visual quality than WGAN-GP, despite achieving better approximation in Wasserstein metric. As noted by [48], these observations suggest that the performance of WGAN can not be explained by its relationship to the Wasserstein distance. When comparing a variety of GAN formulations with a fixed budget for hyperparameter tuning, [159] did not find systematic differences in their performance. This provides additional evidence that the key to GAN performance does not lie in the choice of a particular divergence between probability measures.

The metric-informed divergences considered so far were all based on the Euclidean distance between images. Other researchers have tried using different metrics on image space such as Sobolev or Besov norms [6, 177, 239], or kernel maximum mean discrepancy distances [39, 148, 151]. However, none of these metrics do a good job at capturing perceptual similarity either, which explains why these variants have not been observed to outperform WGAN(-GP) in general. Researchers in computer vision have proposed more sophisticated domain-specific distance measures [220], kernel functions [110, 224], and features maps [58]. Alternatively, methods from differential geometry have been used for image inter– and extrapolation [36, 70, 238]. But none of these approaches achieves performance comparable to that of neural networks, making them unlikely solutions for the GAN-dilemma.

A way out. Generative modeling means producing new samples that are *similar* to the training samples, but *not too similar* to each other. Thus, every generative method needs to choose how to measure similarity between samples, implicitly or explicitly.

When analyzing GANs from the minimax perspective, this assessment of image similarity seems to rely exclusively on the classical metrics and divergences used for their formulation. But modeling perceptual similarity is hard, and most commonly used GAN formulations are based on measures of similarity that are known to be terrible at this task. Thus, the minimax point of view can not explain why GANs produce images of higher visual quality than any other method. The key to image classification is to map *similar* images to *similar* labels. The fact that deep neural networks drastically outperform classical methods in these tasks leads us to believe that they capture the perceptual similarity between images far better than any classical model. We believe that the success of GANs is due to their ability to implicitly use the inductive biases of the discriminator network as a notion of similarity. They create images that *look real* to a neural network, which acts as a proxy for *looking real* to the human eye. In the next section, we propose a new mechanism, implicit competitive regularization, to explain this behavior.

9.3 Implicit competitive regularization (ICR)

Implicit regularization. Based on the discussion in the last section, any attempt at understanding GANs needs to involve the inductive biases of the discriminator. However, there is ample evidence that the inductive biases of neural networks do not arise from a limited ability to represent certain functions. Indeed, it is known that modern neural networks can fit almost arbitrary functions [57, 140, 257]. Rather, they seem to arise from the dynamics of gradient-based training that tends to converge to classifiers that generalize well, a phenomenon commonly referred to as *implicit regularization* [18, 19, 107, 145, 160, 183].

Implicit regularization is not enough for GANs. The implicit regularization induced by gradient descent lets neural networks prefer sets of weights with good generalization performance. However, the outputs of even a well-trained neural network are typically not informative about the confidence of the predicted class [108]. Thus, a discriminator trained on finite amounts of real data and data generated by a given generator can be expected to distinguish new real data from new data generated by a similar generator with high accuracy. However, its outputs do

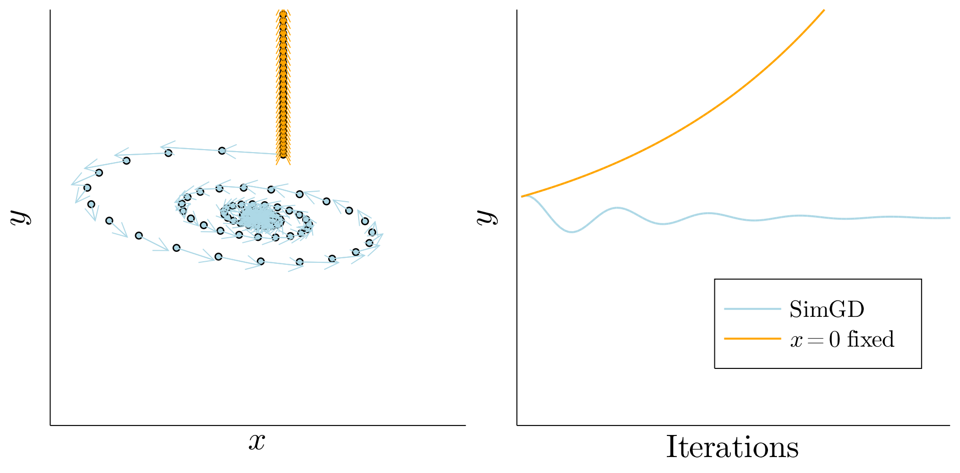


Figure 9.3: **ICR in the quadratic case.** When optimizing only y in Equation (9.3), it diverges rapidly to infinity, for any fixed y. If, however, we simultaneously optimize x and y with respective step sizes $\eta_x = 0.09$ and $\eta_y = 0.01$, we converge to $(0, 0)$.

not quantify the confidence of its prediction and thus of the visual quality of the generated samples. Therefore, even considering implicit regularization, a fully trained discriminator does not provide useful gradients for training the generator.

Implicit *competitive* regularization. We think that GAN training relies on *implicit competitive regularization* (ICR), an additional implicit regularization due to the simultaneous training of generator and discriminator. When training generator and discriminator simultaneously, ICR selectively stabilizes good generators that would not be stable when training one player while keeping the other player fixed.

Consider the game given by

$$\min_x \max_y x^2 + 10xy + y^2. \tag{9.3}$$

In this problem, for any fixed x, any choice of y will be sub-optimal, and gradient ascent on y (with x fixed) will diverge to infinity for almost all initial values.

What about simultaneous gradient descent? As has been observed before [168], simultaneous gradient descent with step sizes $\eta_x = 0.09$ for x and $\eta_y = 0.01$ for y will converge to $(0, 0)$, despite it being a locally *worst* strategy for the maximizing player. (See Figure 9.3 for an illustration.) This is a first example of ICR, whereby the simultaneous optimization of the two agents introduces additional attractive points to the dynamics that are *not* attractive when optimizing one of the players using gradient descent while keeping the other player fixed.

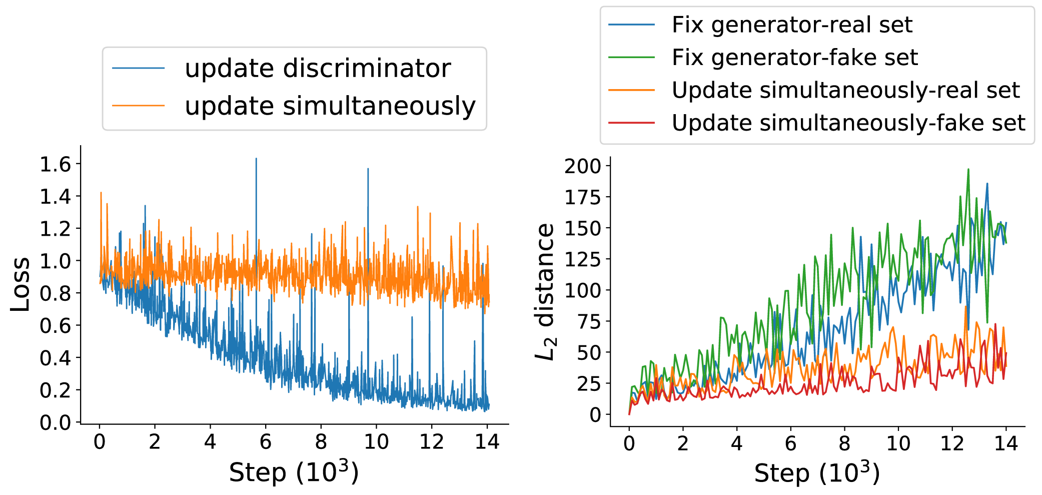


Figure 9.4: **ICR on MNIST.** We train a GAN on MNIST until we reach a *checkpoint* where it produces good images. (First image) We fix the generator and only train the discriminator, observing that it can reach near-zero loss. When instead training generator and discriminator jointly, the loss stays stable. (Second image) When trained individually, the discriminator moves significantly slower slower when trained jointly with the generator, as measured by its output on a set of thousand reference images.

As outlined in Section 9.2, the key to the performance of GANs has to lie in the simultaneous optimization process. We now provide evidence that the solutions found by GANs are indeed stabilized by ICR. To this end, we train a GAN on MNIST until it creates good images. We refer to the resulting generator and discriminator as the *checkpoint* generator and discriminator. We observe that the loss of both generator and discriminator, as well as the image quality, is somewhat stable even though it would diverge after a long time of training. If instead, starting at the checkpoint, we optimize only the discriminator while keeping the generator fixed, we observe that the discriminator loss drops rapidly. For the same number of iterations and using the same learning rate, the discriminator moves away from the checkpoint significantly faster as measured both by the Euclidean norm of the weights and the output on real and fake images. The observation that the discriminator diverges from the checkpoint faster when trained individually than when trained simultaneously with the generator suggests that the checkpoint, which produced good images, was stabilized by ICR.

9.4 How ICR lets GANs generate

A (hypo)thesis. In the example in the last section, the checkpoint producing good images was stabilized by ICR. However, we have not yet given a reason why points stabilized by ICR should have better generators, in general. For GANs to produce visually plausible images, there has to be some correspondence between the training of neural networks and human visual perception. Since learning and generalization are poorly understood even for ordinary neural network classifiers, we cannot avoid making an assumption on the nature of this relationship. This section relies on the following hypothesis.

Hypothesis How quickly the discriminator can pick up on an imperfection of the generator is correlated with the visual prominence of said imperfection.

It is common intuition in training neural network classifiers that more visually obvious patterns are learned in fewer iterations and from less data. It is also in line with the *coherent gradient hypothesis* of [45] that explains the generalization of neural networks with the fact that systematic patterns in the data generate more coherent gradients and are therefore learned faster. While a thorough verification of the hypothesis is beyond the scope of this work, we provide some empirical evidence in Figure 9.5.

This section argues for the following thesis.

Thesis ICR selectively stabilizes generators for which the discriminator can only improve its loss *slowly*. By the hypothesis, these generators will produce high-quality samples.

An argument in the quadratic case. We begin with the quadratic problem in Equation 9.3 and model the different speeds of learning of the two agents by changing their step sizes η_x and η_y. In Figure 9.6, we see that for $\left(\eta_x, \eta_y\right) = (0.03, 0.03)$, the two agents slowly diverge to infinity and for $\left(\eta_x, \eta_y\right) = (0.01, 0.09)$, divergence occurs rapidly. In general, stable points $\bar{x}$ of an iteration $(x_{k+1} = x_k + F(x_k)$ are characterized by **(1):** $F(\bar{x}) = 0$ and **(2)** $D_x F(\bar{x})$ having a spectral radius smaller than one [173, Proposition 3]. For SimGD applied to a zero sum game with the loss of x given by f, these are points with vanishing gradients such that

$$\mathrm{Id} - M := \mathrm{Id} - \begin{pmatrix} \eta_x D^2_{xx} f & \eta_x D^2_{xy} f \\ -\eta_y D^2_{yx} f & -\eta_y D^2_{yy} f \end{pmatrix}$$

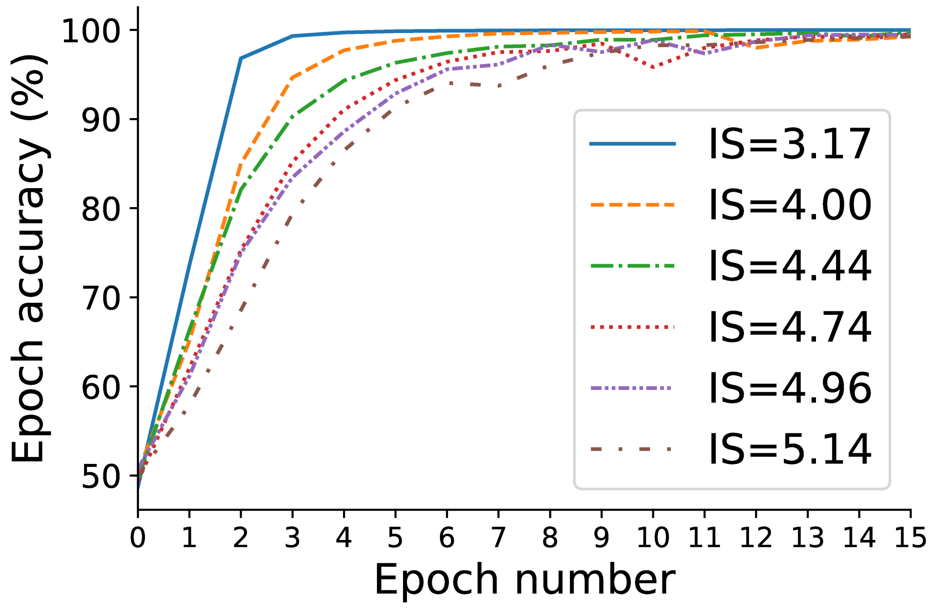


Figure 9.5: **Discriminator learning and image quality.** By prematurely stopping the training process, we obtain generators of different image quality on CIFAR10 (higher inception score (IS) reflects better image quality). We then train a new discriminator against this *fixed* generator and measure how quickly it increases its classification performance. We use a model trained on the 10-class classification task as a starting point for the discriminator to prevent the initial phase of training from polluting the measurements. While all discriminators achieve near-perfect accuracy eventually, the *rate* of improvement is inversely correlated to the inception score of the generator.

has a spectral radius smaller than one. For univariate x and y, we can set $a := D^2_{xx}f$, $b := D^2_{xy}f$, and $c := D^2_{yy}f$, and compute the characteristic polynomial of M as

$$p(\lambda) = \lambda^2 - (\eta_x a - \eta_y c)\lambda + (-\eta_x\eta_y ac + \eta_x\eta_y b^2).$$

For $\eta_x a > \eta_y c$ and $\eta_x\eta_y b^2 > \eta_x\eta_y ac$, the solutions of this equation have positive real part and therefore the eigenvalues of M have positive real part. By multiplying η_x and η_y by a small enough factor, we can obtain a spectral radius smaller than one (c. f. [168]). Thus, a small enough η_y and large enough mixed derivative b can ensure convergence even for positive c.

If we think of the maximizing player as the discriminator, slow learning (modeled by small η_y) is correlated to good images produced by the generator. Thus, in this interpretation, a good generator leads to ICR stabilizing the point $(0, 0)$ more strongly.

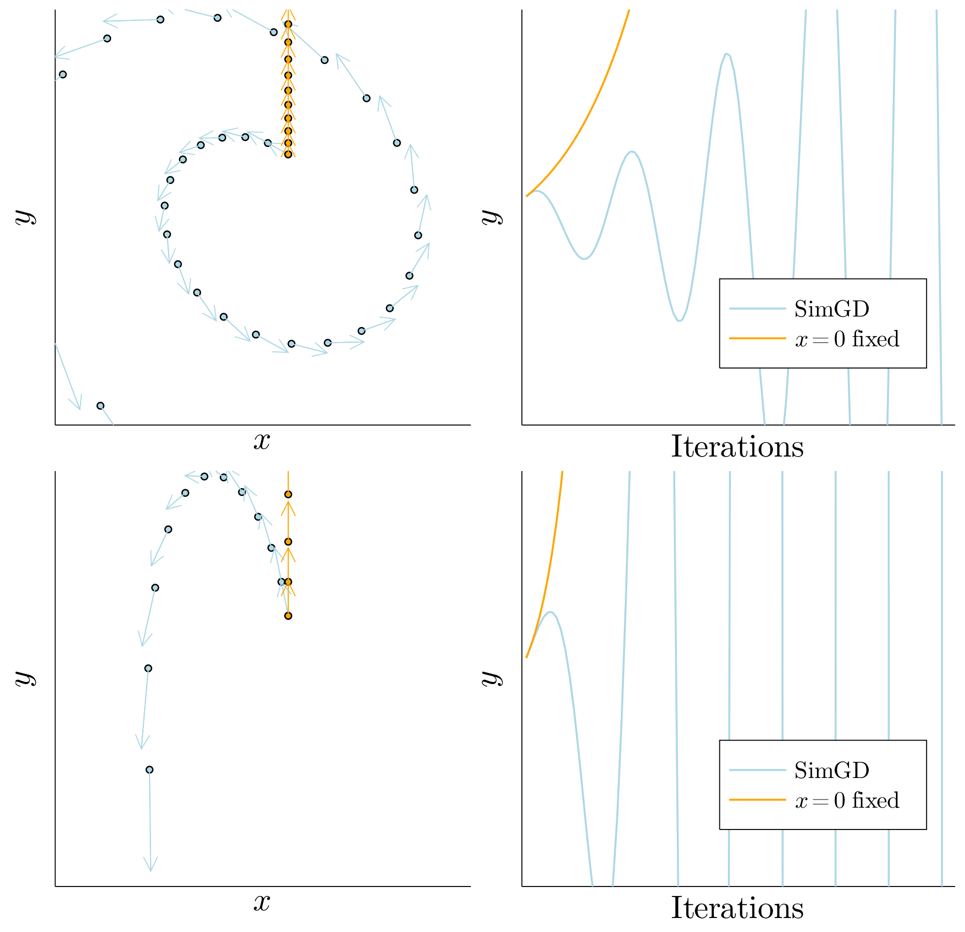


Figure 9.6: **ICR depends on speed of learning.** When changing the learning rates to $(\eta_x, \eta_y) = (0.03, 0.03)$ (top) or $(\eta_x, \eta_y) = (0.01, 0.09)$ (bottom), SimGD diverges.

Adversarial training as projection. Surprisingly, ICR allows us to compute a projection with respect to the perceptual distance of a neural network without quantifying this distance explicitly. Let us consider the following example. We construct a *generator* $\mathcal{G}$ that maps its 28 weights to a bivariate output. This nonlinear map is modelled as a tiny neural network with two hidden layers, with the final layer restricting the output to the set $\mathcal{S} := \left\{ (e^{s+t}, e^{s-t}) \middle| s \in \left[-\frac{1}{2}, \frac{1}{2}\right], t \in \mathbb{R} \right\} \subset \mathbb{R}^2$. We think of this as mapping a set of weights to a generative model that is characterized by only two parameters. In this parameterization, we assume that the target distribution is represented by the point $P_{\text{data}} = (2, 2)$. Importantly, as shown in Figure 9.7, there is no set of weights that allow the generator to output *exactly* P_{data}. This is to model the fact that in general, the generator will not be able to exactly reproduce the target distribution. We construct a *discriminator* $\mathcal{D}$ that maps a generative model (a pair of real numbers) and a set of 28 weights to a real number by a small, densely

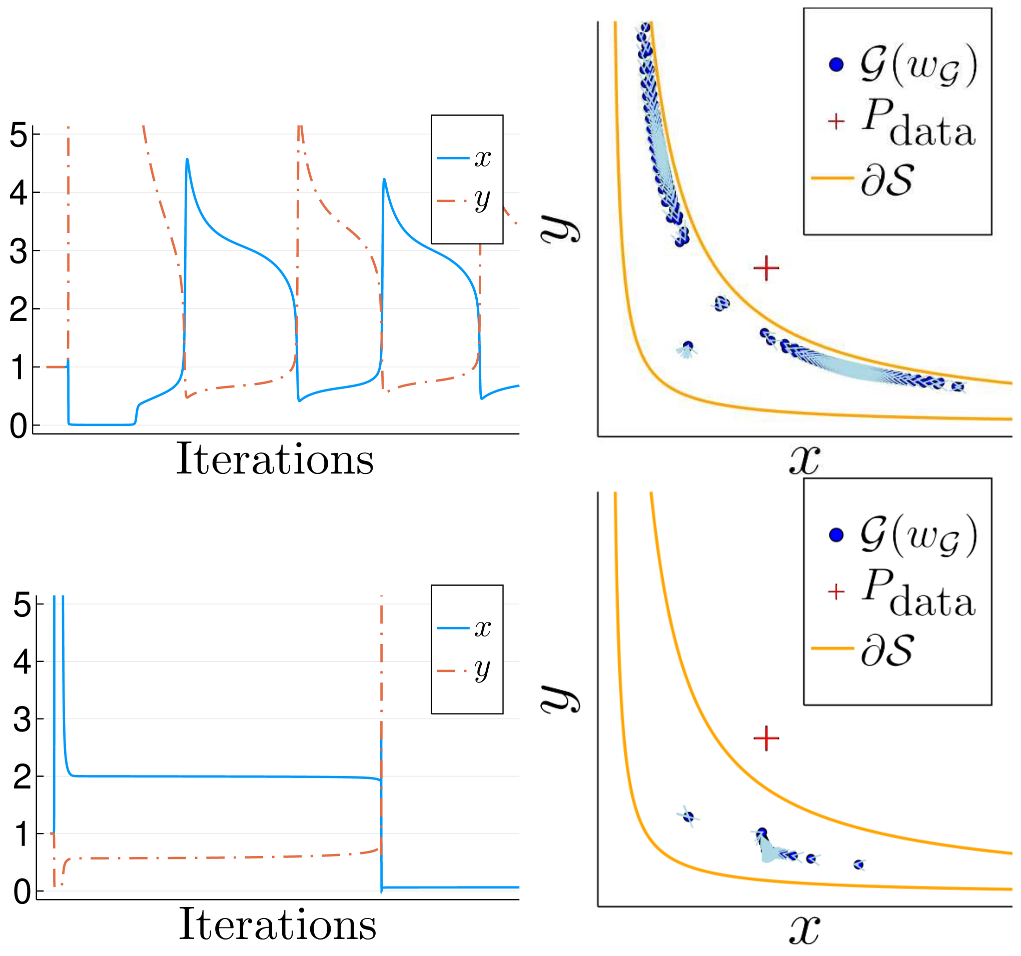


Figure 9.7: **Approximate projection via adversarial training.** On the left column, the discriminator picks up on errors in the x- and y-direction equally quickly. Therefore, the generator tries to satisfy the criteria alternatingly, leading to a cyclic pattern. In the right column, the discriminator picks up on errors in the x-direction much more quickly. This causes the generator to try to stay accurate in the x-direction.

connected neural network.

We want to model the difference in visual prominence of the two components of P_{data}. To this end, we assume that before being passed to the discriminator, $\mathcal{G}$ and P_{data} are rescaled by a diagonal matrix $\eta \in \mathbb{R}^{2\times 2}$. Thus, η determines the relative size of the gradients of $\mathcal{D}$ with respect to the first and second components of the input data. This models the hypothesis that a real discriminator will pick up more quickly on visually prominent features. Importantly, we assume η to be unknown since we do not have access to a metric measuring "visual similarity to a neural network".

We will now show how adversarial training can be used to approximate a projection with respect to η, without knowing η. We use the loss

$$\min_{w_{\mathcal{G}} \in \mathbb{R}^{28}} \max_{w_{\mathcal{D}} \in \mathbb{R}^{28}} \mathcal{D}\left(\eta P_{\text{data}}, w_{\mathcal{D}}\right) - \mathcal{D}\left(\eta \mathcal{G}\left(w_{\mathcal{G}}\right), w_{\mathcal{D}}\right) \tag{9.4}$$

and train the two networks using simultaneous gradient descent. For η equal to the identity, we see oscillatory training behavior as $\mathcal{G}$ tries to be accurate first in one, then the other direction. If we instead use $\eta = \begin{pmatrix} 1 & 0 \\ 0 & 10^{-2} \end{pmatrix}$, we are modelling the first component as being more visually prominent. Instead of the oscillatory patterns from before, we observe long periods where the value of the first component of $\mathcal{G}\left(w_{\mathcal{G}}\right)$ is equal to the first component of P_{data} (see Figure 9.7). Without knowing η, we have approximated the projection of P_{data} onto $\mathcal{S}$ with respect to the metric given by $(x, y) \mapsto \|\eta(x, y)\|$. To do so, we used the fact that this point is subject to the slowest learning discriminator, and thus the strongest ICR.

We believe that GANs use the same mechanism to compute generators that are close to the true data in the perceptual distance of the discriminator, which in turn acts as a proxy for the perceptual distance of humans.

9.5 Competitive gradient descent amplifies ICR

How to strengthen ICR. We have provided evidence that GANs' ability to generate visually plausible images can be explained by ICR selectively stabilizing good generators. It is well known that GANs often exhibit unstable training behavior, which is mirrored by the observations in Figures 9.3, 9.4 and 9.7 that ICR often only leads to weak, temporary stability. Thus, it would be desirable to find algorithms that induce stronger ICR than SimGD. To this end, we will find a game-theoretic point of view useful.

Cooperation in a zero-sum game? As discussed in the last section, ICR can stabilize solutions that are locally suboptimal for at least one of the players. Since we did not model either of the two players as altruistic, this behavior may seem puzzling. It is likely for this reason that ICR has mostly been seen as a flaw, rather than a feature of SimGD.

Convergence by competition. The quadratic example in Equation (9.3) shows that the bilinear term xy is crucial for the presence of ICR. Otherwise, SimGD reduces to each player moving independently according to gradient descent. In fact, the strength of ICR decreases rapidly as $|\alpha|$ and $|\beta|$ diverge to infinity.

The mixed term xy models the ability of each player to retaliate against actions of the other player. In the case of $\beta < 0$, as the maximizing player y moves to plus infinity in order to maximize its reward, it becomes a locally optimal strategy for the minimizing player x to move towards negative infinity in order to minimize the dominant term xy. If $|\beta| \ll 1$, it is favorable for the maximizing player to move back towards zero in order to maximize the dominant term xy. The reason for the maximizing player to stay in the sub-optimal point $y = 0$ (the *maximizer* of its loss, for $x = 0$) is that the minimizing player can use the mixed term xy to punish every move of y with a counterattack. Thus, the need to avoid counterattacks justifies the seemingly sub-optimal decision of the maximizing player to stay in $y = 0$.

The generator strikes back! This phenomenon is also present in the example of Figure 9.4. Consider the checkpoint generator from Figure 9.4 and the over-trained discriminator that achieves near-perfect score against the discriminator. As we can see in Figure 9.8, training the generator while keeping the over-trained discriminator fixed leads to a rapidly increasing discriminator loss. The over-trained discriminator has become vulnerable to counterattack by the generator! If instead the generator is trained against the checkpoint discriminator, the loss increases only slowly. Thus, ICR can be interpreted as the discriminator trying to avoid counterattack by the generator.

Agent modelling for stronger ICR. The update (x, y) of SimGD applied to the loss function f can be interpreted as the two players solving, at each step, the local optimization problem

$$\min_x x^\top \nabla_x f(x_k, y_k) + \frac{\|x\|^2}{2\eta}, \quad \max_y y^\top \nabla_y f(x_k, y_k) - \frac{\|y\|^2}{2\eta}.$$

The terms $x^\top \nabla_x f(x_k, y_k), y^\top \nabla_y f(x_k, y_k)$ express the *belief* about the loss associated to different actions, based on local information. The quadratic regularization terms express their *uncertainty* about these beliefs, letting them avoid extreme actions (large steps). However, y (x) does not appear in the local optimization problem of x (y). Thus, the two players are not taking the presence of their opponent into account when choosing their actions. Accordingly, ICR arises only because of the players' reaction to, rather than anticipation of, each others' actions. We propose to strengthen ICR by using local optimization problems that model the players' *anticipation* of each other's action.

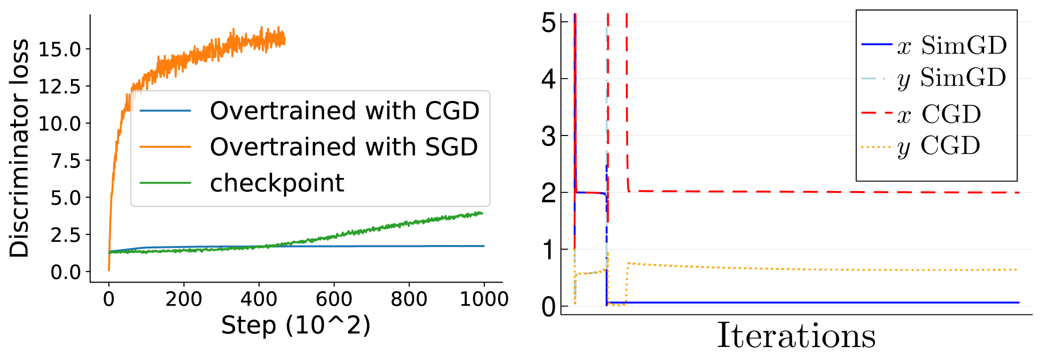


Figure 9.8: **ICR and opponent-awareness.** When training the generator for just a few iterations against the over-trained discriminator of Figure 9.4, the discriminator loss increases rapidly. When attempting to over-train with CGD instead of Adam, the resulting discriminator is even more robust. Similarly, CGD is able to significantly increase the duration for which the generator stays accurate in the (more important) x-direction in Figure 9.7.

Competitive gradient descent. The updates of competitive gradient descent (CGD) described in Chapter 7 are obtained as Nash equilibria of the local game

$$\min_x x^\top \nabla_x f(x_k, y_k) + x^\top [D_{xy} f(x_k, y_k))] y + \frac{\|x\|^2}{2\eta},$$
$$\max_y y^\top \nabla_y f(x_k, y_k) + y^\top [D_{yx} f(x_k, y_k))] x - \frac{\|y\|^2}{2\eta}.$$

Under CGD, the players are aware of each other's presence at every step, since the mixed Hessian $x^\top [D_{xy} f(x_k, y_k)] y$ informs each player how the simultaneous actions of the other player could affect the loss incurred due to their own action. This element of anticipation strengthens ICR, as indicated by the convergence results provided in Chapter 7. Providing additional evidence, we see in Figure 9.8 that attempting to over-train the discriminator using CGD leads to a discriminator that is even more robust than the checkpoint discriminator. Applying CGD to the example of Figure 9.7 also increases the stability of the approximate projection of P_{data} onto $\mathcal{S}$ according to the metric implicit in the discriminator. These results suggest the use of CGD to strengthen ICR in GAN training, which we will investigate in the next section. We also expect methods such as LOLA [86] or SGA [24, 89] to strengthen ICR, but a detailed comparison is beyond the scope of this work.

9.6 Empirical study on CIFAR10

Experimental setup. Based on the last section, CGD strengthens the effects of ICR and should therefore improve GAN performance. We will now investigate this question empirically. In order to make for a fair comparison with Adam, we combine CGD with a simple RMSprop-type heuristic to adjust learning rates, obtaining adaptive CGD (ACGD, see Appendix for details). As loss functions, we use the original GAN loss (OGAN) of (9.1) and the Wasserstein GAN loss function (WGAN) given by

$$\min_{\mathcal{G}} \max_{\mathcal{D}} \ \mathbb{E}_{x \sim P_{\text{data}}} \left[\mathcal{D}(x) \right] - \mathbb{E}_{x \sim P_{\mathcal{G}}} \left[\mathcal{D}(x) \right].$$

When using Adam on OGAN, we stick to the common practice of replacing the generator loss by $\mathbb{E}_{x \sim P_{\mathcal{G}}} \left[-\log \left(\mathcal{D}(x) \right) \right]$, as this has been found to improve training stability [96, 97]. In order to be generous towards existing methods, we use an existing architecture intended for the use with WGAN gradient penalty [106]. As regularizers, we consider no regularization (NOREG), ℓ_2 penalty on the discriminator with different weights (L2), spectral normalization [176] on the discriminator (SN), or 1-centered gradient penalty on the discriminator, following [106] (GP). Following the advice in [97], we train generator and discriminator simultaneously, with the exception of WGAN-GP and Adam, for which we follow [106] in making five discriminator updates per generator update. We use the Pytorch implementation of inception score (IS) [211] to compare generator quality.[3]

Experimental results. We will now summarize our main experimental findings, (see Figure 9.9). **(1)** When restricting our attention to the top-performing models, we observe that the combination of ACGD with the WGAN loss and without any regularization achieves a higher inception score than all other combinations tested. **(2)** The improvement obtained from training with ACGD persists when measuring image quality according to the Frechét-inception-distance (FID) [117]. **(3)** When comparing the number of gradient computations and Hessian-vector products, ACGD is significantly slower than WGAN loss with spectral normalization trained with Adam, because of the iterative solution of the matrix inverse in ACGD's update rule. **(4)** The only instance where we observe erratic behavior with ACGD is when using OGAN without regularization or with a small ℓ_2 penalty. However, ACGD still outperforms Adam in those cases. In particular, training with Adam breaks down completely when using the original saturating loss (as we do for ACGD). **(5)** When plotting the difference between the inception scores obtained by ACGD and Adam for the same model over the number of iterations, for all models, we observe that ACGD often performs significantly better and hardly ever significantly worse.

[3]The Pytorch implementation gives slightly different scores than Tensorflow. We report Tensorflow IS in the Appendix; the relative performance is largely the same.

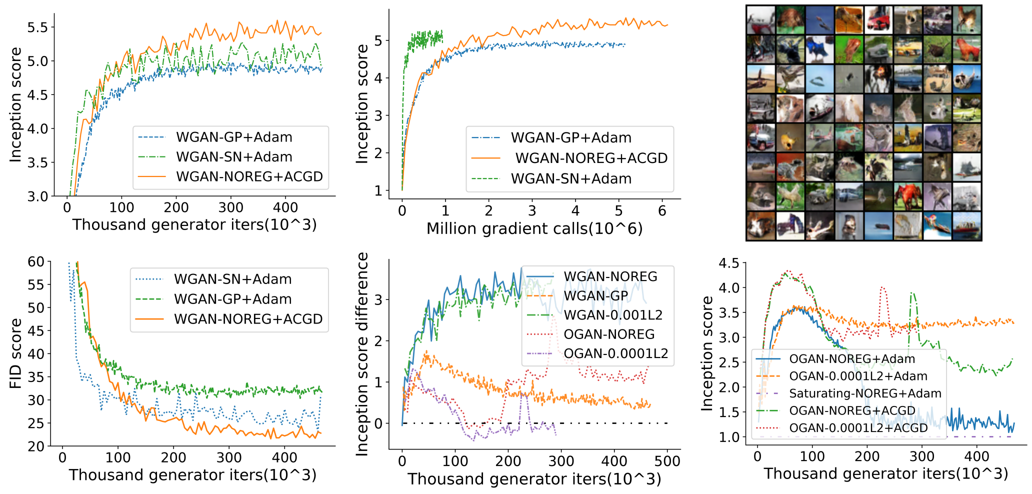


Figure 9.9: **Experiments on CIFAR10.** We plot the inception score (IS) against the number of iterations (first panel) and gradient or Hessian-vector product computation (second panel). In the third panel we show final samples of WGAN trained with ACGD and without explicit regularization. In panel four, we compare measure image quality using the Frechet-inception-distance (FID, smaller is better). The results are consistent with those obtained using IS. In panel five, we plot the difference between inception scores between ACGD and Adam (positive values correspond to a larger score for ACGD) over different iterations and models. The only cases where we observe non-convergence of ACGD are OGAN without regularization or with weight decay of weight 0.0001, as shown in the last panel. The inception score is, however, still higher than for the same model trained with Adam. When using Adam on the original saturating GAN loss (which we used with ACGD), training breaks down completely.

Since CGD strengthens the effects of ICR, the performance improvements obtained with CGD provide further evidence that ICR is a key factor to GAN performance.

BIBLIOGRAPHY

[1] Milton Abramowitz and Irene A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, volume 55 of *National Bureau of Standards Applied Mathematics Series*. U.S. Government Printing Office, Washington, D.C., 1964.

[2] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 22–31. JMLR. org, 2017.

[3] Loyce M. Adams and Harry F. Jordan. Is SOR color-blind? *SIAM Journal on Scientific and Statistical Computing*, 7(2):490–506, 1986.

[4] Loyce M Adams and James M Ortega. A multi-color SOR method for parallel computation. In *ICPP*, pages 53–56. Citeseer, 1982.

[5] Robert A. Adams and John J. F. Fournier. *Sobolev Spaces*, volume 140 of *Pure and Applied Mathematics (Amsterdam)*. Elsevier/Academic Press, Amsterdam, second edition, 2003.

[6] Jonas Adler and Sebastian Lunz. Banach Wasserstein gan. In *Advances in Neural Information Processing Systems*, pages 6754–6763, 2018.

[7] Raymond Alcouffe, Achi Brandt, Joel Dendy, Jr., and James W. Painter. The multi-grid method for the diffusion equation with strongly discontinuous coefficients. *SIAM J. Sci. Stat. Comput.*, 2(4):430–454, 1981.

[8] Shun-ichi Amari. *Information geometry and its applications*, volume 194. Springer, 2016.

[9] Shun-ichi Amari and Hiroshi Nagaoka. *Methods of information geometry*, volume 191. American Mathematical Soc., 2007.

[10] 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.

[11] Sivaram Ambikasaran, Daniel Foreman-Mackey, Leslie Greengard, David W. Hogg, and Michael O'Neil. Fast direct methods for Gaussian processes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(2):252–265, 2016. doi: 10.1109/TPAMI.2015.2448083.

[12] Patrick R. Amestoy, Timothy A. Davis, and Iain S. Duff. An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications*, 17(4):886–905, 1996.

[13] Brandon Anderson, Truong Son Hy, and Risi Kondor. Cormorant: Covariant molecular neural networks. In *Advances in Neural Information Processing Systems*, pages 14510–14519, 2019.

[14] Brian D. O. Anderson and John B. Moore. *Optimal Control: Linear Quadratic Methods*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1990. ISBN 0-13-638560-5.

[15] Martín Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=Hk4_qw5xe.

[16] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223, 2017.

[17] Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. Generalization and equilibrium in generative adversarial nets (gans). In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 224–232. JMLR. org, 2017.

[18] Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. Implicit regularization in deep matrix factorization. *arXiv preprint arXiv:1905.13655*, 2019.

[19] Navid Azizan, Sahin Lale, and Babak Hassibi. Stochastic mirror descent on overparameterized nonlinear models: Convergence, implicit regularization, and generalization. *arXiv preprint arXiv:1906.03830*, 2019.

[20] Ivo Babuška and John E. Osborn. Can a finite element method perform arbitrarily badly? *Math. Comp.*, 69(230):443–462, 2000. doi: 10.1090/S0025-5718-99-01085-6.

[21] Francis R. Bach and Michael I. Jordan. Kernel independent component analysis. *J. Mach. Learn. Res.*, 3(1):1–48, 2003. doi: 10.1162/153244303768966085.

[22] Pierre-Luc Bacon, Florian Schäfer, Clement Gehring, Animashree Anandkumar, and Emma Brunskill. A Lagrangian method for inverse problems in reinforcement learning, 2019.

[23] Christopher T. H Baker. *The numerical treatment of integral equations*. Clarendon Press ;, Oxford [Oxfordshire] :, 1978. URL http://caltech.tind.io/record/383605. Reprinted with corrections.

[24] David Balduzzi, Sebastien Racaniere, James Martens, Jakob Foerster, Karl Tuyls, and Thore Graepel. The mechanics of n-player differentiable games. *arXiv preprint arXiv:1802.05642*, 2018.

[25] Sudipto Banerjee, Alan E. Gelfand, Andrew O. Finley, and Huiyan Sang. Gaussian predictive process models for large spatial data sets. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 70(4):825–848, 2008. doi: 10.1111/j.1467-9868.2008.00663.x.

[26] Jing Yu Bao, Fei Ye, and Ying Yang. Screening effect in isotropic Gaussian processes. *Acta Mathematica Sinica, English Series*, 36(5):512–534, 2020.

[27] Ricardo Baptista, Olivier Zahm, and Youssef Marzouk. An adaptive transport framework for joint and conditional density estimation. *arXiv preprint arXiv:2009.10303*, 2020.

[28] M. Bebendorf and S. Rjasanow. Adaptive low-rank approximation of collocation matrices. *Computing*, 70(1):1–24, 2003. doi: 10.1007/s00607-002-1469-6.

[29] Mario Bebendorf and Wolfgang Hackbusch. Existence of $\mathcal{H}$-matrix approximants to the inverse FE-matrix of elliptic operators with L^∞-coefficients. *Numer. Math.*, 95(1):1–28, 2003. doi: 10.1007/s00211-002-0445-6.

[30] Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

[31] Michele Benzi. Localization in matrix computations: Theory and applications. In Michele Benzi and Valeria Simoncini, editors, *Exploiting Hidden Structure in Matrix Computations: Algorithms and Applications : Cetraro, Italy 2015*, pages 211–317. Springer International Publishing, Cham, 2016. doi: 10.1007/978-3-319-49887-4_4.

[32] Michele Benzi and Valeria Simoncini. Decay bounds for functions of Hermitian matrices with banded or Kronecker structure. *SIAM J. Matrix Anal. Appl.*, 36(3):1263–1282, 2015. doi: 10.1137/151006159.

[33] Michele Benzi and Miroslav Tůma. A comparative study of sparse approximate inverse preconditioners. *Appl. Numer. Math.*, 30(2-3):305–340, 1999. ISSN 0168-9274. doi: 10.1016/S0168-9274(98)00118-4. URL https://doi.org/10.1016/S0168-9274(98)00118-4. Iterative Methods and Preconditioners (Berlin, 1997).

[34] Michele Benzi and Miroslav Tůma. Orderings for factorized sparse approximate inverse preconditioners. *SIAM J. Sci. Comput.*, 21(5):1851–1868, 2000. doi: 10.1137/S1064827598339372.

[35] Hugo Berard, Gauthier Gidel, Amjad Almahairi, Pascal Vincent, and Simon Lacoste-Julien. A closer look at the optimization landscapes of generative adversarial networks. *arXiv preprint arXiv:1906.04848*, 2019.

[36] Benjamin Berkels, Alexander Effland, and Martin Rumpf. Time discrete geodesic paths in the space of images. *SIAM Journal on Imaging Sciences*, 8 (3):1457–1488, 2015.

[37] Dimitris Bertsimas, David B. Brown, and Constantine Caramanis. Theory and applications of robust optimization. *SIAM Rev.*, 53(3):464–501, 2011. ISSN 0036-1445. doi: 10.1137/080734510. URL https://doi.org/10.1137/080734510.

[38] Gregory Beylkin, Ronald Coifman, and Vladimir Rokhlin. Fast wavelet transforms and numerical algorithms. I. *Comm. Pure Appl. Math.*, 44(2): 141–183, 1991. doi: 10.1002/cpa.3160440202.

[39] Mikołaj Bińkowski, Dougal J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying MMD GANs. *arXiv preprint arXiv:1801.01401*, 2018.

[40] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, 2004. doi: 10.1017/CBO9780511804441.

[41] Achi Brandt. Multi-level adaptive techniques (MLAT) for partial differential equations: ideas and software. In *Mathematical software, III (Proc. Sympos., Math. Res. Center, Univ. Wisconsin, Madison, Wis.,1977)*, pages 277–318. Publ. Math. Res. Center, No. 39. Academic Press, New York, 1977.

[42] Achi Brandt, James Brannick, Karsten Kahl, and Irene Livshits. Bootstrap AMG. *SIAM J. Sci. Comput.*, 33(2):612–632, 2011.

[43] Donald L. Brown, Joscha Gedicke, and Daniel Peterseim. Numerical homogenization of heterogeneous fractional Laplacians. *Multiscale Model. Simul.*, 16(3):1305–1332, 2018. doi: 10.1137/17M1147305.

[44] George W. Brown. Iterative solution of games by fictitious play. *Activity analysis of production and allocation*, 13(1):374–376, 1951.

[45] Sat Chatterjee. Coherent gradients: An approach to understanding generalization in gradient descent-based optimization. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=ryeFY0EFwS.

[46] Yanqing Chen, Timothy A. Davis, William W. Hager, and Sivasankaran Rajamanickam. Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. *ACM Trans. Math. Softw.*, 35(3), October 2008. ISSN 0098-3500. doi: 10.1145/1391989.1391995. URL https://doi.org/10.1145/1391989.1391995.

[47] Yifan Chen, Bamdad Hosseini, Houman Owhadi, and Andrew M Stuart. Solving and learning nonlinear pdes with gaussian processes. *arXiv preprint arXiv:2103.12959*, 2021.

[48] Yucheng Chen, Matus Telgarsky, Chao Zhang, Bolton Bailey, Daniel Hsu, and Jian Peng. A gradual, semi-discrete approach to generative network training via explicit wasserstein minimization. *arXiv preprint arXiv:1906.03471*, 2019.

[49] Jean-Paul Chilès and Pierre Delfiner. *Geostatistics: Modeling Spatial Uncertainty*. Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., Hoboken, NJ, second edition, 2012. doi: 10.1002/9781118136188.

[50] Edmond Chow and Aftab Patel. Fine-grained parallel incomplete LU factorization. *SIAM journal on Scientific Computing*, 37(2):C169–C193, 2015.

[51] Edmond Chow and Yousef Saad. Preconditioned Krylov subspace methods for sampling multivariate Gaussian distributions. *SIAM Journal on Scientific Computing*, 36(2):A588–A608, 2014.

[52] Jon Cockayne, Chris Oates, Tim Sullivan, and Mark Girolami. Probabilistic numerical methods for pde-constrained bayesian inverse problems. *AIP Conference Proceedings*, 1853(1):060001, 2017. doi: 10.1063/1.4985359. URL `https://aip.scitation.org/doi/abs/10.1063/1.4985359`.

[53] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International Conference on Machine Learning*, pages 2990–2999, 2016.

[54] Andrew Cotter, Maya Gupta, Heinrich Jiang, Nathan Srebro, Karthik Sridharan, Serena Wang, Blake Woodworth, and Seungil You. Training well-generalizing classifiers for fairness metrics and other data-dependent constraints. *arXiv preprint arXiv:1807.00028*, 2018.

[55] Andrew Cotter, Heinrich Jiang, and Karthik Sridharan. Two-player games for efficient non-convex constrained optimization. *arXiv preprint arXiv:1804.06500*, 2018.

[56] Elizabeth Cuthill and James McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th National Conference*, pages 157–172, 1969.

[57] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

[58] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE, 2005.

[59] Raffaello D'Andrea. Lmi approach to mixed h-2 and h-infinity performance objective controller design. *IFAC Proceedings Volumes*, 29(1):3198–3203, 1996.

[60] Yair Daon and Georg Stadler. Mitigating the influence of the boundary on PDE-based covariance operators. *Inverse Probl. Imaging*, 12(5):1083–1102, 2018. doi: 10.3934/ipi.2018045.

[61] Constantinos Daskalakis, Andrew Ilyas, Vasilis Syrgkanis, and Haoyang Zeng. Training gans with optimism. *arXiv preprint arXiv:1711.00141*, 2017.

[62] Abhirup Datta, Sudipto Banerjee, Andrew O Finley, and Alan E. Gelfand. Hierarchical nearest-neighbor Gaussian process models for large geostatistical datasets. *Journal of the American Statistical Association*, 111(514):800–812, 2016.

[63] Timothy A. Davis. *Direct methods for sparse linear systems*. SIAM, 2006.

[64] Timothy A Davis, Sivasankaran Rajamanickam, and Wissam M. Sid-Lakhdar. A survey of direct methods for sparse linear systems. *Acta Numer.*, 25:383–566, 2016.

[65] S. Dekel and D. Leviatan. The Bramble–Hilbert lemma for convex domains. *SIAM J. Math. Anal.*, 35(5):1203–1212, 2004. doi: 10.1137/S0036141002417589.

[66] S. Dekel and D. Leviatan. The Bramble–Hilbert lemma for convex domains. *SIAM J. Math. Anal.*, 35(5):1203–1212, 2004. doi: 10.1137/S0036141002417589.

[67] Denis Demidov. AMGCL: An efficient, flexible, and extensible algebraic multigrid implementation. *Lobachevskii Journal of Mathematics*, 40(5):535–546, 2019.

[68] Stephen Demko, William F. Moss, and Philip W. Smith. Decay rates for inverses of band matrices. *Math. Comp.*, 43(168):491–499, 1984. doi: 10.2307/2008290.

[69] Persi Diaconis. Bayesian numerical analysis. In *Statistical decision theory and related topics, IV, Vol. 1 (West Lafayette, Ind., 1986)*, pages 163–175. Springer, New York, 1988.

[70] Alexander Effland, Martin Rumpf, and Florian Schäfer. Image extrapolation for the time discrete metamorphosis model: Existence and applications. *SIAM Journal on Imaging Sciences*, 11(1):834–862, 2018.

[71] Shinto Eguchi and John Copas. Interpreting Kullback-Leibler divergence with the Neyman-Pearson lemma. *J. Multivariate Anal.*, 97(9):2034–2040, 2006. ISSN 0047-259X. doi: 10.1016/j.jmva.2006.03.007. URL https://doi.org/10.1016/j.jmva.2006.03.007.

[72] Stanley C. Eisenstat. Efficient implementation of a class of preconditioned conjugate gradient methods. *SIAM Journal on Scientific and Statistical Computing*, 2(1):1–4, 1981.

[73] Ivar Ekeland and Roger Temam. *Convex analysis and variational problems*, volume 28. Siam, 1999.

[74] Howard C. Elman and Elvira Agrón. Ordering techniques for the preconditioned conjugate gradient method on parallel computers. *Computer Physics Communications*, 53(1-3):253–269, 1989.

[75] A. Yu. Eremin, L. Yu. Kolotilina, and A. A. Nikishin. Factorized sparse approximate inverse preconditionings. III. Iterative construction of preconditionings. *Zap. Nauchn. Sem. S.-Peterburg. Otdel. Mat. Inst. Steklov. (POMI)*, 248(Chisl. Metody i Vopr. Organ. Vychisl. 13):17–48, 247, 1998. ISSN 0373-2703. doi: 10.1007/BF02672769. URL https://doi.org/10.1007/BF02672769.

[76] Francisco Facchinei and Jong-Shi Pang. *Finite-dimensional variational inequalities and complementarity problems. Vol. II*. Springer Series in Operations Research. Springer-Verlag, New York, 2003. ISBN 0-387-95581-X.

[77] Yuwei Fan, Cindy Orozco Bohorquez, and Lexing Ying. BCR-Net: A neural network based on the nonstandard wavelet form. *Journal of Computational Physics*, 384:1–15, 2019.

[78] Yuwei Fan, Jordi Feliu-Faba, Lin Lin, Lexing Ying, and Leonardo Zepeda-Núnez. A multiscale neural network based on hierarchical nested bases. *Research in the Mathematical Sciences*, 6(2):1–28, 2019.

[79] Yuwei Fan, Lin Lin, Lexing Ying, and Leonardo Zepeda-Núnez. A multiscale neural network based on hierarchical matrices. *Multiscale Modeling & Simulation*, 17(4):1189–1213, 2019.

[80] G. E. Fasshauer. Meshfree methods. *Handbook of theoretical and computational nanotechnology, American Scientific Publishers*, 27:33–97, 2006.

[81] R. P. Fedorenko. A relaxation method of solution of elliptic difference equations. *Ž. Vyčisl. Mat. i Mat. Fiz.*, 1:922–927, 1961.

[82] Michael Feischl and Daniel Peterseim. Sparse compression of expected solution operators, 2018. arXiv:1807.01741.

[83] Massimiliano Ferronato, Carlo Janna, and Giuseppe Gambolati. A novel factorized sparse approximate inverse preconditioner with supernodes. *Procedia Computer Science*, 51:266–275, 2015.

[84] Tanner Fiez, Benjamin Chasnov, and Lillian J Ratliff. Convergence of learning dynamics in Stackelberg games. *arXiv preprint arXiv:1906.01217*, 2019.

[85] Shai Fine and Katya Scheinberg. Efficient SVM training using low-rank kernel representations. *J. Mach. Learn. Res.*, 2:243–264, 2001.

[86] Jakob Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 122–130. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

[87] Charless Fowlkes, Serge Belongie, Fan Chung, and Jitendra Malik. Spectral grouping using the Nyström method. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(2):214–225, 2004. doi: 10.1109/TPAMI.2004.1262185.

[88] Reinhard Furrer, Marc G. Genton, and Douglas Nychka. Covariance tapering for interpolation of large spatial datasets. *J. Comput. Graph. Statist.*, 15(3): 502–523, 2006. doi: 10.1198/106186006X132178.

[89] Ian Gemp and Sridhar Mahadevan. Global convergence to the equilibrium of gans using variational inequalities. *arXiv preprint arXiv:1808.01531*, 2018.

[90] Alan George. Nested dissection of a regular finite element mesh. *SIAM Journal on Numerical Analysis*, 10(2):345–363, 1973.

[91] Alan George and Joseph W. H. Liu. The evolution of the minimum degree ordering algorithm. *SIAM Rev.*, 31(1):1–19, 1989. doi: 10.1137/1031001.

[92] John R. Gilbert and Robert Endre Tarjan. The analysis of a nested dissection algorithm. *Numer. Math.*, 50(4):377–404, 1987. doi: 10.1007/BF01396660.

[93] Andrew Gilpin, Samid Hoda, Javier Pena, and Tuomas Sandholm. Gradient-based algorithms for finding nash equilibria in extensive form games. In *International Workshop on Web and Internet Economics*, pages 57–69. Springer, 2007.

[94] D. Gines, G. Beylkin, and J. Dunn. *LU* factorization of non-standard forms and direct multiresolution solvers. *Appl. Comput. Harmon. Anal.*, 5(2):156–201, 1998. doi: 10.1006/acha.1997.0227.

[95] Tilmann Gneiting and Martin Schlather. Stochastic models that separate fractal dimension and the Hurst effect. *SIAM Rev.*, 46(2):269–282, 2004. doi: 10.1137/S0036144501394387.

[96] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[97] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

[98] Ivan G Graham, Frances Y Kuo, Dirk Nuyens, Robert Scheichl, and Ian H Sloan. Analysis of circulant embedding methods for sampling stationary random fields. *SIAM Journal on Numerical Analysis*, 56(3):1871–1895, 2018.

[99] Joseph F Grcar. Mathematicians of gaussian elimination. *Notices of the AMS*, 58(6):782–792, 2011.

[100] Leslie Greengard and Vladimir Rokhlin. A fast algorithm for particle simulations. *J. Comput. Phys.*, 73(2):325–348, 1987. doi: 10.1016/0021-9991(87)90140-9.

[101] Paulina Grnarova, Kfir Y Levy, Aurelien Lucchi, Thomas Hofmann, and Andreas Krause. An online learning approach to generative adversarial networks. *arXiv preprint arXiv:1706.03269*, 2017.

[102] Marcus J. Grote and Thomas Huckle. Parallel preconditioning with sparse approximate inverses. *SIAM J. Sci. Comput.*, 18(3):838–853, 1997.

[103] Peter D. Grünwald, A. Philip Dawid, et al. Game theory, maximum entropy, minimum discrepancy and robust bayesian decision theory. *Annals of Statistics*, 32(4):1367–1433, 2004.

[104] Joseph Guinness. Permutation methods for sharpening Gaussian process approximations. *Technometrics*, 60(4):415–429, 2018. doi: 10.1080/00401706.2018.1437476.

[105] Joseph Guinness. Permutation and grouping methods for sharpening Gaussian process approximations. *Technometrics*, 60(4):415–429, 2018.

[106] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

[107] Suriya Gunasekar, Blake E. Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nati Srebro. Implicit regularization in matrix factorization. In *Advances in Neural Information Processing Systems*, pages 6151–6159, 2017.

[108] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1321–1330. JMLR. org, 2017.

[109] Peter Guttorp and Tilmann Gneiting. Studies in the history of probability and statistics. XLIX. On the Matérn correlation family. *Biometrika*, 93(4): 989–995, 2006. doi: 10.1093/biomet/93.4.989.

[110] Bernard Haasdonk and Hans Burkhardt. Invariant kernel functions for pattern analysis and machine learning. *Machine Learning*, 68(1):35–61, 2007.

[111] W. Hackbusch. A fast iterative method for solving Poisson's equation in a general region. In *Numerical treatment of differential equations (Proc. Conf., Math. Forschungsinst., Oberwolfach, 1976)*, pages 51–62. Lecture Notes in Math., Vol. 631. Springer, Berlin, 1978.

[112] Wolfgang Hackbusch. A sparse matrix arithmetic based on $\mathcal{H}$-matrices. I. Introduction to $\mathcal{H}$-matrices. *Computing*, 62(2):89–108, 1999. doi: 10.1007/s006070050015.

[113] Wolfgang Hackbusch. *Multi-Grid Methods and Applications*, volume 4 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin Heidelberg, 2013. doi: 10.1007/978-3-662-02427-0.

[114] Wolfgang Hackbusch and Steffen Börm. Data-sparse approximation by adaptive $\mathcal{H}^2$-matrices. *Computing*, 69(1):1–35, 2002. doi: 10.1007/s00607-002-1450-4.

[115] Wolfgang Hackbusch and Boris N. Khoromskij. A sparse $\mathcal{H}$-matrix arithmetic. II. Application to multi-dimensional problems. *Computing*, 64(1): 21–47, 2000.

[116] Philipp Hennig, Michael A. Osborne, and Mark Girolami. Probabilistic numerics and uncertainty in computations. *Proc. A.*, 471(2179):20150142, 17, 2015. ISSN 1364-5021. doi: 10.1098/rspa.2015.0142. URL `https://doi.org/10.1098/rspa.2015.0142`.

[117] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

[118] Kenneth L. Ho and Lexing Ying. Hierarchical interpolative factorization for elliptic operators: Integral equations. *Comm. Pure Appl. Math.*, 69(7): 1314–1353, 2016. doi: 10.1002/cpa.21577.

[119] Roger A. Horn and Charles R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, Cambridge, 1994. doi: 10.1017/CBO9780511840371. Corrected reprint of the 1991 original.

[120] Thomas Y. Hou and Pengchuan Zhang. Sparse operator compression of higher-order elliptic operators with rough coefficients. *Res. Math. Sci.*, 4: Paper No. 24, 49, 2017. doi: 10.1186/s40687-017-0113-1.

[121] Thomas Y. Hou, De Huang, Ka Chun Lam, and Pengchuan Zhang. An adaptive fast solver for a general class of positive definite matrices via energy decomposition. *Multiscale Modeling & Simulation*, 16(2):615–678, 2018.

[122] Gabriel Huang, Hugo Berard, Ahmed Touati, Gauthier Gidel, Pascal Vincent, and Simon Lacoste-Julien. Parametric adversarial divergences are good task losses for generative modeling. *arXiv preprint arXiv:1708.02511*, 2017.

[123] Hua Huang, Xin Xing, and Edmond Chow. H2pack: High-performance h 2 matrix package for kernel matrices using the proxy point method. *ACM Transactions on Mathematical Software (TOMS)*, 47(1):1–29, 2020.

[124] Peter J. Huber and Elvezio M. Ronchetti. *Robust statistics*. Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., Hoboken, NJ, second edition, 2009. ISBN 978-0-470-12990-6. doi: 10.1002/9780470434697. URL https://doi.org/10.1002/9780470434697.

[125] Thore Husfeldt. Graph colouring algorithms. In *Topics in Chromatic Graph Theory*, volume 156 of *Encyclopedia Math. Appl.*, pages 277–303. Cambridge Univ. Press, Cambridge, 2015. doi: 10.1017/CBO9781139519793.016.

[126] Alfredo N. Iusem. On the convergence properties of the projected gradient method for convex optimization. *Computational & Applied Mathematics*, 22 (1):37–52, 2003.

[127] Takeshi Iwashita and Masaaki Shimasaki. Block red-black ordering: A new ordering strategy for parallelization of ICCG method. *Int. J. Parallel Program.*, 31(1):55–75, 2003. doi: 10.1023/A:1021738303840.

[128] Stephane Jaffard. Propriétés des matrices "bien localisées" près de leur diagonale et quelques applications. *Ann. Inst. H. Poincaré Anal. Non Linéaire*, 7(5):461–476, 1990.

[129] W. James and C. Stein. Estimation with quadratic loss. *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability*, 1: 361–379, 1961. doi: 10.1177/0278364907080252.

[130] Chi Jin, Praneeth Netrapalli, and Michael I Jordan. Minmax optimization: Stable limit points of gradient descent ascent are locally optimal. *arXiv preprint arXiv:1902.00618*, 2019.

[131] Andre G. Journel and Ch. J. Huijbregts. *Mining Geostatistics*. Academic Press, 1978.

[132] Olav Kallenberg and Olav Kallenberg. *Foundations of modern probability*, volume 2. Springer, 1997.

[133] I. E. Kaporin. An alternative approach to the estimation of the number of iterations in the conjugate gradient method. In *Numerical Methods and Software (Russian)*, pages 55–72. Akad. Nauk SSSR, Otdel Vychisl. Mat., Moscow, 1990.

[134] Matthias Katzfuss. A multi-resolution approximation for massive spatial datasets. *J. Amer. Stat. Assoc.*, 2016. doi: 10.1080/01621459.2015.1123632.

[135] Matthias Katzfuss and Joseph Guinness. A general framework for Vecchia approximations of Gaussian processes. *Statistical Science*, forthcoming, 2019. URL `http://arxiv.org/abs/1708.06302`.

[136] Matthias Katzfuss, Joseph Guinness, Wenlong Gong, and Daniel Zilber. Vecchia approximations of Gaussian-process predictions. *arXiv:1805.03309*, 2018.

[137] Yuehaw Khoo and Lexing Ying. Switchnet: A neural network model for forward and inverse scattering problems. *SIAM Journal on Scientific Computing*, 41(5):A3182–A3201, 2019.

[138] George S. Kimeldorf and Grace Wahba. A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. *The Annals of Mathematical Statistics*, 41(2):495–502, 1970.

[139] Naveen Kodali, Jacob Abernethy, James Hays, and Zsolt Kira. On convergence and stability of gans. *arXiv preprint arXiv:1705.07215*, 2017.

[140] Andrey Nikolaevich Kolmogorov. The representation of continuous functions of several variables by superpositions of continuous functions of a smaller number of variables. *Doklady Akademii Nauk SSSR*, 108(2):179–182, 1956.

[141] L. Yu. Kolotilina and A. Yu. Yeremin. Factorized sparse approximate inverse preconditionings. I. Theory. *SIAM J. Matrix Anal. Appl.*, 14(1):45–58, 1993. ISSN 0895-4798. doi: 10.1137/0614004. URL `https://doi.org/10.1137/0614004`.

[142] Ralf Kornhuber, Daniel Peterseim, and Harry Yserentant. An analysis of a class of variational multiscale methods based on subspace decomposition. *Math. Comp.*, 87(314):2765–2774, 2018. doi: 10.1090/mcom/3302. URL `https://doi.org/10.1090/mcom/3302`.

[143] G. M. Korpelevich. Extragradient method for finding saddle points and other problems. *Matekon*, 13(4):35–49, 1977.

[144] Ilya Krishtal, Thomas Strohmer, and Tim Wertz. Localization of matrix factorizations. *Found. Comput. Math.*, 15(4):931–951, 2015. doi: 10.1007/s10208-014-9196-x.

[145] Masayoshi Kubo, Ryotaro Banno, Hidetaka Manabe, and Masataka Minoji. Implicit regularization in over-parameterized neural networks. *arXiv preprint arXiv:1903.01997*, 2019.

[146] F. M. Larkin. Gaussian measure in Hilbert space and applications in numerical analysis. *Rocky Mountain J. Math.*, 2(3):379–421, 1972. ISSN 0035-7596. doi: 10.1216/RMJ-1972-2-3-379. URL https://doi.org/10.1216/RMJ-1972-2-3-379.

[147] Alistair Letcher, David Balduzzi, Sébastien Racanière, James Martens, Jakob Foerster, Karl Tuyls, and Thore Graepel. Differentiable game mechanics. *Journal of Machine Learning Research*, 20(84):1–40, 2019. URL http://jmlr.org/papers/v20/19-008.html.

[148] Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. Mmd gan: Towards deeper understanding of moment matching network. In *Advances in Neural Information Processing Systems*, pages 2203–2213, 2017.

[149] Jerry Li, Aleksander Madry, John Peebles, and Ludwig Schmidt. On the limitations of first-order approximation in GAN dynamics. *arXiv preprint arXiv:1706.09884*, 2017.

[150] Shengguo Li, Ming Gu, Cinna Julie Wu, and Jianlin Xia. New efficient and robust HSS Cholesky factorization of SPD matrices. *SIAM Journal on Matrix Analysis and Applications*, 33(3):886–904, 2012.

[151] Yujia Li, Kevin Swersky, and Rich Zemel. Generative moment matching networks. In *International Conference on Machine Learning*, pages 1718–1727, 2015.

[152] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[153] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020.

[154] Tengyuan Liang and James Stokes. Interaction matters: A note on non-asymptotic local convergence of generative adversarial networks. *arXiv preprint arXiv:1802.06132*, 2018.

[155] Finn Lindgren, Håvard Rue, and Johan Lindström. An explicit link between Gaussian fields and Gaussian Markov random fields: The stochastic partial differential equation approach. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 73(4): 423–498, 2011. doi: 10.1111/j.1467-9868.2011.00777.x.

[156] Richard J. Lipton, Donald J. Rose, and Robert Endre Tarjan. Generalized nested dissection. *SIAM journal on numerical analysis*, 16(2):346–358, 1979.

[157] Bo Liu, Ji Liu, Mohammad Ghavamzadeh, Sridhar Mahadevan, and Marek Petrik. Proximal gradient temporal difference learning algorithms. In *IJCAI*, pages 4195–4199, 2016.

[158] Joseph W. H. Liu, Esmond G. Ng, and Barry W. Peyton. On finding supernodes for sparse matrix computations. *SIAM J. Matrix Anal. Appl.*, 14(1): 242–252, 1993. doi: 10.1137/0614019.

[159] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are GANs created equal? a large-scale study. *arXiv preprint arXiv:1711.10337*, 2017.

[160] Cong Ma, Kaizheng Wang, Yuejie Chi, and Yuxin Chen. Implicit regularization in nonconvex statistical estimation: Gradient descent converges linearly for phase retrieval, matrix completion and blind deconvolution. *arXiv preprint arXiv:1711.10467*, 2017.

[161] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[162] Stephane G Mallat. A theory for multiresolution signal decomposition: the wavelet representation. In *Fundamental Papers in Wavelet Theory*, pages 494–513. Princeton University Press, 2009.

[163] Axel Målqvist and Daniel Peterseim. Localization of elliptic multiscale problems. *Math. Comp.*, 83(290):2583–2603, 2014. doi: 10.1090/S0025-5718-2014-02868-8.

[164] Per-Gunnar Martinsson. Compressing rank-structured matrices via randomized sampling. *SIAM J. Sci. Comput.*, 38(4):A1959–A1986, 2016. doi: 10.1137/15M1016679.

[165] Per-Gunnar Martinsson and Joel Tropp. Randomized numerical linear algebra: Foundations & algorithms. *arXiv preprint arXiv:2002.01387*, 2020.

[166] Youssef Marzouk, Tarek Moselhy, Matthew Parno, and Alessio Spantini. Sampling via measure transport: An introduction. In *Handbook of uncertainty quantification. Vol. 1, 2, 3*, pages 785–825. Springer, Cham, 2017.

[167] Bertil Matérn. *Spatial Variation: Stochastic Models and Their Application to Some Problems in Forest Surveys and Other Sampling Investigations*. Meddelanden Fran Statens Skogsforskningsinstitut, Band 49, Nr.5, Stockholm, 1960.

[168] Eric Mazumdar and Lillian J. Ratliff. On the convergence of gradient-based learning in continuous games. *arXiv preprint arXiv:1804.05464*, 2018.

[169] Eric V. Mazumdar, Michael I. Jordan, and S. Shankar Sastry. On finding local nash equilibria (and only local nash equilibria) in zero-sum games. *arXiv preprint arXiv:1901.00838*, 2019.

[170] J. A. Meijerink and H. A. van der Vorst. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Math. Comp.*, 31(137):148–162, 1977. doi: 10.2307/2005786.

[171] Panayotis Mertikopoulos, Bruno Lecouat, Houssam Zenati, Chuan-Sheng Foo, Vijay Chandrasekhar, and Georgios Piliouras. Optimistic mirror descent in saddle-point problems: Going the extra (gradient) mile. *arXiv preprint arXiv:1807.02629*, 2018.

[172] Panayotis Mertikopoulos, Houssam Zenati, Bruno Lecouat, Chuan-Sheng Foo, Vijay Chandrasekhar, and Georgios Piliouras. Optimistic mirror descent in saddle-point problems: Going the extra (gradient) mile. In *ICLR'19: Proceedings of the 2019 International Conference on Learning Representations*, 2019.

[173] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. The numerics of gans. In *Advances in Neural Information Processing Systems*, pages 1825–1835, 2017.

[174] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*, 2016.

[175] Sobhan Miryoosefi, Kianté Brantley, Hal Daume III, Miro Dudik, and Robert E. Schapire. Reinforcement learning with convex constraints. In *Advances in Neural Information Processing Systems*, pages 14070–14079, 2019.

[176] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[177] Youssef Mroueh, Chun-Liang Li, Tom Sercu, Anant Raj, and Yu Cheng. Sobolev gan. *arXiv preprint arXiv:1711.04894*, 2017.

[178] Harikrishna Narasimhan, Andrew Cotter, and Maya Gupta. Optimizing generalized rate metrics with three players. In *Advances in Neural Information Processing Systems*, pages 10746–10757, 2019.

[179] Maxim Naumov. Parallel incomplete-LU and Cholesky factorization in the preconditioned iterative methods on the GPU. *Nvidia Technical Report NVR-2012-003*, 2012.

[180] A. S. Nemirovsky and D. B. and Yudin. *Problem complexity and method efficiency in optimization.* A Wiley-Interscience Publication. John Wiley & Sons, Inc., New York, 1983. ISBN 0-471-10345-4. Translated from the

Russian and with a preface by E. R. Dawson, Wiley-Interscience Series in Discrete Mathematics.

[181] Yu Nesterov. Excessive gap technique in nonsmooth convex minimization. *SIAM Journal on Optimization*, 16(1):235–249, 2005.

[182] Yurii Nesterov and B. T. Polyak. Cubic regularization of Newton method and its global performance. *Math. Program.*, 108(1, Ser. A):177–205, 2006. ISSN 0025-5610. doi: 10.1007/s10107-006-0706-8. URL https://doi.org/10.1007/s10107-006-0706-8.

[183] Behnam Neyshabur. Implicit regularization in deep learning. *arXiv preprint arXiv:1709.01953*, 2017.

[184] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani. *Algorithmic Game Theory*. Cambridge university press, 2007.

[185] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York, second edition, 2006. ISBN 978-0387-30303-1; 0-387-30303-0.

[186] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-GAN: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems*, pages 271–279, 2016.

[187] Houman Owhadi. Bayesian numerical homogenization. *Multiscale Model. Simul.*, 13(3):812–828, 2015. ISSN 1540-3459. doi: 10.1137/140974596. URL https://doi.org/10.1137/140974596.

[188] Houman Owhadi. Multigrid with rough coefficients and multiresolution operator decomposition from hierarchical information games. *SIAM Rev.*, 59 (1):99–149, 2017. doi: 10.1137/15M1013894.

[189] Houman Owhadi and Clint Scovel. Universal scalable robust solvers from computational information games and fast eigenspace adapted multiresolution analysis, 2017. arXiv:1703.10761.

[190] Houman Owhadi and Clint Scovel. *Operator-Adapted Wavelets, Fast Solvers, and Numerical Homogenization: From a Game Theoretic Approach to Numerical Approximation and Algorithm Design*, volume 35 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, 2019. doi: 10.1017/9781108594967.

[191] Houman Owhadi, Lei Zhang, and Leonid Berlyand. Polyharmonic homogenization, rough polyharmonic splines and sparse super-localization. *ESAIM Math. Model. Numer. Anal.*, 48(2):517–552, 2014. doi: 10.1051/m2an/2013118.

[192] Dianne P O'Leary. Ordering schemes for parallel processing of certain mesh problems. *SIAM Journal on Scientific and Statistical Computing*, 5(3):620–632, 1984.

[193] I. Palasti and A. Renyi. On interpolation theory and the theory of games. *MTA Mat. Kat. Int. Kozl*, 1:529–540, 1956.

[194] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 16–17, 2017.

[195] Barak A. Pearlmutter. Fast exact multiplication by the Hessian. *Neural Computation*, 6(1):147–160, 1994.

[196] David Pfau and Oriol Vinyals. Connecting generative adversarial networks and actor-critic methods. *arXiv preprint arXiv:1610.01945*, 2016.

[197] Henri Poincaré. *Calcul des probabilités*. Les Grands Classiques Gauthier-Villars. [Gauthier-Villars Great Classics]. Éditions Jacques Gabay, Sceaux, 1987. ISBN 2-87647-001-2. Reprint of the second (1912) edition.

[198] Manish Prajapat, Kamyar Azizzadenesheli, Alexander Liniger, Yisong Yue, and Anima Anandkumar. Competitive policy optimization. *arXiv preprint arXiv:2006.10611*, 2020.

[199] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate Gaussian process regression. *J. Mach. Learn. Res.*, 6 (Dec):1939–1959, 2005.

[200] 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.

[201] Alexander Rakhlin and Karthik Sridharan. Online learning with predictable sequences. In *Conference on Learning Theory*, pages 993–1019. PMLR, 2013.

[202] Garvesh Raskutti and Sayan Mukherjee. The information geometry of mirror descent. *IEEE Transactions on Information Theory*, 61(3):1451–1457, 2015.

[203] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, 2006. doi: 10.7551/mitpress/3206.001.0001.

[204] J. Revels, M. Lubin, and T. Papamarkou. Forward-mode automatic differentiation in julia. *arXiv:1607.07892 [cs.MS]*, 2016. URL https://arxiv.org/abs/1607.07892.

[205] Lassi Roininen, Markku S. Lehtinen, Sari Lasanen, Mikko Orispää, and Markku Markkanen. Correlation priors. *Inverse Probl. Imaging*, 5(1):167–184, 2011. doi: 10.3934/ipi.2011.5.167.

[206] Lassi Roininen, Petteri Piiroinen, and Markku Lehtinen. Constructing continuous stationary covariances as limits of the second-order stochastic difference equations. *Inverse Probl. Imaging*, 7(2):611–647, 2013. doi: 10.3934/ipi.2013.7.611.

[207] Lassi Roininen, Janne M. J. Huttunen, and Sari Lasanen. Whittle–Matérn priors for Bayesian statistical inversion with applications in electrical impedance tomography. *Inverse Probl. Imaging*, 8(2):561–586, 2014. doi: 10.3934/ipi.2014.8.561.

[208] Kevin Roth, Aurelien Lucchi, Sebastian Nowozin, and Thomas Hofmann. Stabilizing training of generative adversarial networks through regularization. In *Advances in Neural Information Processing Systems*, pages 2018–2028, 2017.

[209] Edward Rothberg and Anoop Gupta. An efficient block-oriented approach to parallel sparse Cholesky factorization. *SIAM J. Sci. Comput.*, 15(6):1413–1439, 1994. doi: 10.1137/0915085.

[210] Yousef Saad. *Iterative methods for sparse linear systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, second edition, 2003. ISBN 0-89871-534-2. doi: 10.1137/1.9780898718003. URL https://doi.org/10.1137/1.9780898718003.

[211] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

[212] Huiyan Sang and Jianhua Z. Huang. A full scale approximation of covariance functions for large spatial data sets. *J. R. Stat. Soc. Ser. B. Stat. Methodol.*, 74(1):111–132, 2012. doi: 10.1111/j.1467-9868.2011.01007.x.

[213] Arthur Sard. *Linear approximation*. American Mathematical Society, Providence, R.I., 1963.

[214] Stefan A. Sauter and Christoph Schwab. *Boundary Element Methods*, volume 39 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin Heidelberg, 2011. doi: 10.1007/978-3-540-68093-2.

[215] Anton Schwaighofer and Volker Tresp. Transductive and inductive methods for approximate Gaussian process regression. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15 (NIPS 2002)*, pages 977–984, 2003.

[216] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

[217] Shai Shalev-Shwartz and Yoram Singer. Convex repeated games and Fenchel duality. In *Advances in Neural Information Processing Systems*, pages 1265–1272, 2007.

[218] Jonathan Richard Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, Carnegie-Mellon University. Department of Computer Science, 1994. URL https://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf.

[219] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[220] Patrice Y. Simard, Yann A. LeCun, John S. Denker, and Bernard Victorri. Transformation invariance in pattern recognition—tangent distance and tangent propagation. In *Neural Networks: Tricks of the Trade*, pages 239–274. Springer, 1998.

[221] Alex J. Smola and Peter L. Bartlett. Sparse greedy Gaussian process regression. In *Advances in Neural Information Processing Systems 13 (NIPS 2000)*, pages 619–625, 2001. URL https://papers.nips.cc/paper/1880-sparse-greedy-gaussian-process-regression.

[222] Alex J. Smola and Bernhard Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 911–918, 2000.

[223] Edward Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. In Y. Weiss, P. B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1257–1264. MIT Press, 2006. URL http://papers.nips.cc/paper/2857-sparse-gaussian-processes-using-pseudo-inputs.

[224] Yan Song, Ian Vince McLoughlin, and Li-Rong Dai. Local coding based matching kernel method for image classification. *PloS one*, 9(8), 2014.

[225] Michael L Stein. Fast and exact simulation of fractional brownian surfaces. *Journal of Computational and Graphical Statistics*, 11(3):587–599, 2002.

[226] Michael L. Stein, Zhiyi Chi, and Leah J. Welty. Approximating likelihoods for large spatial data sets. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 66(2): 275–296, 2004. doi: 10.1046/j.1369-7412.2003.05512.x.

[227] Michael L. Stein et al. The screening effect in kriging. *Annals of Statistics*, 30(1):298–323, 2002.

[228] Michael L. Stein et al. 2010 Rietz lecture: When does the screening effect hold? *The Annals of Statistics*, 39(6):2795–2819, 2011.

[229] Russell Stewart and Stefano Ermon. Label-free supervision of neural networks with physics and domain knowledge. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[230] Klaus Stüben. Algebraic multigrid (amg): An introduction with applications. *Multigrid*, 2000.

[231] A. V. Sul' din. Wiener measure and its applications to approximation methods. I. *Izv. Vysš. Učebn. Zaved. Matematika*, 1959(6 (13)):145–158, 1959. ISSN 0021-3446.

[232] A. V. Sul' din. Wiener measure and its applications to approximation methods. II. *Izv. Vysš. Učebn. Zaved. Matematika*, 1960(5 (18)):165–179, 1960. ISSN 0021-3446.

[233] Ying Sun and Michael L. Stein. Statistically and computationally efficient estimating equations for large spatial datasets. *Journal of Computational and Graphical Statistics*, 25(1):187–208, 2016. ISSN 1061-8600. doi: 10.1080/10618600.2014.975230.

[234] Joseph Teran, Eftychios Sifakis, Geoffrey Irving, and Ronald Fedkiw. Robust quasistatic finite elements and flesh simulation. *Symp. on Comp. Anim.*, pages 181–190, 2005.

[235] J. F. Traub, G. W. Wasilkowski, and H. Woźniakowski. *Information-based complexity*. Computer Science and Scientific Computing. Academic Press, Inc., Boston, MA, 1988. ISBN 0-12-697545-0. With contributions by A. G. Werschulz and T. Boult.

[236] Lloyd N. Trefethen and David Bau III. *Numerical Linear Algebra*, volume 50. Siam, 1997.

[237] Trilinos. *The Trilinos Project Website*, 2020. URL https://trilinos.github.io.

[238] Alain Trouvé and Laurent Younes. Metamorphoses through Lie group action. *Foundations of Computational Mathematics*, 5(2):173–198, 2005.

[239] Ananya Uppal, Shashank Singh, and Barnabas Poczos. Nonparametric density estimation &amp; convergence rates for gans under besov ipm losses. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 9086–9097. Curran Associates, Inc., 2019. URL

http://papers.nips.cc/paper/9109-nonparametric-density-estimation-convergence-rates-for-gans-under-besov-ipm-losses.pdf.

[240] Petr Vaněk, Jan Mandel, and Marian Brezina. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. *Computing*, 56(3):179–196, 1996.

[241] AV Vecchia. Estimation and model identification for continuous spatial processes. *Journal of the Royal Statistical Society, Series B*, 50(2):297–312, 1988.

[242] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3540–3549. JMLR. org, 2017.

[243] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in Starcraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[244] John von Neumann and Herman H Goldstine. Numerical inverting of matrices of high order. *Bulletin of the American Mathematical Society*, 53(11):1021–1099, 1947.

[245] Greg Wayne and L. F. Abbott. Hierarchical control using networks trained with higher-level forward models. *Neural computation*, 26(10):2163–2193, 2014.

[246] Holger Wendland. Meshless Galerkin methods using radial basis functions. *Mathematics of Computation*, 68(228):1521–1531, 1999.

[247] Holger Wendland. *Scattered data approximation*, volume 17 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, 2005. ISBN 978-0521-84335-5; 0-521-84335-9.

[248] Peter Whittle. On stationary processes in the plane. *Biometrika*, 41:434–449, 1954. doi: 10.1093/biomet/41.3-4.434.

[249] Peter Whittle. Stochastic processes in several dimensions. *Bull. Inst. Internat. Statist.*, 40(2):974–994, 1963.

[250] Christopher K. I. Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press, 2001. URL http://papers.nips.cc/paper/1866-using-the-nystrom-method-to-speed-up-kernel-machines.

[251] 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.

[252] Xin Xing, Hua Huang, and Edmond Chow. Efficient construction of an HSS preconditioner for symmetric positive definite $\mathcal{H}^2$ matrices. *arXiv preprint arXiv:2011.07632*, 2020.

[253] Jinchao Xu and Ludmil Zikatanov. Algebraic multigrid methods. *Acta Numerica*, 26:591–721, 2017.

[254] Abhay Yadav, Sohil Shah, Zheng Xu, David Jacobs, and Tom Goldstein. Stabilizing adversarial nets with prediction methods. *arXiv preprint arXiv:1705.07364*, 2017.

[255] Irad Yavneh. Why multigrid methods are so efficient. *Computing in Science Engineering*, 8(6):12–22, 2006.

[256] Jing Yu, Clement Gehring, Florian Schäfer, and Anima Anandkumar. Robust reinforcement learning: A constrained game-theoretic approach. *To be presented at Learning for Decision and Control (L4DC) 2021.*, 2021.

[257] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

[258] Fuzhen Zhang, editor. *The Schur Complement and its Applications*, volume 4 of *Numerical Methods and Algorithms*. Springer-Verlag, New York, 2005. doi: 10.1007/b105056.

[259] Kaiqing Zhang, Zhuoran Yang, and Tamer Basar. Policy optimization provably converges to nash equilibria in zero-sum linear quadratic games. In *Advances in Neural Information Processing Systems*, pages 11598–11610, 2019.

.1 Appendix to Chapter 4

Proof of Lemma 1. Applying Lemma 2 with first block $\Theta_{1:q-1,1:q-1}$ and second block $\Theta_{q,q}$ yields

$$\Theta^{(q)} = \begin{pmatrix} & & & 0 \\ & \mathrm{Id} & & \vdots \\ & & & 0 \\ -B^{(q),-1}A^{(q)}_{q,1} & \cdots & -B^{(q),-1}A^{(q)}_{q,q-1} & \mathrm{Id} \end{pmatrix} \tag{5}$$

$$\begin{pmatrix} & & & 0 \\ & \Theta^{(q-1)} & & \vdots \\ & & & 0 \\ 0 & \cdots & 0 & B^{(q),-1} \end{pmatrix} \begin{pmatrix} & & & -A^{(q)}_{1,q}B^{(q),-\top} \\ & \mathrm{Id} & & \vdots \\ & & & -A^{(q)}_{q-1,q}B^{(q),-\top} \\ 0 & \cdots & 0 & \mathrm{Id} \end{pmatrix}. \tag{6}$$

We now repeat this operation recursively. After the k^{th} step, the central matrix has an upper-left block consisting of $\Theta^{(q-k)}$. We then apply Lemma lem:blockChol2Scale to this upper-left block, with the splitting given by $\Theta_{1:q-k-1,1:q-k-1}$ and $\Theta_{q-k,q-k}$. This reduces the central matrix more and more towards the block-diagonal matrix D, while splitting off a triangular factor to either side.

$$\begin{pmatrix} \Theta^{(q)}_{1,1} & \cdots & \Theta^{(q)}_{1,q-1} & \Theta^{(q)}_{1,q} \\ \vdots & \ddots & \vdots & \vdots \\ \Theta^{(q)}_{q-1,1} & \cdots & \Theta^{(q)}_{q-1,q-1} & \Theta^{(q)}_{2,q} \\ \Theta^{(q)}_{q,1} & \cdots & \Theta^{(q)}_{q,q-1} & \Theta^{(q)}_{q,q} \end{pmatrix} \tag{7}$$

$$= \begin{pmatrix} & & & 0 \\ & \mathrm{Id} & & \vdots \\ & & & 0 \\ -B^{(q),-1}A^{(q)}_{q,1} & \cdots & -B^{(q),-1}A^{(q)}_{q,q-1} & \mathrm{Id} \end{pmatrix} \tag{8}$$

$$\begin{pmatrix} & & & 0 & 0 \\ & \mathrm{Id} & & \vdots & \vdots \\ & & & 0 & \vdots \\ -B^{(q-1),-1}A^{(q-1)}_{q-1,1} & \cdots & -B^{(q-1),-1}A^{(q-1)}_{q-1,q-2} & \mathrm{Id} & 0 \\ 0 & \cdots & \cdots & 0 & \mathrm{Id} \end{pmatrix} \cdots \tag{9}$$

$$\cdots \begin{pmatrix} \mathrm{Id} & 0 & 0 \\ -B^{(2),-1}A^{(2)}_{2,1} & \mathrm{Id} & \vdots \\ 0 & \cdots & \mathrm{Id} \end{pmatrix} \tag{10}$$

$$\begin{pmatrix} B^{(1),-1} & 0 & \cdots & \cdots & 0 \\ 0 & B^{(2),-1} & \ddots & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & 0 & \ddots & B^{(q-1),-1} & 0 \\ 0 & 0 & \cdots & 0 & B^{(q),-1} \end{pmatrix} \begin{pmatrix} \mathrm{Id} & -B^{(2)}_{1,2}B^{(2),-T}_{2,2} & 0 \\ 0 & \mathrm{Id} & \vdots \\ 0 & \cdots & \mathrm{Id} \end{pmatrix} \cdots \tag{11}$$

$$\cdots \begin{pmatrix} & & & -A^{(q-1)}_{1,q-1}B^{(q-1),-\top} & 0 \\ & \mathrm{Id} & & \vdots & \vdots \\ & & & -A^{(q-1)}_{q-2,q-1}B^{(q-1),-\top} & \vdots \\ 0 & \cdots & 0 & \mathrm{Id} & 0 \\ 0 & \cdots & \cdots & 0 & \mathrm{Id} \end{pmatrix} \begin{pmatrix} & & & -A^{(q)}_{1,q}B^{(q),-\top} \\ & \mathrm{Id} & & \vdots \\ & & & -A^{(q)}_{q-1,q}B^{(q),-\top} \\ 0 & \cdots & 0 & \mathrm{Id} \end{pmatrix}. \tag{12}$$

We now combine the lower-triangular factors, obtaining

$$\begin{pmatrix} \mathrm{Id} & -A^{(2)}_{1,2}B^{(2),-\top} & 0 \\ 0 & \mathrm{Id} & \vdots \\ 0 & \cdots & \mathrm{Id} \end{pmatrix} \cdots \begin{pmatrix} & & & -A^{(q)}_{1,q}B^{(q),-\top} \\ & \mathrm{Id} & & \vdots \\ & & & -A^{(q)}_{1,q}B^{(q),-\top} \\ 0 & \cdots & 0 & \mathrm{Id} \end{pmatrix} = \tag{13}$$

$$\left(\begin{pmatrix} & & A^{(q)}_{1,q}B^{(q),-\top} \\ \mathrm{Id} & & \vdots \\ & & A^{(q)}_{q-1,q}B^{(q),-\top} \\ 0 \quad \cdots & 0 & \mathrm{Id} \end{pmatrix} \cdots \begin{pmatrix} \mathrm{Id} & A^{(2)}_{1,2}B^{(2),-\top} & 0 \\ 0 & \mathrm{Id} & \vdots \\ 0 & \cdots & \mathrm{Id} \end{pmatrix} \right)^{-1} \tag{14}$$

$$= \begin{pmatrix} \mathrm{Id} & 0 & \cdots & \cdots & 0 \\ B^{(2),-1}A^{(2)}_{2,1} & \mathrm{Id} & \ddots & 0 & \vdots \\ \vdots & B^{(3),-1}A^{(3)}_{3,2} & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \mathrm{Id} & 0 \\ B^{(q),-1}A^{(q)}_{q,1} & B^{(q),-1}A^{(q)}_{q,2} & \cdots & B^{(q),-1}A^{(q)}_{q,q-1} & \mathrm{Id} \end{pmatrix}^{-\top}. \tag{15}$$

Here, we have used the formulae for the inverses and products of elementary lower-triangular matrices [236, pp.150–151],

$$\left(\mathrm{Id} + (0, \ldots, 0, a_{k+1}, \ldots, a_N)^\top \otimes \boldsymbol{e}_k\right)^{-1} = \mathrm{Id} - (0, \ldots, 0, a_{k+1}, \ldots, a_N)^\top \otimes \boldsymbol{e}_k, \tag{16}$$

$$\left(\mathrm{Id} + (0, \ldots, 0, a_{k+1}, \ldots, a_N)^\top \otimes \boldsymbol{e}_k\right) \left(\mathrm{Id} + (0, \ldots, 0, b_{l+1}, \ldots, b_N)^\top \otimes \boldsymbol{e}_l\right) \tag{17}$$

$$= \mathrm{Id} + (0, \ldots, 0, a_{k+1}, \ldots, a_N)^\top \otimes \boldsymbol{e}_k + (0, \ldots, 0, b_{l+1}, \ldots, b_N)^\top \otimes \boldsymbol{e}_l, \tag{18}$$

where $\boldsymbol{e}_k$ is the kth standard Euclidean basis row vector, with $k < l$. □

Proof of Lemma 3. We prove the result in the setting of Examples 1, since the proof for Example 2 is similar. The inequality $\|\phi\|_*^2 \geq \frac{1}{\|\mathcal{L}\|}\|\phi\|^2_{H^{-s}(\Omega)}$ and (4.32) imply that

$$\|\phi\|_*^2 \geq \frac{1}{\|\mathcal{L}\|} \sup_{v \in H^s_0(\Omega)} \sum_{i \in I^{(k)}} 2[\alpha_i \phi_i, v] - \|v\|^2_{H^s_0(\Omega)} \tag{19}$$

$$\geq \frac{1}{\|\mathcal{L}\|} \sum_{i \in I^{(k)}} \sup_{v \in H^s_0\left(B_{(\delta/2)h^k}(x_i)\right)} 2[\alpha_i \phi_i, v] - \|v\|^2_{H^s_0\left(B_{(\delta/2)h^k}(x_i)\right)} \tag{20}$$

$$= \frac{1}{\|\mathcal{L}\|} \sum_{i \in I^{(k)}} |\alpha_i|^2 \, \|\phi_i\|^2_{H^{-s}\left(B_{(\delta/2)h^k}(x_i)\right)} \tag{21}$$

$$\geq \frac{1}{\|\mathcal{L}\|} |\alpha|^2 \inf_i \|\phi_i\|^2_{H^{-s}\left(B_{(\delta/2)h^k}(x_i)\right)}. \tag{22}$$

The identity $\|\phi_i\|^2_{H^{-s}\left(B^2_{(\delta/2)h^k}(x_i)\right)} = h^{2sk}(\delta/2)^{2s-d}\|\boldsymbol{\delta}(\,\cdot\, - 0)\|^2_{H^{-s}(B_1(0))}$ concludes the proof with $C_\Phi := \|\mathcal{L}\|(\delta/2)^{d-2s}\,\|\boldsymbol{\delta}(\,\cdot\, - 0)\|^{-1}_{H^{-s}(B_1(0))}$ and $H := h^s$. □

Proof of Lemma 4 in the case of Example 2. Let ζ be a set of points such that the family $\{B_{\rho h^k}(z)\}_{z\in\zeta}$ covers Ω, and such that $\sup_{x\in\Omega}\#\{z\in\zeta : x\in B_{2\rho h^k}(z)\}\leq C(d)$. For $i\in J^{(l)}$ and $z\in\zeta$, we write $i\rightsquigarrow z$ if z is the element of ζ closest to i (using an arbitrary way to break ties). For $1\leq k<l\leq q$, $\phi\coloneqq\sum_{i\in J^{(l)}}\alpha_i\phi_i$, $\varphi\coloneqq\sum_{i\in J^{(l)}}\alpha_i\varphi_i$, and $\varphi_i\coloneqq\sum_{j\in I^{(k)}}w_{ij}\phi_j^{(k)}$, we have

$$\|\phi-\varphi\|^2_{H^{-s}(\Omega)}=\sup_{v\in H^s_0(\Omega)}\left(\sum_{z\in\zeta}\sum_{i\rightsquigarrow z}\int_{B_{2\rho h^k}(z)}2\alpha_i(\phi_i-\varphi_i)v(x)\,\mathrm{d}x\right)-\|v\|^2_{H^s_0(\Omega)}. \tag{23}$$

The Bramble–Hilbert lemma [66] and the vanishing moment property (4.36) of $\phi_i-\varphi_i$ yield that

$$\sum_{i\rightsquigarrow z}\int_{B_{2\rho h^k}(z)}2\alpha_i(\phi_i-\varphi_i)v(x)\,\mathrm{d}x \tag{24}$$

$$\leq 2\left(2\rho h^k\right)^s\left\|\sum_{i\rightsquigarrow z}\alpha_i\left(\phi_i-\varphi_i\right)\right\|_{L^2(B_{2\rho h^k}(z))}\|D^s v\|_{L^2(B_{2\rho h^k}(z))} \tag{25}$$

$$\leq 2C\left(2\rho h^k\right)^{2s}\left\|\sum_{i\rightsquigarrow z}\alpha_i\left(\phi_i-\varphi_i\right)\right\|^2_{L^2(B_{2\rho h^k}(z))}+\frac{\|D^s v\|^2_{L^2(B_{2\rho h^k}(z))}}{2C}. \tag{26}$$

Summing over all $z\in\zeta$ and choosing the constant C appropriately yields

$$\|\phi-\varphi\|^2_{H^{-s}(\Omega)}\leq C\rho^{2s}h^{2ks}\sum_{z\in\zeta}\left\|\sum_{i\rightsquigarrow z}\alpha_i\left(\phi_i-\varphi_i\right)\right\|^2_{L^2(B_{2\rho h^k}(z))}. \tag{27}$$

Since the ϕ_i are L^2-orthogonal to each other and $\|\phi_i\|^2_{L^2}\leq C$,

$$\sum_{z\in\zeta}\left\|\sum_{i\rightsquigarrow z}\alpha_i\left(\phi_i-\varphi_i\right)\right\|^2_{L^2(B_{2\rho h^k}(z))}\leq 2\sum_{z\in\zeta}\left(\left[\sum_{i\rightsquigarrow z}\alpha_i^2\|\phi_i\|^2_{L^2}\right]+\left\|\sum_{i\rightsquigarrow z}\alpha_i\varphi_i\right\|^2_{L^2(B_{2\rho h^k}(z))}\right) \tag{28}$$

$$\leq C\left(|\alpha|^2+\sum_{z\in\zeta}\left\|\sum_{i\rightsquigarrow z}\alpha_i\varphi_i\right\|^2_{L^2}\right). \tag{29}$$

Inserting the definition of the φ_i yields

$$\sum_{z\in\zeta}\left\|\sum_{i\rightsquigarrow z}\alpha_i\varphi_i\right\|_{L^2}^2 = \sum_{z\in\zeta}\left\|\sum_{j\in\tilde{I}^{(k)}}\sum_{i\rightsquigarrow z}\alpha_i w_{ij}\phi_j^{(k)}\right\|_{L^2}^2 \leq \sum_{z\in\zeta}\sum_{j\in\tilde{I}^{(k)}}\left(\sum_{i\rightsquigarrow z}\alpha_i w_{ij}\right)^2\left\|\phi_j^{(k)}\right\|_{L^2}^2 \tag{30}$$

$$\leq Ch^{-kd}\sum_{z\in\zeta}\sum_{j\in\tilde{I}^{(k)}}\left(\sum_{i\rightsquigarrow z}|\alpha_i||w_{ij}|\right)^2. \tag{31}$$

We will now use the fact that on $\mathbb{R}^n$, we have the norm inequalities $n^{-1/2}|\cdot|_1 \leq |\cdot|_2 \leq |\cdot|_1$. By a sphere-packing argument, for any $z \in \zeta$, we have $\left\{i \in J^{(l)} \,\middle|\, i \rightsquigarrow z\right\} \leq C(d)(\rho/\delta)^d h^{d(k-l)}$ Thus, the number of summands in the innermost sum is at most $C(d)(\rho/\delta)^d h^{(k-l)d}$ and using the above norm inequalites, we obtain

$$h^{-kd}\sum_{z\in\zeta}\sum_{j\in\tilde{I}^{(k)}}\left(\sum_{i\rightsquigarrow z}|\alpha_i||w_{ij}|\right)^2 \tag{32}$$

$$\leq C(\rho/\delta)^d h^{-ld}\sum_{z\in\zeta}\sum_{i\rightsquigarrow z}\sum_{j\in\tilde{I}^{(k)}}\left(|\alpha_i||w_{ij}|\right)^2 \leq C(\rho/\delta)^d h^{-ld}\omega_{l,k}^2|\alpha|^2. \tag{33}$$

Putting the above together yields the result. □

Proof of Lemma 10. Define

$$R := \mathrm{Id} - \frac{2}{\|A\| + \|A^{-1}\|^{-1}}A, \qquad\qquad r := \frac{1 - \frac{1}{\|A^{-1}\|\|A\|}}{1 + \frac{1}{\|A^{-1}\|\|A\|}},$$

and observe that $\|R\| = r$. Since $A = \frac{\|A\|+\|A^{-1}\|^{-1}}{2}(\mathrm{Id} - R)$, it follows from a Neumann series argument that $A^{-1} = \frac{2}{\|A\|+\|A^{-1}\|^{-1}}\sum_{k=0}^{\infty} R^k$. The positive definiteness of A implies that

$$|R_{i,j}| \leq \max\left\{1, \frac{2C}{\|A\| + \|A^{-1}\|^{-1}}\right\}\exp(-\gamma d(i,j)).$$

Let $C_R := \max\left\{1, \frac{2C}{\|A\|+\|A^{-1}\|^{-1}}\right\}$. Lemma 9 implies that

$$|R^k_{i,j}| \leq \left(c_d\,(\gamma/2)\right)^{k-1} C_R^k \exp\left(-\frac{\gamma}{2}d(i,j)\right).$$

Combining the above estimates yields

$$\frac{\|A\| + \|A^{-1}\|^{-1}}{2} \left|\left(A^{-1}\right)_{i,j}\right| \leq (n+1)\left(c_d\left(\gamma/2\right)\right)^{n-1} C_R^n \exp\left(-\frac{\gamma}{2}d(i,j)\right) + \frac{r^{n+1}}{1-r} \tag{34}$$

$$\leq \exp\left(\left(1+\log\left(c_d\left(\gamma/2\right)\right)+\log(C_R)\right)n - \frac{\gamma}{2}d(i,j)\right) \tag{35}$$

$$+ \exp\left(-\log(1-r)+\log(r)(n+1)\right). \tag{36}$$

By choosing

$$\nu := \frac{\frac{\gamma}{2}d(i,j) - \log(1-r)}{\left(1+\log\left(c_d\left(\gamma/2\right)\right)+\log(C_R)\right) - \log(r)}, \tag{37}$$

and $n+1 := \lceil \nu \rceil$, we obtain

$$\exp\left(\left(1+\log\left(c_d\left(\gamma/2\right)\right)+\log(C_R)\right)n - \frac{\gamma}{2}d(i,j)\right) + \exp\left(-\log(1-r)+\log(r)(n+1)\right) \tag{38}$$

$$\leq \exp\left(\left(1+\log\left(c_d\left(\gamma/2\right)\right)+\log(C_R)\right)\nu - \frac{\gamma}{2}d(i,j)\right) + \exp\left(-\log(1-r)+\log(r)\nu\right) \tag{39}$$

$$= 2\exp\left(\frac{-\log(1-r)\left(1+\log\left(c_d\left(\gamma/2\right)\right)+\log(C_R)\right)+\log(r)\frac{\gamma}{2}d(i,j)}{\left(1+\log\left(c_d\left(\gamma/2\right)\right)+\log(C_R)\right)-\log(r)}\right). \tag{40}$$

This yields the upper bound

$$\left|\left(A^{-1}\right)_{i,j}\right| \leq \frac{4\cdot\exp\left(\frac{-2\log(1-r)\left(1+\log\left(c_d(\gamma/2)\right)+\log(C_R)\right)+\log(r)\frac{\gamma}{2}d(i,j)}{\left(1+\log\left(c_d(\gamma/2)\right)+\log(C_R)\right)-\log(r)}\right)}{\|A\|+\|A^{-1}\|^{-1}} \tag{41}$$

$$= \frac{4}{\|A\|+\|A^{-1}\|^{-1}}\cdot\exp\left(\frac{\log(r)}{\left(1+\log\left(c_d\left(\gamma/2\right)\right)+\log(C_R)\right)-\log(r)}\frac{\gamma}{2}d(i,j)\right) \tag{42}$$

$$\cdot\exp\left(\frac{-2\log(1-r)\left(1+\log\left(c_d\left(\gamma/2\right)\right)+\log(C_R)\right)}{\left(1+\log\left(c_d\left(\gamma/2\right)\right)+\log(C_R)\right)-\log(r)}\right). \tag{43}$$

Optimising the term on line (43) over $\left(1+\log\left(c_d\left(\gamma/2\right)\right)+\log(C_R)\right)$ yields

$$\left|(A^{-1})_{i,j}\right| \leq \frac{4}{\left(\|A\|+\|A^{-1}\|^{-1}\right)(1-r)^2}\exp\left(\frac{\frac{\gamma}{2}d(i,j)\log(r)}{\left(1+\log\left(c_d\left(\gamma/2\right)\right)+\log(C_R)\right)-\log(r)}\right). \tag{44}$$

□

Proof of Lemma 11. In this proof, we will use the notation $k{:}l$ to denote the individual indices from k to l, as opposed to matrix blocks. We will establish the result by showing that, for all $1 \leq k \leq N$, the k^{th} column of L (when considered as an element of $\mathbb{R}^{I \times I}$ by zero padding) satisfies the exponential decay stated in the lemma. Let $S^{(k)} := B_{k:N,k:N} - B_{k:N,1:k-1}(B_{1:k-1,1:k-1})^{-1}B_{1:k-1,k:N}$. Then $L_{k:N,k} = S^{(k)}_{:,1}/\sqrt{S^{(k)}_{k,k}}$. Lemma 2 implies that $S^{(k)} = (B_{k:N,k:N})^{-1}$, and hence Lemma 10 yields that

$$\left|\left(S^{(k)}\right)_{i,j}\right| \leq \frac{4}{\left(\|B\| + \|B^{-1}\|^{-1}\right)(1-r)^2} \exp\left(\frac{\log(r)\frac{\gamma}{2}d(i,j)}{1 + \log\left(c_d\left(\gamma/2\right)\right) + \log(C_R) - \log(r)}\right). \tag{45}$$

Here we used the facts that the spectrum of $B_{k:n,k:n}$ is contained in $[\lambda_{\min}(B), \lambda_{\max}(B)]$ and that the right-hand side of the above estimate is increasing in r and C_R. The estimate $S^{(k)}_{k,k} \geq \frac{1}{\|S^{(k),-1}\|} \geq \frac{1}{\|B\|}$ completes the proof. □

Proof of Lemma 12. For any matrix T for which the Neumann series $\sum_{k=0}^{\infty} T^k$ converges in the operator norm, we have $(\mathrm{Id} - T)^{-1} = \sum_{k=0}^{\infty} T^k$. Therefore, $L^{-1} = \sum_{k=0}^{\infty}(\mathrm{Id} - L)^k$ if the right-hand side series is convergent. Since $\mathrm{Id} - L$ has the block-lower-triangular structure

$$(\mathrm{Id} - L) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ -L_{2,1} & \ddots & 0 & 0 \\ \vdots & \ddots & \ddots & \vdots \\ -L_{q,1} & \dots & -L_{q,q-1} & 0 \end{pmatrix}, \tag{46}$$

it follows that $\mathrm{Id}-L$ is q-nilpotent, i.e. $(\mathrm{Id}-L)^q = 0$ and the Neuman series terminates after the first q summands. Using this, we will now show that the exponential decay of L is preserved under inversion. To this end, consider the $(k,l)^{\text{th}}$ block of $(\mathrm{Id}-L)^p$ and observe that

$$\left|\left(((\mathrm{Id} - L)^p)_{k,l}\right)_{ij}\right| = \left|(-1)^p \sum_{k=s_1>s_2>\cdots>s_p>s_{p+1}=l} \left(\prod_{m=1}^{p} L_{s_m,s_{m+1}}\right)_{ij}\right| \tag{47}$$

$$\leq \sum_{k=s_1>s_2>\cdots>s_p>s_{p+1}=l} \left(c_d\left(\gamma/2\right)C\right)^p \exp\left(-\frac{\gamma}{2}d(i,j)\right) \tag{48}$$

$$= \binom{k-l-1}{p-1}\left(c_d\left(\gamma/2\right)C\right)^p \exp\left(-\frac{\gamma}{2}d(i,j)\right), \tag{49}$$

where the inequality follows from Lemma 9.

Summing (47) over p, we obtain, for $i \neq j$,

$$\left|\left(L_{k,l}^{-1}\right)_{ij}\right| \leq \sum_{p=1}^{k-l-1} \binom{k-l-1}{p-1} \left(c_d\left(\gamma/2\right)C\right)^p \exp\left(-\frac{\gamma}{2}d(i,j)\right) \tag{50}$$

$$\leq \left(1 + c_d\left(\gamma/2\right)C\right)^{k-l} \exp\left(-\frac{\gamma}{2}d(i,j)\right) \tag{51}$$

$$\leq 2^q \left(c_d\left(\gamma/2\right)C\right)^q \exp\left(-\frac{\gamma}{2}d(i,j)\right), \tag{52}$$

which concludes the proof of the lemma. □

With the above results on the propagation of exponential decay, we can now conclude the proof of Theorem 8.

Proof of Theorem 8. Applying Lemma 10, Condition1, and the condition number bound in Condition 2 yields the following estimate for $B^{(k),-1}$:

$$\left|\left(B^{(k),-1}\right)_{ij}\right| \leq \frac{4\exp\left(\frac{\log(r)}{(1+\log(c_d(\gamma/2))+\log(C_R)-\log(r))}\frac{\gamma}{2}d(i,j)\right)}{\left(\left\|B^{(k)}\right\| + \left\|B^{(k),-1}\right\|^{-1}\right)(1-r)^2}, \tag{53}$$

with $C_R = \max\left\{1, \frac{2C_\gamma\left\|B^{(k),-1}\right\|}{1+\kappa}\right\}$ and $r = \frac{1-\kappa^{-1}}{1+\kappa^{-1}}$. Lemma 2 and Condition 2 yield

$$\lambda_{\max}\left(B^{(k)}\right) \leq \lambda_{\max}\left(A^{(k)}\right) \leq C_\Phi H^{-2k}, \tag{54}$$

$$\lambda_{\min}\left(B^{(k)}\right) \geq \frac{1}{C_\Phi}H^{-2(k-1)}. \tag{55}$$

Using these estimates, we obtain

$$\left|\left(B^{(k),-1}\right)_{ij}\right| \leq \frac{2C_\Phi H^{2(k-1)}}{(1-r)^2}\exp\left(-\tilde{\gamma}d(i,j)\right), \tag{56}$$

where $\tilde{C}_R = \max\left\{1, \frac{2C_\gamma C_\Phi}{1+\kappa}\right\}$, $r = \frac{1-\kappa^{-1}}{1+\kappa^{-1}}$ and $\tilde{\gamma} := \frac{-\log(r)}{\left(1+\log(c_d(\gamma/2))+\log\left(\tilde{C}_R\right)-\log(r)\right)}\frac{\gamma}{2}$. Applying Lemma 9 to the products $B^{(i),-1}A_{ij}^{(i)}$ appearing in the definition of $\bar{L}^{-1}$ in Lemma 1, we obtain

$$\left|\left(\bar{L}^{-1}\right)_{ij}\right| \leq \frac{2C_\Phi C_\gamma\left(c_d\left(\tilde{\gamma}/2\right)\right)^2}{(1-r)^2}\exp\left(-\frac{\tilde{\gamma}}{2}d(i,j)\right). \tag{57}$$

Lemma 12 now yields the following decay bound for $\bar{L}$:

$$|\bar{L}_{ij}| \leq \left(4c_d\left(\tilde{\gamma}/4\right)\frac{C_\Phi C_\gamma\left(c_d\left(\tilde{\gamma}/2\right)\right)^2}{(1-r)^2}\right)^q \exp\left(-\frac{\tilde{\gamma}}{4}d(i,j)\right). \tag{58}$$

For a positive-definite matrix M, let chol (M) denote its lower-triangular Cholesky factor and set $L^{(k)} := \operatorname{chol}\left(B^{(k),-1}\right)$. Following the same procedure as in the bound of the decay of $B^{(k)}$ yields the decay bound

$$\left|L_{ij}^{(k)}\right| \leq \frac{2C_\Phi H^{(k-1)}}{(1-r)^2} \exp\left(-\tilde{\gamma} d(i,j)\right). \tag{59}$$

Applying Lemma 9 to the product $\bar{L}\operatorname{chol}(D) = \operatorname{chol}(\Theta)$ yields the decay bound

$$\left|(\operatorname{chol}(\Theta))_{ij}\right| \leq \frac{2C_\Phi c_d\left(\tilde{\gamma}/8\right)^2}{(1-r)^2}\left(4c_d\left(\tilde{\gamma}/4\right)\frac{C_\Phi C_\gamma\left(c_d\left(\tilde{\gamma}/2\right)\right)^2}{(1-r)^2}\right)^q \exp\left(-\frac{\tilde{\gamma}}{8}d(i,j)\right). \tag{60}$$

□

.2 Appendix to Chapter 5

.2.1 Correctness and computational complexity of the maximum-minimum distance ordering

Recall the variables used in Algorithm 11: the integer array P contains the minimax ordering; the real array $l[i]$ contains the distances of each point to the points that are already included in the minimax ordering; and the arrays of integer arrays c and p will contain the entries of the sparsity pattern in the sense that

$$(j \in c[i] \text{ and } i \in p[i]) \iff \texttt{dist}(i,j) \leq \rho l[i]. \tag{61}$$

We begin by showing correctness of the algorithm.

Theorem 26. *The ordering and sparsity pattern produced by Algorithm 11 coincide with those described in Section 6.3.1. Furthermore, whenever the while-loop in Line 22 is entered,*

(1) for all $i \in P$, $\ell[i]$ is as defined in Section 6.3.1;

(2) the array $c[P[1]]$ contains all $1 \leq j \leq N$ and for all other i in P, $c[i]$ contains exactly those $1 \leq j \leq N$ that satisfy $\texttt{dist}(i,j) \leq \rho\ell[i]$; and

(3) for all $1 \leq j \leq N$, $p[j]$ consists of $P[1]$ and all those $i \in P$ that satisfy $\texttt{dist}(i,j) \leq \rho\ell[i]$.

Proof. It is easy to see that if the for-loop in Line 27 were running over all $1 \leq j \leq N$, then the algorithm would yield the correct result. We claim that the restriction of the running variable to $\{j \in c[k] \mid \texttt{dist}(j,k) \leq \texttt{dist}(i,k) + \rho l[i]\}$ does not change the result of the algorithm. The proof will proceed by induction. Let us assume that Algorithm 11 has been correct up to a given time that Line 27 is visited. Then, by choice of k and the triangle inequality, any j that is omitted by the for-loop must satisfy $\texttt{dist}(i,j) > \rho l[i]$. Since i was chosen to have maximal minimal distance among the points remaining in H, and $\rho > 1$, this means that adding i to the maximin ordering can not decrease the maximal minimal distance of j. Thus, skipping the `decrease`! operation does not change the choice of P and l. Similarly, $\texttt{dist}(i,j) > \rho l[i]$ implies that the if-statement inside of the for-loop is false, meaning that skipping j does not change the update of c or p, from which the result follows. □

Having established Theorem 26, we will now use $\prec$, $\ell[i]$, and i_k to refer to the maximin ordering, the length-scale of the point with index i, and the k^{th} index in the maximin ordering. We will now bound the complexity of Algorithm 11 in the setting of Theorem 11.

Theorem 27. *In the setting of Theorem 11, there exists a constant depending C depending only on d, Ω, and δ, such that, for $\rho > C$, Algorithm 11 has computational complexity $C\rho^d N \log N$ in space and $CN\left(\rho^d \log^2 N + C_{\texttt{dist}_{\partial\Omega}}\right)$ in time, where $C_{\texttt{dist}_{\partial\Omega}}$ is the computational complexity of invoking the function $\texttt{dist}_{\partial\Omega}$.*

Proof. As a first step, we will upper-bound the number of iterations of the for-loop in Line 27 throughout the algorithm. To simplify the notation, C will denote a positive constant that depends on d, Ω and δ that may change throughout the proof. We claim that there exists $1 \leq k_{\min} \leq N$ depending only on d, Ω, and δ, such that, for all $i \succ i_{k_{\min}}$, by the time it appears in the while-loop at Line 22, there exists an index $k \prec i$ such that $l[k] \geq 2l[j]$ and $\texttt{dist}(i,k) \leq C\ell = C\ell[i]$. Indeed, since Ω has Lipschitz boundary, it satisfies an interior cone condition [5] in the sense that there exist $\theta \in (0, 2\pi]$ and $r > 0$ such that every point $x \in \Omega$ is the tip of a spherical cone within Ω with opening angle θ and radius r. This spherical cone contains a ball with radius r_γ, which depends only on θ and r. Let γ_i be such a cone with tip x_i. By a scaling argument, the spherical cone $\gamma_i \cap B_{\tilde{r}}(x_i)$ then contains a ball of radius $r_\gamma(\tilde{r}/r)$, for all $\tilde{r} < r$. For any $i \in I$ and any ball $B \subset \Omega$ with radius at least $4\ell[i]/\delta$, there exists a $k \prec i$ such that $l[k] \geq 2\ell[i]$ and $x_k \in B$. Thus, for $\ell[i] \leq \delta r_\gamma/4$, there exists a $k \prec i$ with $x_k \subset \gamma_i \cap B_{2\ell[i]}(x_i)$. By a sphere-packing argument, we can find a $k_{\min}$ such that, for all $i \succ i_{k_{\min}}$, $\ell[i] \leq \delta r_\gamma/4$, which yields the claim. Because of the above, for $\rho > C$, there exists a point satisfying the constraint in Line 25 with $\texttt{dist}(k,i) \leq 2C\ell[i]$. Thus, the number of times the for-loop in Line 27 is visited for a given index i is bounded above by $C_i := \#\{j \in I \mid \texttt{dist}(i,j) \leq 2(C+\rho)\ell[i]\}$. By a sphere-packing argument, $C_{i_m} \leq C(N/m)\rho^d$, for a constant C depending only on d, Ω, and δ. Summing the above over $1 \leq m \leq N$ yields the upper bound $C\rho^d N \log N$. The most costly step in the for-loop in Line 27 is the `decrease!` operation requiring the restoration of the heap property, which has computational complexity $O(\log N)$. Thus, the overall computational complexity is at most $CN\left(\rho^d \log^2 N + C_{\texttt{dist}_{\partial\Omega}}\right)$. The bound on the space complexity follows, since each iteration of the for-loop consumes $O(1)$ memory. □

Proof of Theorem19. Theorem 19 follows from Theorems 26 and 27. □

Algorithm 11 uses only pairwise distances between points, and thus automatically adapts to low-dimensional structure in the $\{x_i\}_{i\in I}$. Indeed, for $\Omega = \mathbb{R}^d$, the computational complexity of Algorithm 11 depends only on the intrinsic dimension of the dataset.

Condition 3 (Intrinsic dimension)**.** *There exist constants $C_{\tilde{d}}, \tilde{d} > 0$, independent of N, such that, for all $r, R > 0$ and $x \in \mathbb{R}^d$,*

$$\max\left\{|A| \,\middle|\, A \subset I, i,j \in A \implies \operatorname{dist}(x_i,x), \operatorname{dist}(x_j,x) \leq R, \operatorname{dist}(x_i,x_j) \geq r\right\} \leq C_{\tilde{d}}\left(\frac{R}{r}\right)^{\tilde{d}}.$$

We say that the point set $\{x_i\}_{i\in I}$ has intrinsic dimension *$\tilde{d}$.*

Condition 4 (Polynomial Scaling)**.** *There exists a polynomial $\boldsymbol{p}$ for which*

$$\frac{\max_{i\neq j\in I}\operatorname{dist}(x_i,x_j)}{\min_{i\neq j\in I}\operatorname{dist}(x_i,x_j)} \leq \boldsymbol{p}(N).$$

Theorem 28. *Let $\Omega = \mathbb{R}^d$ and $\rho \geq 2$. Then the computational complexity of Algorithm 11 is at most $C\rho^{\tilde{d}} N \log N$ in space and $CN\left(\log(N)\rho^{\tilde{d}}(\log N + C_{\texttt{dist}}) + C_{\texttt{dist}_{\partial\Omega}}\right)$ in time, for a constant $C = C\left(C_{\tilde{d}}, \tilde{d}, \boldsymbol{p}\right)$ depending only on the constants in Conditions 3 and 4.*

Proof. The proof is analogous to that of Theorem 27. The main difference is that the claim on the existence of $k_{\min}$ is replaced by the fact — which follows directly from the definition of the maximin ordering — that, for all i, there exists a $k \prec i$ such that $l[k] \geq 2\ell[i]$ and $\texttt{dist}(k,i) \leq 2\ell[i]$. In particular, any $\rho \geq 2$ leads to near-linear computational complexity. □

.3 Appendix to Chapter 6

.3.1 Computation of the KL-minimizer

Computation for the aggregated sparsity pattern

We first introduce some additional notation, defined in terms of an r-maximin ordering $\prec$ (see Section .3.2) and aggregated sparsity set $S = \tilde{S}_{\prec,\ell,\rho,\lambda}$, which we assume to be fixed. As before, I is the index set keeping track over the degrees of freedom, and $\tilde{I}$ is the index set indexing the supernodes. For a matrix A and sets of indices $\tilde{i}$ and $\tilde{j}$, we denote as the $A_{\tilde{i},\tilde{j}}$ the submatrix obtained by restricting the indices of A to $\tilde{i}$ and $\tilde{j}$, and as $A_{\tilde{i},:}$ ($A_{:,\tilde{j}}$) the matrix obtained by only restricting the row (column) indices. We adopt the convention of indexing having precedence over inversion, i.e. $A^{-1}_{\tilde{i},\tilde{j}} = (A_{\tilde{i},\tilde{j}})^{-1}$. For a supernode $\tilde{k} \in \tilde{I}$ and a degree of freedom $j \in I$, we write $j \in \tilde{k}$ if there exists a $k \rightsquigarrow \tilde{k}$ such that $k \preceq j$ and $(k, j) \in S$, and we accordingly form submatrices $A_{\tilde{i},\tilde{j}} \coloneqq (A_{ij})_{i \in \tilde{i}, j \in \tilde{j}}$. Note that by definition of the supernodes, we have $s_k \subset \tilde{k}$ for all $k \rightsquigarrow \tilde{k}$. Since we assume the sparsity pattern S to contain the diagonal, we furthermore have $k \rightsquigarrow \tilde{k} \Rightarrow k \in \tilde{k}$.

We first show how to efficiently compute the inverse Cholesky factor for the aggregated sparsity pattern (as has been observed before by [83], and [104]). For $\tilde{k} \in \tilde{I}$, we define $U^{\tilde{k}}$ as the unique upper triangular matrix such that $\Theta_{\tilde{k},\tilde{k}} = U^{\tilde{k}} U^{\tilde{k},\top}$. $U^{\tilde{k}}$ can be computed in complexity $O((\#\tilde{k})^3)$ in time and $O((\#\tilde{k})^2)$ in space by computing the Cholesky factorization of $\Theta_{\tilde{k},\tilde{k}}$ after reverting the ordering of its rows and columns, and then reverting the order of the rows and columns of the resulting Cholesky factor. The upper triangular structure of $U^{\tilde{k}}$ implies the following properties

$$\Theta_{s_k,s_k} = U^{\tilde{k}}_{s_k,s_k} U^{\tilde{k},\top}_{s_k,s_k}, \qquad U^{\tilde{k},-1}_{s_k,s_k}\mathbf{1} = \frac{1}{U^{\tilde{k}}_{kk}}\mathbf{e}_1, \tag{62}$$

$$U^{\tilde{k},-\top}_{s_k,s_k}\mathbf{1} = \left(U^{\tilde{k},-\top}\mathbf{e}_k\right)_{s_k,s_k}, \qquad U^{\tilde{k},-1}_{s_k,s_k} v_{s_k} = \left(U^{\tilde{k},-1} v\right)_{s_k}, \tag{63}$$

where $v \in \mathbb{R}^{\tilde{k}}$ is chosen arbitrarily. For any $k \rightsquigarrow \tilde{k}$, the first three properties above imply

$$L^{\rho}_{:,k} = \frac{\Theta^{-1}_{s_k}\mathbf{e}_1}{\sqrt{\mathbf{e}_1^\top \Theta^{-1}_{s_k}\mathbf{e}_1}} = U^{\tilde{k},-\top}_{s_k,s_k}\mathbf{e}_1 = U^{\tilde{k},-\top}\mathbf{e}_k. \tag{64}$$

Thus, computing the columns $L_{:,k}$ for all $k \rightsquigarrow \tilde{k}$ has computational complexity $O((\#\tilde{k})^3)$ in time and $O((\#\tilde{k})^2)$ in space. Algorithm 13 implements the formulae derived above.

GP regression in $\mathcal{O}(N + \rho^{2d})$ space complexity

As mentioned in Section 6.4.3, for many important operations arising in GP regression, the inverse-Cholesky factors L of the training covariance matrix need never be formed in full. Instead, matrix-vector multiplies with L or $L^{\top}$, as well as the computation of the log-determinant of L can be performed by computing the columns of L in an arbitrary order, using them to update the result, and deleting them again. For the example of computing the posterior mean μ and covariance C, this is done in Algorithm 19 (without aggregation) and 20 (with aggregation). In Section .3.4, we show how to compute the reverse maximin ordering and aggregated sparsity pattern in space complexity $\mathcal{O}(N + \rho^{d})$, thus allowing the entire algorithm to be run in space complexity $\mathcal{O}(N + \rho^{d})$ when using the aggregated sparsity pattern.

.3.2 Postponed proofs

Our theoretical results apply to more general orderings, called reverse r-maximin orderings, which for $r \in (0, 1]$ have the following property.

Definition 20. *An elimination ordering $\prec$ is called reverse r-maximin with length scales $\{\ell_i\}_{i \in I}$ if for every $j \in I$ we have*

$$\ell_j \coloneqq \min_{i \succ j} \operatorname{dist}(x_j, \{x_i\} \cup \partial\Omega) \geq r \max_{j \succ k} \min_{i \succ j} \operatorname{dist}(x_k, \{x_i\} \cup \partial\Omega). \qquad (65)$$

We note that the reverse maximin ordering from Section 6.3.1 is a reverse 1-maximin ordering; reverse r-maximin orderings with $r < 1$ can be computed in computational complexity $\mathcal{O}(N \log(N))$ (see Section .3.4). We define the sparsity patterns $S_{\prec,\ell,\rho}$ and $\tilde{S}_{\prec,\ell,\rho,\lambda}$ analogously to the case of the reverse maximin ordering, and we will write L^{ρ} for the incomplete Cholesky factors of Θ^{-1} computed using (6.3) based on the sparsity pattern $S_{\prec,\ell,\rho}$ or $\tilde{S}_{\prec,\ell,\rho,\lambda}$.

Computational complexity

Our estimates only depend on the *intrinsic dimension of the dataset* which is defined by counting the number of balls of radius r that can be fit into balls of radius R, for different $r, R > 0$.

Algorithm 19 Without aggregation

Input: $\mathcal{G}$, $\{x_i\}_{i\in I}$, $\prec$, $S_{\prec,\ell,\rho}$
Output: Cond. mean μ and cov. C

```
1:  for $k \in I_{\mathrm{Pr}}$ do
2:      $\mu_k \leftarrow 0$
3:  end for
4:  for $i \in I_{\mathrm{Tr}}, j \in I_{\mathrm{Pr}}$ do
5:      $(\Theta_{\mathrm{Tr,Pr}})_{ij} \leftarrow \mathcal{G}(x_i, x_j)$
6:  end for
7:  for $i \in I_{\mathrm{Pr}}, j \in I_{\mathrm{Pr}}$ do
8:      $(\Theta_{\mathrm{Pr,Pr}})_{ij} \leftarrow \mathcal{G}(x_i, x_j)$
9:  end for
10: for $k \in I_{\mathrm{Tr}}$ do
11:     for $i, j \in s_k$ do
12:         $\left(\Theta_{s_k,s_k}\right)_{ij} \leftarrow \mathcal{G}(x_i, x_j)$
13:     end for
14:     $v \leftarrow \Theta^{-1}_{s_k,s_k}\mathbf{e}_k$
15:     $v \leftarrow v/v_k$
16:     $\mu_{k,:} \leftarrow \mu_{k,:} + v_k\Theta_{k,\mathrm{Pr}}$
17:     $B_{k,:} \leftarrow v^{\top}\Theta_{\mathrm{Tr,Pr}}$
18: end for
19: $C \leftarrow \Theta_{\mathrm{Pr,Pr}} - B^{\top}B$
20: return $\mu, C$
```

Algorithm 20 With aggregation

Input:$\mathcal{G}$, $\{x_i\}_{i\in I}$, $\prec$, $S_{\prec,\ell,\rho,\lambda}$
Output: Cond. mean μ and cov. C

```
1:  for $k \in I_{\mathrm{Pr}}$ do
2:      $\mu_k \leftarrow 0$
3:  end for
4:  for $i \in I_{\mathrm{Tr}}, j \in I_{\mathrm{Pr}}$ do
5:      $(\Theta_{\mathrm{Tr,Pr}})_{ij} \leftarrow \mathcal{G}(x_i, x_j)$
6:  end for
7:  for $i \in I_{\mathrm{Pr}}, j \in I_{\mathrm{Pr}}$ do
8:      $(\Theta_{\mathrm{Pr,Pr}})_{ij} \leftarrow \mathcal{G}(x_i, x_j)$
9:  end for
10: for $\tilde{k} \in \tilde{I}$ do
11:     for $i, j \in s_{\tilde{k}}$ do
12:         $\left(\Theta_{s_{\tilde{k}},s_{\tilde{k}}}\right)_{ij} \leftarrow \mathcal{G}(x_i, x_j)$
13:     end for
14:     $U \leftarrow P^{\updownarrow}\,\mathrm{chol}(P^{\updownarrow}K_{s_{\tilde{k}},s_{\tilde{k}}}P^{\updownarrow})P^{\updownarrow}$
15:     for $k \rightsquigarrow \tilde{k}$ do
16:         $v \leftarrow U^{-\top}\mathbf{e}_k$
17:         $\mu_{k,:} \leftarrow \mu_{k,:} + v_k\Theta_{k,\mathrm{Pr}}$
18:         $B_{k,:} \leftarrow v^{\top}\Theta_{\mathrm{Tr,Pr}}$
19:     end for
20: end for$C \leftarrow \Theta_{\mathrm{Pr,Pr}} - B^{\top}B$
21: return $\mu, C$
```

Figure .10: **Linear memory complexity.** Prediction and uncertainty quantification using KL-minimization with and without aggregation in $O(N + \rho^{2\tilde{d}})$ memory complexity.

Condition 5 (Intrinsic dimension)**.** *We say that $\{x_i\}_{i\in I} \subset \mathbb{R}^d$ has intrinsic dimension $\tilde{d}$ if there exists a constant $C_{\tilde{d}}$, independent of N, such that for all $r, R > 0$, $x \in \mathbb{R}^d$, we have*

$$\max\left\{|A| : i, j \in A \Rightarrow \mathrm{dist}(x_i, x), \mathrm{dist}(x_j, x) \le R, \mathrm{dist}(x_i, x_j) \ge r\right\} \le C_{\tilde{d}}\,(R/r)^{\tilde{d}}\,. \tag{66}$$

Remark 3. *Note that we always have $\tilde{d} \le d$.*

We also make a mild technical assumption requiring that most of the points belong to the finer scales of the ordering:

Condition 6 (Regular refinement)**.** *We say that $\{x_i\}_{i\in I} \subset \mathbb{R}^d$ fulfills the regular refinement condition for λ and ℓ with constant $C_{\lambda,\ell}$, if*

$$\sum_{k=\lfloor \log(\ell_1)/\log(\lambda) \rfloor}^{\infty} \#\{i : \lambda^k \leq \ell_i\} \leq C_{\lambda,\ell} N.$$

This condition excludes pathological cases like $x_i = 2^{-i}$ for which each scale contains the same number of points.

We obtain the following computational complexity:

Theorem 29. *Under Condition 5 with $C_{\tilde{d}}$ and $\tilde{d}$, using an r-reverse maximin ordering $\prec$ and $S_{\prec,\ell,\rho}$, Algorithm 12 computes L^{ρ} in complexity $CN\rho^{\tilde{d}}$ in space and $CN\rho^{3\tilde{d}}$ in time. If we assume in addition that $\{x_i\}_{i\in I}$ fulfills Condition 6 for λ and l with constant $C_{\lambda,\ell}$, then, using $\tilde{S}_{\prec,\ell,\rho,\lambda}$ or $\bar{S}_{\prec,\ell,\rho,\lambda}$, Algorithm 13 computes L^{ρ} in complexity $CN\rho^{\tilde{d}}$ in space and $C_{\lambda,\ell}CN\rho^{2\tilde{d}}$ in time. Here, the constant C depends only on $C_{\tilde{d}}$, $\tilde{d}$, r, λ, and the maximal cost of evaluating a single entry of Θ, but not on N or d.*

Proof. We begin by showing that the number of nonzero entries of an arbitrary column of $S_{\prec,\ell,\rho}$ is bounded above as $C\rho^{\tilde{d}}$. Considering the i-th column, the reverse r-maximin ordering ensures that for all $j, k \succ i$, we have $\operatorname{dist}(x_j, x_i) \geq r\ell_i$. Since for all $(i,j) \in S_{\prec,\ell,\rho}$ we have $i \prec j$ and $\operatorname{dist}(x_i, x_j) \leq \rho\ell_i$, Condition 5 implies that $\#\left\{j : (i,j) \in S_{\prec,\ell,\rho}\right\} \leq C_{\tilde{d}} \left(\frac{\rho\ell_i}{r\ell_i}\right)^{\tilde{d}}$. Computing the i-th column of L^{ρ} requires the inversion of the Matrix Θ_{s_i,s_i} which can be done in computational complexity $C\rho^{3\tilde{d}}$, leaving us with a total time complexity of $CN\rho^{\tilde{d}}$. We now want to bound the computational complexity when using the aggregated sparsity patterns $\tilde{S}_{\prec,\ell,\rho,\lambda}$ or $\bar{S}_{\prec,\ell,\rho,\lambda}$. As before, we write $j \in s$ if j is a child of the supernode s, that is if there exists a $i \rightsquigarrow s$ such that (i,j) is contained in $\tilde{S}_{\prec,\ell,\rho,\lambda}$ or $\bar{S}_{\prec,\ell,\rho,\lambda}$. We write $\#s$ to denote the number of children of s. By the same argument as above, the number of *children* in each supernode s is bounded by $C\rho^{\tilde{d}}$. We now want to show that the sum of the numbers of children of all supernodes is bounded as CN. For a supernode s, we write $\sqrt{s} \in I$ to denote the index that was first added to the supernode (see the construction described in Section 6.3.2). We now observe that for two distinct supernodes s and t with $c \leq \ell_{\sqrt{s}}, \ell_{\sqrt{t}} \leq c\lambda$, we have $\operatorname{dist}(x_{\sqrt{s}}, x_{\sqrt{t}}) \geq c\rho$, since otherwise we would have either $\sqrt{s} \rightsquigarrow t$ or$\sqrt{t} \rightsquigarrow s$. Thus, for every index $i \in I$ and $k \in \mathbb{Z}$, there exist at most C supernodes s with $i \in s$, $\lambda^k \leq \ell_{\sqrt{s}} < \lambda^{k+1}$. By using

Condition 6, we thus obtain

$$\begin{aligned}\sum_{s \in \tilde{I}} \# s = \sum_{i \in I} \# \left\{ s \in \tilde{I} : i \in s \right\} = \sum_{k \in \mathbb{Z}} \sum_{i \in I} \# \left\{ s \in \tilde{s} : i \in s, \lambda^k \leq \ell_{\sqrt{s}} < \lambda^{k+1} \right\} \\ \leq \sum_{k \in \mathbb{Z}} \sum_{i \in I : \ell_i \geq \lambda^k} C \leq NC.\end{aligned}$$

We now know that there are at most CN child-parent relationships between indices and supernodes and that each supernode can have at most $C\rho^{\tilde{d}}$ children. The worst case is thus that we have $CN/\rho^{\tilde{d}}$ supernodes, each having $C\rho^{\tilde{d}}$ children. This leads to the bounds on time and space complexity of the algorithm. □

Approximation accuracy

Our goal is to prove the following theorem:

Theorem 30. *Using an r-maximin ordering $\prec$ and sparsity patterns $S_{\prec,\ell,\rho}$ or $\tilde{S}_{\prec,\ell,\rho,\lambda}$, there exists a constant C depending only on d, Ω, r, λ, s, $\|\mathcal{L}\|$, $\|\mathcal{L}^{-1}\|$, and δ, such that for $\rho \geq C \log(N/\epsilon)$, we have*

$$\mathbb{D}_{\mathrm{KL}}\left(\mathcal{N}(0, \Theta) \,\|\, \mathcal{N}(0, \left(L^{\rho} L^{\rho,\top}\right)^{-1})\right) + \left\| \Theta - (L^{\rho} L^{\rho,\top})^{-1} \right\|_{\mathrm{Fro}} \leq \epsilon. \tag{67}$$

Thus, Algorithm 12 computes an ϵ-accurate approximation of Θ in computational complexity $CN \log^{d}(N/\epsilon)$ in space and $CN \log^{3d}(N/\epsilon)$ in time, from $CN \log^{d}(N/\epsilon)$ entries of Θ. Similarly, Algorithm 13 computes an ϵ-accurate approximation of Θ in computational complexity $CN \log^{d}(N/\epsilon)$ in space and $CN \log^{2d}(N/\epsilon)$ in time, from $CN \log^{d}(N/\epsilon)$ entries of Θ.

Theorem 10 implies that, under the assumptions of Theorem 22, the Cholesky factor of $A = \Theta^{-1}$ decays exponentially away from the diagonal.

Theorem 31. *In the setting of Theorem 22, there exists a constant C depending only on δ, r, d, Ω, s, $\|\mathcal{L}\|$, and $\|\mathcal{L}^{-1}\|$, such that for $\rho \geq C \log(N/\epsilon)$,*

$$S \supset \{(i,j) \in I \times I : \operatorname{dist}(x_i, x_j) \leq \rho \min(\ell_i, \ell_j)\} \tag{68}$$

and

$$L^{S}_{ij} := \begin{cases} \left(\operatorname{chol}(A)\right)_{ij}, & (i,j) \in S, \\ 0, & \textit{otherwise}, \end{cases} \tag{69}$$

we have $\left\| A - L^{S} L^{S,\top} \right\|_{\mathrm{Fro}} \leq \epsilon$.

In order to prove the approximation accuracy of the KL-minimizer, we have to compare the approximation accuracy in Frobenius norm and in KL-divergence. For brevity, we write $\mathbb{D}_{\mathrm{KL}}(A \parallel B) := \mathbb{D}_{\mathrm{KL}}(\mathcal{N}(0, A) \parallel \mathcal{N}(0, B))$.

Lemma 18. *Let $\lambda_{\min}$, $\lambda_{\max}$ be the minimal and maximal eigenvalues of Θ, respectively. Then there exists a universal constant C such that for any matrix $M \in \mathbb{R}^{I \times I}$, we have*

$$\lambda_{\max} \left\| A - MM^{\top} \right\|_{\mathrm{Fro}} \leq C \Rightarrow \mathbb{D}_{\mathrm{KL}}\left(\Theta \,\middle\|\, \left(MM^{\top}\right)^{-1}\right) \leq \lambda_{\max} \left\| A - MM^{\top} \right\|_{\mathrm{Fro}},$$
$$\mathbb{D}_{\mathrm{KL}}\left(\Theta \,\middle\|\, \left(MM^{\top}\right)^{-1}\right) \leq C \Rightarrow \left\| A - MM^{\top} \right\|_{\mathrm{Fro}} \leq \lambda_{\min}^{-1} \mathbb{D}_{\mathrm{KL}}\left(\Theta \,\middle\|\, \left(MM^{\top}\right)^{-1}\right).$$

Proof. Writing $L := \mathrm{chol}(A)$ and $\phi_{\mathrm{Fro}}(x) := x^2$ and $\phi_{\mathrm{KL}}(x) := (x - \log(1 + x))/2$, we have

$$\begin{aligned}
\lambda_{\min} \left\| A - MM^{\top} \right\|_{\mathrm{Fro}} &= \lambda_{\min} \left\| LL^{-1} \left(A - MM^{\top}\right) L^{-\top} L^{\top} \right\|_{\mathrm{Fro}} \\
&\leq \left\| \mathrm{Id} - L^{-1} MM^{\top} L^{-\top} \right\|_{\mathrm{Fro}} = \sum_{k=1}^{N} \phi_{\mathrm{Fro}} \left(\lambda_k \left(L^{-1} MM^{\top} L^{-\top} \right) - 1 \right) \\
&= \left\| L^{-1} \left(A - MM^{\top}\right) L^{-\top} \right\|_{\mathrm{Fro}} \leq \lambda_{\max} \left\| A - MM^{\top} \right\|_{\mathrm{Fro}}
\end{aligned}$$

and

$$\mathbb{D}_{\mathrm{KL}}\left(\Theta \,\middle\|\, \left(MM^{\top}\right)^{-1}\right) = \sum_{k=1}^{N} \phi_{\mathrm{KL}} \left(\lambda_k \left(L^{-1} MM^{\top} L^{-\top} \right) \right), \tag{70}$$

where $(\lambda_k(\cdot))_{1 \leq k \leq N}$ returns the eigenvalues ordered from largest to smallest, while $\lambda_{\min}(\cdot)$ ($\lambda_{\max}(\cdot)$) returns the smallest (largest) eigenvalue. The leading-order Taylor expansion of ϕ_{KL} around 0 is given by $x \mapsto x^2/4$. Thus, there exists a constant C such that for $\min(|x|, \phi_{\mathrm{Fro}}(x), \phi_{\mathrm{KL}}(x)) \leq C$ we have $\phi_{\mathrm{KL}}(x) \leq \phi_{\mathrm{Fro}}(x) \leq 8\phi_{\mathrm{KL}}(x)$. Therefore, for $\lambda_{\max} \|A - MM^{\top}\|_{\mathrm{Fro}} \leq C$, we have $\mathbb{D}_{\mathrm{KL}}\left(\Theta \,\middle\|\, (MM^{\top})^{-1}\right) \leq \lambda_{\max} \|A - MM^{\top}\|_{\mathrm{Fro}}$. For $\mathbb{D}_{\mathrm{KL}}\left(\Theta \,\middle\|\, (MM^{\top})^{-1}\right) \leq C$, this implies $\|A - MM^{\top}\|_{\mathrm{Fro}} \leq \lambda_{\min}^{-1} \mathbb{D}_{\mathrm{KL}}\left(\Theta \,\middle\|\, (MM^{\top})^{-1}\right)$. □

Using Lemma 18, we can conclude Theorem 22.

Proof of Theorem 30. Theorem 7 implies that there exists a polynomial $\mathbf{p}$ depending only on $(d, s, \delta, \mathcal{L})$ such that $\lambda_{\max}, \lambda_{\min}^{-1} \leq \mathbf{p}(N)$. Thus, by choosing $\rho \geq C \log(N)$ we can deduce by Theorem 31 that $\lambda_{\max} \left\| A - L^S L^{S,\top} \right\| \leq C$ where C is the constant in Lemma 18. Thus, we have $\mathbb{D}_{\mathrm{KL}}\left(\Theta \,\middle\|\, \left(L^S L^{S,\top}\right)^{-1}\right) \leq \lambda_{\max} \left\| A - L^S L^{S,\top} \right\|$. The KL-optimality of L^ρ implies $\mathbb{D}_{\mathrm{KL}}\left(\Theta \,\middle\|\, \left(L^\rho L^{\rho,\top}\right)^{-1}\right) \leq \lambda_{\max} \left\| A - L^S L^{S,\top} \right\| \leq C$. Using one more time Lemma 18, we also obtain

$$\left\| A - L^\rho L^{\rho,\top} \right\| \leq \lambda_{\min}^{-1} \mathbb{D}_{\mathrm{KL}}\left(\Theta \,\middle\|\, \left(L^\rho L^{\rho,\top}\right)^{-1}\right) \leq \lambda_{\max}/\lambda_{\min} \left\| A - L^S L^{S,\top} \right\|. \tag{71}$$

□

.3.3 Including the prediction points

Ordering the prediction points first

Algorithm 21 describes how to compute the inverse Cholesky factor when forcing the prediction points to be ordered before the training points. In order to compute the ordering of the prediction points after the ordering of the training points has been fixed, we need to compute the distance of each prediction point to the closest training point. When using Algorithm 24, this can be done efficiently while computing the maximin ordering of the training points by including the prediction points into the initial list of children of i, as is done in Line 11 of the algorithm. Once the joint inverse Cholesky factor $L = \begin{pmatrix} L_{\mathrm{Pr},\mathrm{Pr}} & 0 \\ L_{\mathrm{Tr},\mathrm{Pr}} & L_{\mathrm{Tr},\mathrm{Tr}} \end{pmatrix}$ has been computed, we have $\mathbb{E}\left[X_{\mathrm{Pr}} | X_{\mathrm{Tr}} = y\right] = L_{\mathrm{Pr},\mathrm{Pr}}^{-\top} L_{\mathrm{Tr},\mathrm{Pr}}^{\top} y$ and $\mathbb{C}\mathrm{ov}\left[X_{\mathrm{Pr}} | X_{\mathrm{Tr}}\right] = L_{\mathrm{Pr},\mathrm{Pr}}^{-\top} L_{\mathrm{Pr},\mathrm{Pr}}^{-1}$. We note that the conditional expectation can be computed by forming the columns of L one by one, without ever having to hold the entire matrix in memory, thus leading to linear space complexity similar to Section 6.4.3, while the same is not possible for the conditional covariance matrix.

Ordering the prediction points last, for accurate extrapolation

Splitting the prediction set $I_{\mathrm{Pr}} = \bigcup_{1 \leq b \leq m_{\mathrm{Pr}}} J_b$ into m_{Pr} batches of n_{Pr} predictions, we want to compute the conditional mean vector and covariance matrix of the variables in each batch separately, by using the inverse Cholesky factor $\bar{L}^{\rho}$ of the joint covariance matrix obtained from KL-minimization subject to the sparsity constraint given by $\bar{S} = S_{<,l,\rho,\lambda} \cup \{(i,j) : j \in J_b\}$. Naively, this requires us to recompute the inverse Cholesky factor L for every batch, leading to a computational complexity of $O\left(m_{\mathrm{Pr}}(N + n_{\mathrm{Pr}})(\rho^{\tilde{d}} + n_{\mathrm{Pr}})^2\right)$. However, by reusing a part of the computational complexity across different batches, Algorithm 23 is to instead achieve computational complexity of $O\left((N + n_{\mathrm{Pr}})(\rho^{2\tilde{d}} + m_{\mathrm{Pr}}\left(\rho^{\tilde{d}} + n_{\mathrm{Pr}}^2\right)\right)$. In the following, we derive the formulae used by this algorithm to compute the conditional mean and covariance. For a fixed batch J_b, define $\bar{\Theta}$ as the approximate joint covariance matrix implied by the inverse Cholesky factor $\bar{L}^{\rho}$. It has the block-structure

$$\begin{pmatrix} \bar{\Theta}_{\mathrm{Tr},\mathrm{Tr}} & \bar{\Theta}_{\mathrm{Tr},b} \\ \bar{\Theta}_{b,\mathrm{Tr}} & \bar{\Theta}_{b,b} \end{pmatrix} =: \begin{pmatrix} \bar{A}_{\mathrm{Tr},\mathrm{Tr}} & \bar{A}_{\mathrm{Tr},b} \\ \bar{A}_{b,\mathrm{Tr}} & \bar{A}_{b,b} \end{pmatrix}^{-1} = \begin{pmatrix} \bar{L}_{\mathrm{Tr},\mathrm{Tr}}^{\top} & \bar{L}_{b,\mathrm{Tr}}^{\top} \\ 0 & \bar{L}_{b,b}^{\top} \end{pmatrix}^{-1} \begin{pmatrix} \bar{L}_{\mathrm{Tr},\mathrm{Tr}} & 0 \\ \bar{L}_{b,\mathrm{Tr}} & \bar{L}_{b,b} \end{pmatrix}^{-1} =: \bar{L}^{\rho,-\top} \bar{L}^{\rho,-1}, \tag{72}$$

where $\bar{L}^{\rho}$ is the inverse-Cholesky factor obtained by applying KL-minimization to the joint covariance matrix subject to the sparsity constraint given by $\bar{S}$. We can then write the posterior mean and covariance of a GP $X \sim \mathcal{N}(0, \bar{\Theta})$ as

$$\mathbb{E}\left[X_b | X_{\mathrm{Tr}} = y\right] = \bar{\Theta}_{b,\mathrm{Tr}} \bar{\Theta}_{\mathrm{Tr},\mathrm{Tr}}^{-1} y = -\bar{A}_{b,b}^{-1} \bar{A}_{b,\mathrm{Tr}} y = -\left(\bar{L}_{b,\mathrm{Tr}} \bar{L}_{b,\mathrm{Tr}}^{\top} + \bar{L}_{b,b} \bar{L}_{b,b}^{\top}\right)^{-1} \bar{L}_{b,\mathrm{Tr}} \bar{L}_{\mathrm{Tr},\mathrm{Tr}}^{\top} y \tag{73}$$

$$\mathbb{C}\mathrm{ov}\left[X_b | X_{\mathrm{Tr}}\right] = \bar{\Theta}_{b,b} - \bar{\Theta}_{b,\mathrm{Tr}} \bar{\Theta}_{\mathrm{Tr},\mathrm{Tr}}^{-1} \bar{\Theta}_{\mathrm{Tr},b} = \bar{A}_{b,b}^{-1} = \left(\bar{L}_{b,\mathrm{Tr}} \bar{L}_{b,\mathrm{Tr}}^{\top} + \bar{L}_{b,b} \bar{L}_{b,b}^{\top}\right)^{-1}. \tag{74}$$

Algorithm 21 Prediction variables first

Input: $\mathcal{G}$, $\{x_i\}_{i \in I_{\mathrm{Tr}}}$, Ω, $\{x_i\}_{i \in I_{\mathrm{Tr}}}$, ρ, (λ)
Output: $L \in \mathbb{R}^{N \times N}$ l. triang. in $\prec$

```
1: Comp. $\prec_{\mathrm{Pr}}$, $l_{\mathrm{Pr}}$ from $\{x_i\}_{i \in I_{\mathrm{Pr}}}$, $\tilde{\Omega}$
2: Comp. $\prec_{\mathrm{Tr}}$, $l_{\mathrm{Tr}}$ from $\{x_i\}_{i \in I_{\mathrm{Tr}}}$, $\Omega$
3: $\prec \leftarrow (\prec_{\mathrm{Pr}}, \prec_{\mathrm{Tr}})$
4: $l \leftarrow (l_{\mathrm{Pr}}, l_{\mathrm{Tr}})$
5: $S \leftarrow S_{\prec,l,\rho}$ $(S \leftarrow S_{\prec,l,\rho,\lambda})$
6: Comp. $L$ using Algorithm 12(13)
7: return $L$
```

Algorithm 22 Predictions last, $n_{\mathrm{Pr}} = 1$

Input: $\mathcal{G}$, $\{x_i\}_{i \in I_{\mathrm{Tr}}}$, Ω, $\{x_i\}_{i \in I_{\mathrm{Tr}}}$, ρ, λ
Output: Cond. mean and var. $\mu, \sigma \in \mathbb{R}^{I_{\mathrm{Pr}}}$

```
1: Comp. $\prec$, $S_{\prec,l,\rho,\lambda}$ from $\{x_i\}_{i \in I_{\mathrm{Tr}}}$, $\Omega$
2: for $k \in I_{\mathrm{Pr}}$ do
3:     $\delta_k, \sigma_k \leftarrow \mathcal{G}(x_k, x_k), \mathcal{G}(x_k, x_k)^{-1}$
4:     $\mu_k \leftarrow 0$
5: end for
6: for $\tilde{k} \in \tilde{I}$ do
7:     $U \leftarrow P^{\updownarrow} \operatorname{chol}(P^{\updownarrow} \Theta_{s_{\tilde{k}}, s_{\tilde{k}}} P^{\updownarrow}) P^{\updownarrow}$
8:     for $k \in s_{\tilde{k}}, l \in I_{\mathrm{Pr}}$ do
9:         $B_{kl} \leftarrow \mathcal{G}(x_k, x_l)$
10:     end for
11:     $B \leftarrow U^{-1} B$
12:     $\tilde{y} \leftarrow U^{-1} y_{s_{\tilde{k}}}$
13:     for $k \rightsquigarrow \tilde{k}$ do
14:         $\alpha \leftarrow \tilde{y}_{s_k}^{\top} B_{s_k,\mathrm{Pr}}$
15:         $\beta \leftarrow B_{s_k,\mathrm{Pr}}^{\top} B_{s_k,\mathrm{Pr}}$
16:         $\gamma \leftarrow \sqrt{1 + (\delta - \beta)^{-1} B_{k,\mathrm{Pr}}^2}$
17:         $\ell \leftarrow -\delta^{-1} \gamma^{-1} B_{k,\mathrm{Pr}}^{\top} \left(1 + \frac{\beta}{\delta - \beta}\right)$
18:         $\mu \leftarrow \mu + \ell / \gamma \left(\tilde{y}_k + \frac{B_{k,\mathrm{Pr}} \alpha}{\delta - \beta}\right)$
19:         $\sigma \leftarrow \sigma + \ell^2$
20:     end for
21: end for
22: $\sigma \leftarrow \sigma^{-1}$
23: $\mu \leftarrow -\sigma \mu$
24: return $\mu, \sigma$
```

Algorithm 23 Prediction variables last

Input: $\mathcal{G}$, $\{x_i\}_{i \in I_{\mathrm{Tr}}}$, Ω, $\{x_i\}_{i \in I_{\mathrm{Tr}}}$, ρ, λ, batched predictions $I_{\mathrm{Pr}} = \bigcup_{b=1}^{m_{\mathrm{Pr}}} J_b$
Output: Per batch cond. mean $\{\mu_b\}_{1 \leq b \leq m_{\mathrm{Pr}}}$ and cov. $\{C_b\}_{1 \leq b \leq m_{\mathrm{Pr}}}$

```
1: Comp. $\prec$, $S_{\prec,l,\rho,\lambda}$ from $\{x_i\}_{i \in I_{\mathrm{Tr}}}$, $\Omega$
2: for $\tilde{k} \in \tilde{I}$ do
3:     $U^{\tilde{k}} \leftarrow P^{\updownarrow} \operatorname{chol}(P^{\updownarrow} \Theta_{s_{\tilde{k}}, s_{\tilde{k}}} P^{\updownarrow}) P^{\updownarrow}$
4: end for
5: for $b \in \{1, \ldots, m_{\mathrm{Pr}}\}$ do
6:     for $\tilde{k} \in \tilde{I}$ do
7:         for $k \in s_{\tilde{k}}, l \in J_b$ do
8:             $(\Theta_{s_{\tilde{k}},b})_{kl} \leftarrow \mathcal{G}(x_k, x_l)$
9:         end for
10:         for $k, l \in J_b$ do
11:             $(\Theta_{b,b})_{kl} \leftarrow \mathcal{G}(x_k, x_l)$
12:         end for
13:         $C_b \leftarrow \Theta_{b,b}^{-1}$
14:         $B^{\tilde{k}} \leftarrow U^{\tilde{k},-1} \Theta_{\tilde{k},b}$
15:         $y^{\tilde{k}} \leftarrow U^{\tilde{k},-1} y_{s_{\tilde{k}}}$
16:         for $k \rightsquigarrow \tilde{k}$ do
17:             $v \leftarrow$
                $B_{s_k,b}^{\tilde{k}} \left(\Theta_{b,b} - B_{s_k,b}^{\tilde{k},\top} B_{s_k,b}\right)^{-1} B_{k,b}^{\tilde{k},\top}$
18:             $c \leftarrow \sqrt{1 + v_k}$
19:             $L_{b,k} \leftarrow -\frac{1}{c_k} \Theta_{b,b}^{-1} B_{s_k,b}^{\tilde{k},\top} (\mathbf{e}_1 + v)$
20:             $C_b \leftarrow C_b + L_{b,k} L_{b,k}^{\top}$
21:             $\mu_b \leftarrow \mu_b + L_{b,k} (\frac{1}{c_k} y_{s_k}^{\tilde{k},\top} (\mathbf{e}_1 + v))$
22:         end for
23:     end for
24:     $\mu_b \leftarrow -C_b \mu_b$
25: end for
26: $C_b \leftarrow C_b^{-1}$
27: return $(\mu_b, C_b)_{1 \leq b \leq m_{\mathrm{Pr}}}$
```

Figure .11: **Algorithms for including prediction points.**

Expanding the matrix multiplications into sums, this can be rewritten as

$$\mathbb{E}\left[X_b | X_{\mathrm{Tr}}\right] = -\left(\bar{L}_{b,b}\bar{L}_{b,b}^{\top} + \sum_{k \in I_{\mathrm{Tr}}} \bar{L}_{b,k} \otimes \bar{L}_{b,k}\right)^{-1} \sum_{k \in I_{\mathrm{Tr}}} \bar{L}_{b,k}\left(y^{\top}\bar{L}_{\mathrm{Tr},k}\right) \tag{75}$$

$$\mathbb{C}\mathrm{ov}\left[X_b | X_{\mathrm{Tr}}\right] = \left(\bar{L}_{b,b}\bar{L}_{b,b}^{\top} + \sum_{k \in I_{\mathrm{Tr}}} \bar{L}_{b,k} \otimes \bar{L}_{b,k}\right)^{-1}. \tag{76}$$

$\bar{L}_{b,b}$ is simply the Cholesky factor of $\Theta_{b,b}^{-1}$. Thus, given $\left(y^{\top}\bar{L}_{\mathrm{Tr},k}, L_{b,k}\right)_{k \rightsquigarrow \tilde{k}}$, the above expressions can be evaluated in computational complexity $O(n_{\mathrm{Pr}}^3 + N_{\mathrm{Tr}} n_{\mathrm{Pr}}^2 + n_{\mathrm{Pr}} \# S)$ in time and $O(n_{\mathrm{Pr}}^2 + \max_{\tilde{l} \in \tilde{I}_k} \# \tilde{l})$ in space. Naively, computing the $\left(y^{\top} L_{\mathrm{Tr},k}, L_{b,k}\right)_{k \rightsquigarrow \tilde{k}}$ for each batch has computational complexity $O(m_{\mathrm{Pr}}(\#\tilde{k} + n_{\mathrm{Pr}})^3)$ which becomes the bottleneck for large numbers of batches. However, as we will see, $\langle L_{\mathrm{Tr},k}, y\rangle$ and $L_{b,k}$ can be computed in computational complexity $O((\#\tilde{k} + n_{\mathrm{Pr}})^3 + m_{\mathrm{Pr}}(\#\tilde{k} + n_{\mathrm{Pr}})^2)$ by reusing parts of the computation. Fix a supernodal index $\tilde{k} \in \tilde{I}$ and define the corresponding exact joint covariance matrix as

$$\Theta^{\tilde{k}} := \left(\Theta_{ij}\right)_{\{i,j \in \tilde{k} \cup J_b\}} = \begin{pmatrix} \Theta_{\tilde{k},\tilde{k}} & \Theta_{\tilde{k},b} \\ \Theta_{b,\tilde{k}} & \Theta_{b,b} \end{pmatrix}. \tag{77}$$

For any $k \rightsquigarrow \tilde{k}$, the column $L^{\rho}_{:,k}$ is, according to (6.3), equal to

$$\frac{\left(\Theta^{\tilde{k}}_{k:,k:}\right)^{-1}\mathbf{e}_1}{\sqrt{\mathbf{e}_1^{\top}\left(\Theta^{\tilde{k}}_{k:,k:}\right)^{-1}\mathbf{e}_1}}. \tag{78}$$

Let as before $U^{\tilde{k}}U^{\tilde{k},\top} = \Theta_{\tilde{k},\tilde{k}}$. Using the Sherman-Morrison-Woodbury matrix identity, we can then rewrite $\Theta^{\tilde{k},-1}_{k:,k:}\mathbf{e}_1$ as

$$\Theta^{\tilde{k},-1}_{k:,k:}\mathbf{e}_1 = \begin{pmatrix} \mathrm{Id} & 0 \\ -\Theta_{b,b}^{-1}\Theta_{b,s_k} & \mathrm{Id} \end{pmatrix}\begin{pmatrix} \left(\Theta_{s_k,s_k} - \Theta_{s_k,b}\Theta_{b,b}^{-1}\Theta_{b,s_k}\right)^{-1} & 0 \\ 0 & \Theta_{b,b}^{-1} \end{pmatrix}\begin{pmatrix} \mathrm{Id} & -\Theta_{s_k,b}\Theta_{b,b}^{-1} \\ 0 & \mathrm{Id} \end{pmatrix}\mathbf{e}_1$$

$$= \begin{pmatrix} \left(\Theta_{s_k,s_k} - \Theta_{s_k,b}\Theta_{b,b}^{-1}\Theta_{b,s_k}\right)^{-1}\mathbf{e}_1 \\ -\Theta_{b,b}^{-1}\Theta_{b,s_k}\left(\Theta_{s_k,s_k} - \Theta_{s_k,b}\Theta_{b,b}^{-1}\Theta_{b,s_k}\right)^{-1}\mathbf{e}_1 \end{pmatrix}$$

$$= \begin{pmatrix} \left(\Theta_{s_k,s_k}^{-1} - \Theta_{s_k,s_k}^{-1}\Theta_{s_k,b}\left(-\Theta_{b,b} + \Theta_{b,s_k}\Theta_{s_k,s_k}^{-1}\Theta_{s_k,b}\right)^{-1}\Theta_{b,s_k}\Theta_{s_k,s_k}^{-1}\right)\mathbf{e}_1 \\ -\Theta_{b,b}^{-1}\Theta_{b,s_k}\left(\Theta_{s_k,s_k}^{-1} - \Theta_{s_k,s_k}^{-1}\Theta_{s_k,b}\left(-\Theta_{b,b} + \Theta_{b,s_k}\Theta_{s_k,s_k}^{-1}\Theta_{s_k,b}\right)^{-1}\Theta_{b,s_k}\Theta_{s_k,s_k}^{-1}\right)\mathbf{e}_1 \end{pmatrix}$$

Using Equation (62) and setting $B^{\tilde{k}} := U^{\tilde{k},-1}\Theta_{\tilde{k},b}$, we obtain

$$\bar{\Theta}^{\tilde{k},-1}_{k:,k:}\mathbf{e}_1 = \frac{1}{U^{\tilde{k}}_{k,k}}\begin{pmatrix} U^{\tilde{k},-\top}_{s_k,s_k}\left(\mathbf{e}_1 + B^{\tilde{k}}_{s_k,b}\left(\Theta_{b,b} - B^{\tilde{k},\top}_{s_k,b}B^{\tilde{k}}_{s_k,b}\right)^{-1}B^{\tilde{k},\top}_{k,b}\right) \\ -\Theta_{b,b}^{-1}B^{\tilde{k},\top}_{s_k,b}\left(\mathbf{e}_1 + B^{\tilde{k}}_{s_k,b}\left(\Theta_{b,b} - B^{\tilde{k},\top}_{s_k,b}B^{\tilde{k}}_{s_k,b}\right)^{-1}B^{\tilde{k},\top}_{k,b}\right) \end{pmatrix}. \tag{79}$$

Setting $y^{\tilde{k}} = U^{\tilde{k},-1} y_{s_{\tilde{k}}}$, this yields the formulae

$$y^{\top} \bar{L}_{\mathrm{Tr},k} = \frac{y_k^{\tilde{k},\top} + y_{s_k}^{\tilde{k},\top} B_{s_k,b}^{\tilde{k}} \left(\Theta_{b,b} - B_{s_k,b}^{\tilde{k},\top} B_{s_k,b}^{\tilde{k}}\right)^{-1} B_{k,b}^{\tilde{k},\top}}{c_k} \tag{80}$$

$$\bar{L}_{b,k} = \frac{-\Theta_{b,b}^{-1} B_{s_k,b}^{\tilde{k},\top} \left(\mathbf{e}_1 + B_{s_k,b}^{\tilde{k}} \left(\Theta_{b,b} - B_{s_k,b}^{\tilde{k},\top} B_{s_k,b}^{\tilde{k}}\right)^{-1} B_{k,b}^{\tilde{k},\top}\right)}{c_k}, \tag{81}$$

where

$$c_k := \sqrt{1 + B_{k,b}^{\tilde{k}} \left(\Theta_{b,b} - B_{s_k,b}^{\tilde{k},\top} B_{s_k,b}^{\tilde{k}}\right)^{-1} B_{k,b}^{\tilde{k},\top}}. \tag{82}$$

Algorithm 23 implements the formulae above. Since $U^{\tilde{k}}$ does not depend on b, it only has to be computed once and can be used to compute the $B^{\tilde{k}}$ and $y^{\tilde{y}}$ for all $1 \leq b \leq m_{\mathrm{Pr}}$.

.3.4 Computation of the reverse maximin ordering and sparsity pattern

We will now explain how to compute the ordering and sparsity pattern described in Section 6.3, using only near-linearly many evaluations of an oracle $\mathtt{dist}(i, j)$ that returns the distance between the points x_i and x_j. To do so efficiently in general, we need to impose a mild additional assumption on the dataset (c.f. Section .2.1).

Condition 7 (Polynomial Scaling)**.** *There exists a polynomial* $\mathbf{p}$ *for which*

$$\frac{\max_{i \neq j \in I} \operatorname{dist}(x_i, x_j)}{\min_{i \neq j \in I} \operatorname{dist}(x_i, x_j)} \leq \mathbf{p}(N).$$

Under Conditions 5 and 7, Algorithm 11 allows us to compute the maximin ordering $\prec$ and sparsity pattern $\left\{(i, j) : \operatorname{dist}(x_i, x_j) \leq \rho \max(\ell_i, \ell_j)\right\}$ in computational complexity $O(N \log(N) \rho^{\tilde{d}})$ in space and time. The resulting pattern is larger than the reverse maximin sparsity pattern $S_{\prec,l\rho}$ of Section 6.3.1, which can thus be obtained by truncating the pattern obtained by Algorithm 11. By performing the truncation of the sets of children and parents c, p as used by Algorithm 11 during execution of the algorithm, as opposed to truncating the sparsity pattern after execution of the algorithm, the space complexity for obtaining $\prec$ and $S_{\prec,l,\rho}$ can be reduced to $O(N\rho^{\tilde{d}})$. Algorithm 24 is a minor modification of Algorithm 11 that performs such a truncation.

Theorem 32 (Variant of Algorithm 11)**.** *Let* $\Omega = \mathbb{R}^d$ *and* $\rho \geq 2$. *Algorithm 24 computes the reverse maximin ordering* $\prec$ *and sparsity pattern* $S_{\prec,l,\rho}$ *in computational complexity* $C\rho^{\tilde{d}}N$ *in space and* $CN\log(N)\rho^{\tilde{d}}(\log N + C_{\mathtt{dist}})$ *in time. Here,*

$C = C(\tilde{d}, C_{\tilde{d}}, \mathbf{p})$ *depends only on the constants appearing in Conditions 5 and refcond:polynomialScaling, and* $C_{\texttt{dist}}$ *is the cost of evaluating* `dist`.

Proof. The proof is essentially the same as the proof of Theorem 19. □

Similarly, the proof of Theorem 27 can be adapted to show that in the setting of Theorem 22, there exists a constant C depending only on d, Ω, and δ, such that for $\rho > C$, Algorithm 24 computes the maximin ordering in computational complexity $CN(\log(N)\rho^d + C_{\texttt{dist}_\Omega})$ in time and $CN\rho^d$ in space, where $C_{\texttt{dist}_\Omega}$ is an upper bound on the complexity of computing the distance of an arbitrary point $x \in \Omega$ to $\partial\Omega$.

We furthermore note that a reverse r-maximin ordering with $r < 1$ (see Definition 20) can be computed in computational complexity $O(N \log(N))$ by quantizing the values of $(\log(\ell_i))_{i \in I}$ in multiples of $\log(r)$, which avoids the complexity incurred by the restoration of the heap property in Line 23 of Algorithm 11.

As described in Section 6.3.2, the aggregated sparsity pattern $S_{\prec,l,\rho,\lambda}$ can be computed efficiently from $S_{\prec,l,\rho}$. However, forming the pattern $S_{\prec,l,\rho}$ using a variant of Algorithm 11 has complexity $O(N \log(N)\rho^{\tilde{d}})$ in time and $O(N\rho^{\tilde{d}})$ in space, while the aggregated pattern $S_{\prec,l,\rho,\lambda}$ only has space complexity $O(N)$, begging the question if this computational complexity can be improved. Let $\{s_{\tilde{i}}\}_{\tilde{i} \in \tilde{I}}$ be the supernodes as constructed in Section 6.3.2 and identify each supernodal index $\tilde{i}$ with the first (w.r.t. $\prec$) index $i \in I$ such that $i \rightsquigarrow \tilde{i}$. We then define

$$\bar{S}_{\prec,l,\rho,\lambda} \coloneqq \bigcup_{\tilde{i} \in \tilde{I}} \left\{(i,j) : i \preceq j, i \rightsquigarrow \tilde{i}, \operatorname{dist}(x_{\tilde{i}}, x_j) \leq \rho(1+\lambda)\tilde{i}\right\}. \tag{83}$$

Algorithm 25 allows us to construct the sparsity pattern $\bar{S}_{\prec,l,\rho,\lambda} \supset \tilde{S}_{\prec,l,\rho,\lambda}$ in complexity $O(N \log(N))$ in time and $O(N)$ in space, given $\prec$ and l. In this algorithm, we will implement supernodes as pairs of arrays of indices $\sigma = (\sigma_m, \sigma_n)$. This encodes the relationship between all indices in σ_m (the *parents*) and all indices in σ_n (the *children*). Naively, this would require $O(\#\sigma_m \#\sigma_n)$ space complexity, but by storing the entries of σ_m and σ_n, the complexity is reduced to $O(\#\sigma_m + \#\sigma_n)$ space complexity, which improves the asymptotic computational complexity.

Algorithm 24 Ordering and sparsity pattern algorithm (cf. Algorithm 11).

Input: A real parameter $\rho \geq 2$ and Oracles $\texttt{dist}(\cdot, \cdot), \texttt{dist}_{\partial\Omega}(\cdot)$ such that $\texttt{dist}(i, j) = \operatorname{dist}(x_i, x_j)$ and $\texttt{dist}_{\partial\Omega}(i) = \operatorname{dist}(x_i, \partial\Omega)$

Output: An array $l[:]$ of distances, an array P encoding the multiresolution ordering, and an array of index pairs S containing the sparsity pattern.

```
1: $P = \emptyset$
2: for $i \in \{1, \dots, N\}$ do
3:     $l[i] \leftarrow \texttt{dist}_{\partial\Omega}(i)$
4:     $p[i] \leftarrow \emptyset$
5:     $c[i] \leftarrow \emptyset$
6: end for
7: {Creates a mutable binary heap, containing pairs of indices and distances as elements:}
8: $H \leftarrow \texttt{MutableMaximalBinaryHeap}\left(\{(i, l[i])\}_{i \in \{1,\dots,N\}}\right)$
9: {Instates the Heap property, with a pair with maximal distance occupying the root of the heap:}
10: $\texttt{heapSort!}(H)$
11: {Processing the first index:}
12: {Get the root of the heap, remove it, and restore the heap property:}
13: $(i, l) = \texttt{pop}(H)$
14: {Add the index as the next element of the ordering}
15: $\texttt{push}(P, i)$
16: for $j \in \{1, \dots, N\}$ do
17:     $\texttt{push}(c[i], j)$
18:     $\texttt{push}(p[j], i)$
19:     $\texttt{sort!}(c[i], \texttt{dist}(\cdot, i))$
20:     $\texttt{decrease!}(H, j, \texttt{dist}(i, j))$
21: end for
22: {Processing remaining indices:}
23: $l_{\text{Trunc}} \leftarrow l$
24: while $H \neq \emptyset$ do
25:     {Get the root of the heap, remove it, and restore the heap property:}
26:     $(i, l) = \texttt{pop}(H)$
27:     $l[i] \leftarrow l$
28:     {Select the parent that has possible children of $i$ amongst its children, and is closest to $i$:}
29:     $k = \arg\min_{j \in p[i] : \texttt{dist}(i,j) + \rho l[i] \leq \rho \min(l_{\text{Trunc}}, l[j])} \texttt{dist}(i, j)$
30:     {Loop through those children of $k$ that are close enough to $k$ to possibly be children of $i$:}
31:     for $j \in c[k] : \texttt{dist}(j, k) \leq \texttt{dist}(i, k) + \rho l[i]$ do
32:         $\texttt{decrease!}(H, j, \texttt{dist}(i, j))$
33:         if $\texttt{dist}(i, j) \leq \rho l[i]$ then
34:             $\texttt{push}(c[i], j)$
35:             $\texttt{push}(p[j], i)$
36:         end if
37:     end for
38:     {Add the index as the next element of the ordering}
39:     $\texttt{push}(P, i)$
40:     {Sort the children according to distance to the parent node, so that the closest children can be found more easily}
41:     $\texttt{sort!}(c[i], \texttt{dist}(\cdot, i))$
42:     {Truncate the sparsity pattern to achieve linear space complexity}
43:     if $\forall j \notin P, \exists i \in P : \texttt{dist}(i, j) < l_{\text{Trunc}}/2$ then
44:         $l_{\text{Trunc}} \leftarrow l_{\text{Trunc}}/2$
45:         for $j \in c[i] \setminus P, \texttt{dist}(i, j) > \rho l_{\text{Trunc}}$ do
46:             $c[i] \leftarrow c[i] \setminus \{j\}$
47:             $p[j] \leftarrow p[j] \setminus \{i\}$
48:         end for
49:     end if
50: end while
51: {Aggregating the lists of children into the sparsity pattern:}
52: for $i \in \{1, \dots, N\}$ do
53:     for $j \in c[i]$ do
54:         $\texttt{push!}(S, (i, j))$
55:         $\texttt{push!}(S, (j, j))$
56:     end for
57: end for
```

Algorithm 25 Computation of $\tilde{S}$ and $\bar{S}$

```
Input: $I, \prec, l, \mathtt{dist}(\cdot,\cdot), \rho, \lambda$
Output: Sets of supernodes $\bar{S}, \tilde{S}$

 1: $i_N, i_{N-1} \leftarrow$ last two ind. w.r.t. $\prec$
 2: $\mathcal{N}, \mathcal{N}^{\geq} \leftarrow \{(\{i_N\}, I)\}, \{(\{i_N\}, \{i_N\})\}$
 3: $r, l_{i_N} \leftarrow l_{i_{N-1}}/\lambda, \infty$
 4: while $r > \min_{i \in I} l_i$ do
 5:     $\mathcal{N}, \tilde{\mathcal{N}}^{\geq} \leftarrow \mathtt{Refine}(\mathcal{N}, l, r, \rho, \lambda)$
 6:     $\mathcal{N}^{\geq} \leftarrow \mathcal{N}^{\geq} \cup \tilde{\mathcal{N}}^{\geq}$
 7:     $r \leftarrow r/\lambda$
 8: end while
 9: $J, \bar{S} \leftarrow \emptyset, \emptyset$
10: for $i \in I$ (in increasing order by $\prec$) do
11:     if $i \notin J$ then
12:         $\tilde{s} \leftarrow (\emptyset, \emptyset)$, Pick $\sigma \in \mathcal{N}^{\geq} : i \in \sigma_m$
13:         for $j \in \sigma_m$ do
14:             if $i \preceq j, l_j \leq \lambda l_i, j \notin J$ then
15:                 $J, \tilde{s}_n \leftarrow J \cup \{j\}, \tilde{s}_n \cup \{j\}$
16:             end if
17:         end for
18:         for $\tilde{\sigma} \in \mathcal{N}^{\geq} : \exists j \in \tilde{\sigma}_m : j \in \sigma_m$ do
19:             for $k \in \tilde{\sigma}_n : \mathtt{dist}(i,k) \leq \rho(1+\lambda)$ do
20:                 $\tilde{s}_m \leftarrow \tilde{s}_m \cup \{k\}$
21:             end for
22:         end for
23:         $\bar{S} \leftarrow \bar{S} \cup \{\tilde{s}\}$
24:     end if
25: end for
26: $\tilde{S} \leftarrow \mathtt{Reduce}(\rho, \prec, l, \bar{S})$
27: return $\bar{S}, \tilde{S}$
```

Algorithm 26 $\mathtt{Reduce}(\rho, \prec, l, \bar{S})$

```
Input: $\prec, l, \mathtt{dist}(\cdot,\cdot), \rho, \bar{S}$
Output$\tilde{S} = \tilde{S}_{\prec,l,\rho}$

 1: $\tilde{S} \leftarrow \emptyset$
 2: for $\sigma \in \bar{S}$ do
 3:     $\tilde{s} \leftarrow (\sigma_m, \emptyset)$
 4:     for $i \in \sigma_m, j \in \sigma_n$ do
 5:         if $\mathtt{dist}(i,j) \leq \rho l_i$ and $i \prec j$ then
 6:             $\tilde{s}_n \leftarrow \tilde{s}_n \cup \{j\}$
 7:         end if
 8:     end for
 9:     $\tilde{S} \leftarrow \tilde{S} \cup \{\tilde{s}\}$
10: end for
11: return $\tilde{S}$
```

Algorithm 27 $\mathtt{Refine}(\mathcal{N}, l, r, \rho, \lambda)$

```
Input: Supernodal set $\mathcal{N}$, $\mathtt{dist}(\cdot,\cdot), l, r, \rho, \lambda$
Output: A new set $\mathcal{M}$ of supernodes, set $\mathcal{N}^{\geq}$ of truncated supernodes

 1: $J, \mathcal{N}^{\geq}, \mathcal{M} \leftarrow \emptyset, \emptyset, \emptyset$
 2: while $J \neq I$ do
 3:     Pick $i \in I \setminus J$
 4:     $J \leftarrow J \cup \{i\}$
 5:     Pick $\sigma \in \mathcal{N}$ satisfying $i \in \sigma_m$
 6:     $\tilde{\sigma}_m, \tilde{\sigma}_n \leftarrow \emptyset, \emptyset$
 7:     for $j \in \sigma_n$ do
 8:         if $\mathtt{dist}(i,j) \leq \rho\lambda r$ then
 9:             $\tilde{\sigma}_m \leftarrow \tilde{\sigma}_m \cup \{j\}$
10:             $J \leftarrow J \cup \{j\}$
11:         end if
12:         if $\mathtt{dist}(i,j) \leq 2\rho\lambda r$ then
13:             $\tilde{\sigma}_n \leftarrow \tilde{\sigma}_n \cup \{j\}$
14:         end if
15:     end for
16:     $\mathcal{M} \leftarrow \mathcal{M} \cup \{(\tilde{\sigma}_m, \tilde{\sigma}_n)\}$
17: end while
18: for $\sigma \in \mathcal{N}$ do
19:     $\sigma_m^{\geq}, \sigma_n^{\geq} \leftarrow \emptyset, \emptyset$
20:     for $i \in \sigma_m$ do
21:         if $r \leq l_i$ then
22:             $\sigma_m^{\geq} \leftarrow \sigma_m^{\geq} \cup \{i\}$
23:         end if
24:     end for
25:     for $i \in \sigma_n$ do
26:         if $r \leq l_i$ then
27:             $\sigma_n^{\geq} \leftarrow \sigma_n^{\geq} \cup \{i\}$
28:         end if
29:     end for
30:     $\mathcal{N}^{\geq} \leftarrow \mathcal{N}^{\geq} \cup \{(\sigma_m^{\geq}, \sigma_n^{\geq})\}$
31: end for
32: return $\mathcal{M}, \mathcal{N}^{\geq}$
```

Figure .12: **Aggregation.** Algorithm for constructing the aggregated sparsity pattern from the reverse maximin ordering $\prec$ and length-scales l

Theorem 33. *Let the $\{x_i\}_{i \in I}$ satisfy Condition 5 with $\tilde{d}$ and $C_{\tilde{d}}$, Condition 6 with constant C_{regref}, and Condition 7 with $\mathbf{p}$ and let $\lambda > 1$. Then there exists a constant $C = C_{\tilde{d}, C_{\tilde{d}}, C_{\mathrm{regref}}, \mathbf{p}, \lambda}$ such that Algorithm 25 can compute $\bar{S}_{<,l,\rho,\lambda}$ in computational complexity $CC_{\mathtt{dist}} N \log(N)$ in time and CN in space and $\tilde{S}_{<,l,\rho,\lambda}$ in computational complexity $CC_{\mathtt{dist}} N(\log(N) + \rho^{\tilde{d}})$ in time and CN in space.*

Proof. To establish correctness, the main observation is that after every application of Algorithm 27, each degree of freedom i can be found in exactly one of the supernodes $\sigma \in \mathcal{N}$. Furthermore, each $\sigma \in \mathcal{N}$ has a *root* $\sqrt{\sigma} \in I$ such that

1. $\sqrt{\sigma} \in \sigma_n, \sigma_m$,
2. $j \in \sigma_m \Rightarrow \mathtt{dist}(\sqrt{\sigma}, j) \leq \rho\lambda r$,
3. $j \in \sigma_n \Leftrightarrow \mathtt{dist}(\sqrt{\sigma}, j) \leq 2\rho\lambda r$,
4. $\sigma \neq \bar{\sigma} \Rightarrow \mathtt{dist}(\sqrt{\sigma}, \sqrt{\bar{\sigma}}) > \rho\lambda r$.

The main reason why the above could fail to hold true is that the inner for loop does not range over all $j \in I$, but only over those in σ_n. However, at the first occurrence of Algorithm 27 we have $\sigma_n = I$ leading to the observations to hold true. For subsequent calls, we can show the invariance of these properties by induction. The set $\mathcal{N}^{\geq}$ is obtained from the set $\mathcal{N}$ by only selecting the points in a certain range of length scales. Therefore, after completion of the while-loop of Algorithm 25, every $i \in I$ is contained in at least one of the $\{\sigma_m\}_{\sigma \in \mathcal{N}^{\geq}}$, and for $i \rightsquigarrow \sigma \in \mathcal{N}^{\geq}$, we have $\left\{j : (i,j) \in \bar{S}\right\} \subset \sigma_n$. Thus, the for-loop of Algorithm 25 indeed computes $\bar{S}$. Since $\tilde{S}_{<,l,\rho,\lambda} \subset \bar{S}_{<,l,\rho,\lambda}$, Algorithm 26 correctly recovers $\tilde{S}_{<,l,\rho,\lambda}$.

We begin by analyzing the computational complexity of the while-loop of Algorithm 25. We first note that at every execution of the loop, r is divided by λ. Thus, Condition 7 implies that the loop is entered at most $C \log(N)$ times. We now claim that that the time complexity of Algrithm 27 is bounded above by $CC_{\mathtt{dist}} N$. To this end it is enough to upper-bound the number of points i for which a given index can be picked as index j in the while-loop of Algorithm 27. By Property 2, for this to happen we need $\mathtt{dist}(i, \sqrt{\sigma}) \leq \rho\lambda r$ and $\mathtt{dist}(j, \sqrt{\sigma}) \leq 2\rho\lambda r$ and hence, by the triangle inequality, $\mathtt{dist}(i,j) \leq 3\rho\lambda r$. On the other hand, i can not be in J already, which means that any two distinct i_1, i_2 have to satisfy $\mathtt{dist}(i_1, i_2) > \rho\lambda r$. By Condition 5, we conclude that the maximum number of indices i for which a given index j gets picked is bounded above by a constant C that depends of $C_{\tilde{d}}$ and d. This upper bounds the computational complexity of the while-loop in Algorithm 27 by CN. The computational complexity of the outermost for-loop of Algorithm 27 can be bounded by CN, by a similar argument. Summarizing the above, we have upper-bounded the time complexity of the while-loop in Algorithm 25 by $CN \log(N)$. In order to bound the space complexity of the while-loop, we need to ensure that the size of $\mathcal{N}^{\geq}$ is bounded above as CN. To this end, we notice that by arguments similar to the above, one can show that at all times, the number $\max_{i \in I} \#\{\sigma \in \mathcal{N} : i \in \sigma_m \text{ or } i \in \sigma_N\}$ is bounded from above by a constant C. By using Condition 6, we can show that the space complexity of $\mathcal{N}^{\geq}$ is bounded by CN. By ways of similar ball packing arguments, the complexity of the outer for-loop of Algorithm 25 can be bounded by CN, as well. The complexity of Algorithm26 is bounded by $CN\rho^{\tilde{d}}$, the number of entries of the sparsity pattern $\bar{S}$, since it iterates over all entries of $\bar{S}$. □

.4 Appendix to Chapter 7

.4.1 Proofs of convergence

Proof of Theorem 2.3. To shorten the expressions below, we set $a := \nabla_x f(x_k)$, $b := \nabla_y f(x_k, y_k)$, $H_{xx} := D^2_{xx} f(x_k, y_k)$, $H_{yy} := D^2_{yy} f(x_k, y_k)$, $N := D^2_{xy} f(x_k, y_k)$, $\tilde{N} := \eta N$, $\tilde{M} := \tilde{N}^\top \tilde{N}$, and $\bar{M} := \tilde{N}\tilde{N}^\top$.

Letting (x, y) be the update step of CGD and using the Taylor expansion, we obtain

$$\begin{aligned}
\nabla_x f(x + x_k, y + y_k) &= a + H_{xx} x + N y + \mathcal{R}_x(x, y) \\
\nabla_y f(x + x_k, y + y_k) &= b + H_{yy} y + N^\top x + \mathcal{R}_y(x, y)
\end{aligned}$$

where the remainder terms $\mathcal{R}_x$ and $\mathcal{R}_y$ are defined as

$$\mathcal{R}_x(x, y) := \int_0^1 \left(D^2_{xx} f(tx + x_k, ty + y_k) - H_{xx}\right) x + \left(D^2_{xy} f(tx + x_k, ty + y_k) - N\right) y \,\mathrm{d}t$$

$$\mathcal{R}_y(x, y) := \int_0^1 \left(D^2_{yy} f(tx + x_k, ty + y_k) - H_{yy}\right) y + \left(D^2_{yx} f(tx + x_k, ty + y_k) - N^\top\right) x \,\mathrm{d}t.$$

Using this formula, we obtain

$$\begin{aligned}
&\|\nabla_x f(x + x_k, y + y_k)\|^2 + \left\|\nabla_y f(x + x_k, y + y_k)\right\|^2 - \|a\|^2 - \|b\|^2 \\
&= 2x^\top H_{xx} a + 2a^\top N y + x^\top H_{xx} H_{xx} x + 2x^\top H_{xx} N y + y^\top N^\top N y \\
&+ 2y^\top H_{yy} b + 2b^\top N^\top x + y^\top H_{yy} H_{yy} y + 2y^\top H_{yy} N^\top x + x^\top N N^\top x \\
&+ 2a^\top \mathcal{R}_x(x, y) + 2x^\top H_{xx} \mathcal{R}_x(x, y) + 2y^\top N^\top \mathcal{R}_x(x, y) + \|\mathcal{R}_x(x, y)\|^2 \\
&+ 2b^\top \mathcal{R}_y(x, y) + 2y^\top H_{yy} \mathcal{R}_y(x, y) + 2x^\top N \mathcal{R}_y(x, y) + \|\mathcal{R}_y(x, y)\|^2.
\end{aligned}$$

We now observe that

$$\begin{aligned}
y^\top N^\top N y &= y^\top N^\top (-x/\eta - a) \\
x^\top N N^\top x &= x^\top N (y/\eta - b).
\end{aligned}$$

Thus, by adding up the two terms we obtain

$$x^\top N N^\top x + y^\top N^\top N y = -y^\top N^\top a - x^\top N b.$$

Plugging this into our computation yields

$$\begin{aligned}
&\|\nabla_x f(x + x_k, y + y_k)\|^2 + \left\|\nabla_y f(x + x_k, y + y_k)\right\|^2 - \|a\|^2 - \|b\|^2 \\
&= 2x^\top H_{xx} a + a^\top N y + x^\top H_{xx} H_{xx} x + 2x^\top H_{xx} N y \\
&+ 2y^\top H_{yy} b + b^\top N^\top x + y^\top H_{yy} H_{yy} y + 2y^\top H_{yy} N^\top x \\
&+ 2a^\top \mathcal{R}_x(x, y) + 2x^\top H_{xx} \mathcal{R}_x(x, y) + 2y^\top N^\top \mathcal{R}_x(x, y) + \|\mathcal{R}_x(x, y)\|^2 \\
&+ 2b^\top \mathcal{R}_y(x, y) + 2y^\top H_{yy} \mathcal{R}_y(x, y) + 2x^\top N \mathcal{R}_y(x, y) + \|\mathcal{R}_y(x, y)\|^2.
\end{aligned}$$

We now plug the update rule of CGD into x and y, and observe that $\tilde{N}^\top(\mathrm{Id} + \bar{M})^{-1} = (\mathrm{Id} + \tilde{M})^{-1}\tilde{N}^\top$ to obtain

$$x^\top N b + a^\top N y = -a^\top \left(\mathrm{Id} + \bar{M}\right)^{-1} \bar{M} a - b^\top \left(\mathrm{Id} + \tilde{M}\right)^{-1} \tilde{M} b,$$

yielding

$$\begin{aligned}
&\|\nabla_x f(x + x_k, y + y_k)\|^2 + \left\|\nabla_y f(x + x_k, y + y_k)\right\|^2 - \|a\|^2 - \|b\|^2 \\
&= 2x^\top H_{xx} a - a^\top \left(\mathrm{Id} + \bar{M}\right)^{-1} \bar{M} a + x^\top H_{xx} H_{xx} x + 2x^\top H_{xx} N y \\
&+ 2y^\top H_{yy} b - b^\top \left(\mathrm{Id} + \tilde{M}\right)^{-1} \tilde{M} b + y^\top H_{yy} H_{yy} y + 2y^\top H_{yy} N^\top x \\
&+ 2a^\top \mathcal{R}_x(x, y) + 2x^\top H_{xx} \mathcal{R}_x(x, y) + 2y^\top N^\top \mathcal{R}_x(x, y) + \|\mathcal{R}_x(x, y)\|^2 \\
&+ 2b^\top \mathcal{R}_y(x, y) + 2y^\top H_{yy} \mathcal{R}_y(x, y) + 2x^\top N \mathcal{R}_y(x, y) + \|\mathcal{R}_y(x, y)\|^2.
\end{aligned}$$

In the above, we have used cancellation across the two players in order to turn the purely interactive terms into terms that induce convergence. We now observe that $\eta N y + \eta a = -x$ and $\eta N^\top x + \eta b = y$ by Equation (7.8), yielding

$$
\begin{aligned}
&\|\nabla_x f(x + x_k, y + y_k)\|^2 + \left\|\nabla_y f(x + x_k, y + y_k)\right\|^2 - \|a\|^2 - \|b\|^2 \\
&= -2\eta a^\top H_{xx} a - a^\top \left(\mathrm{Id} + \bar{M}\right)^{-1} \bar{M} a + x^\top H_{xx} H_{xx} x + 2\left(2x + \eta N y\right)^\top H_{xx} N y \\
&+ 2\eta b^\top H_{yy} b - b^\top \left(\mathrm{Id} + \tilde{M}\right)^{-1} \tilde{M} b + y^\top H_{yy} H_{yy} y + 2\left(2y - \eta N^\top x\right)^\top H_{yy} N^\top x \\
&+ 2a^\top \mathcal{R}_x(x, y) + 2x^\top H_{xx} \mathcal{R}_x(x, y) + 2y^\top N^\top \mathcal{R}_x(x, y) + \|\mathcal{R}_x(x, y)\|^2 \\
&+ 2b^\top \mathcal{R}_y(x, y) + 2y^\top H_{yy} \mathcal{R}_y(x, y) + 2x^\top N \mathcal{R}_y(x, y) + \|\mathcal{R}_y(x, y)\|^2.
\end{aligned}
$$

The terms $-2\eta a^\top H_{xx} a - a^\top \left(\mathrm{Id} + \bar{M}\right)^{-1} \bar{M} a$ and $2\eta b^\top H_{yy} b - b^\top \left(\mathrm{Id} + \tilde{M}\right)^{-1} \tilde{M} b$ allow us to use curvature and interaction to show convergence. We now want to estimate the remaining terms.

To do so, we first use the Peter-Paul inequality to collect occurrences of H_{xx} and H_{yy}.

$$
\begin{aligned}
&\|\nabla_x f(x + x_k, y + y_k)\|^2 + \left\|\nabla_y f(x + x_k, y + y_k)\right\|^2 - \|a\|^2 - \|b\|^2 \\
&\leq -2\eta a^\top H_{xx} a - a^\top \left(\mathrm{Id} + \bar{M}\right)^{-1} \bar{M} a + 18 x^\top H_{xx} H_{xx} x + y^\top N^\top \left(\frac{1}{4}\mathrm{Id} + \eta H_{xx}\right) N y \\
&+ 2\eta b^\top H_{yy} b - b^\top \left(\mathrm{Id} + \tilde{M}\right)^{-1} \tilde{M} b + 18 y^\top H_{yy} H_{yy} y + x^\top N \left(\frac{1}{4}\mathrm{Id} + \eta H_{yy}\right) N^\top x \\
&+ 2a^\top \mathcal{R}_x(x, y) + 2y^\top N^\top \mathcal{R}_x(x, y) + 2\|\mathcal{R}_x(x, y)\|^2 \\
&+ 2b^\top \mathcal{R}_y(x, y) + 2x^\top N \mathcal{R}_y(x, y) + 2\|\mathcal{R}_y(x, y)\|^2.
\end{aligned}
$$

Assuming now that $\|\eta H_{xx}\|, \|\eta H_{yy}\| \leq c$, we can estimate

$$
\begin{aligned}
&\|\nabla_x f(x + x_k, y + y_k)\|^2 + \left\|\nabla_y f(x + x_k, y + y_k)\right\|^2 - \|a\|^2 - \|b\|^2 \\
&\leq -2\eta a^\top H_{xx} a - a^\top \left(\mathrm{Id} + \bar{M}\right)^{-1} \bar{M} a + 18 x^\top H_{xx} H_{xx} x + \left(\frac{1}{4} + c\right) \|N y\|^2 \\
&+ 2\eta b^\top H_{yy} b - b^\top \left(\mathrm{Id} + \tilde{M}\right)^{-1} \tilde{M} b + 18 y^\top H_{yy} H_{yy} y + \left(\frac{1}{4} + c\right) \left\|N^\top x\right\|^2 \\
&+ 2a^\top \mathcal{R}_x(x, y) + 2y^\top N^\top \mathcal{R}_x(x, y) + 2\|\mathcal{R}_x(x, y)\|^2 \\
&+ 2b^\top \mathcal{R}_y(x, y) + 2x^\top N \mathcal{R}_y(x, y) + 2\|\mathcal{R}_y(x, y)\|^2.
\end{aligned}
$$

Using Peter-Paul again, we collect the $Ny, N^\top x$ terms

$$
\begin{aligned}
&\|\nabla_x f(x+x_k, y+y_k)\|^2 + \left\|\nabla_y f(x+x_k, y+y_k)\right\|^2 - \|a\|^2 - \|b\|^2 \\
&\leq -2\eta a^\top H_{xx} a - a^\top \left(\mathrm{Id} + \bar{M}\right)^{-1} \bar{M} a + 18 x^\top H_{xx} H_{xx} x + \left(\frac{1}{2} + c\right) \|Ny\|^2 \\
&+ 2\eta b^\top H_{yy} b - b^\top \left(\mathrm{Id} + \tilde{M}\right)^{-1} \tilde{M} b + 18 y^\top H_{yy} H_{yy} y + \left(\frac{1}{2} + c\right) \left\|N^\top x\right\|^2 \\
&+ 2a^\top \mathcal{R}_x(x,y) + 6\|\mathcal{R}_x(x,y)\|^2 \\
&+ 2b^\top \mathcal{R}_y(x,y) + 6\|\mathcal{R}_y(x,y)\|^2.
\end{aligned}
$$

We now compute

$$
\begin{aligned}
\left\|N^\top x\right\|^2 &= \left(a + \tilde{N} b\right)^\top \left(\mathrm{Id} + \bar{M}\right)^{-2} \bar{M} \left(a + \tilde{N} b\right) \\
\|Ny\|^2 &= \left(-b + \tilde{N}^\top a\right)^\top \left(\mathrm{Id} + \tilde{M}\right)^{-2} \tilde{M} \left(-b + \tilde{N}^\top a\right).
\end{aligned}
$$

By adding up the two, we obtain

$$
\begin{aligned}
\left\|N^\top x\right\|^2 + \|Ny\|^2 &= a^\top \left(\mathrm{Id} + \bar{M}\right)^{-2} \left(\bar{M} + \bar{M}^2\right) a + b^\top \left(\mathrm{Id} + \tilde{M}\right)^{-2} \left(\tilde{M} + \tilde{M}^2\right) b \\
&= a^\top \left(\mathrm{Id} + \bar{M}\right)^{-1} \bar{M} a + b^\top \left(\mathrm{Id} + \tilde{M}\right)^{-1} \tilde{M} b.
\end{aligned}
$$

Plugging this into our main computation, we obtain

$$
\begin{aligned}
&\|\nabla_x f(x+x_k, y+y_k)\|^2 + \left\|\nabla_y f(x+x_k, y+y_k)\right\|^2 - \|a\|^2 - \|b\|^2 \\
&\leq -2\eta a^\top H_{xx} a - \left(\frac{1}{2} + c\right) a^\top \left(\mathrm{Id} + \bar{M}\right)^{-1} \bar{M} a + 18 x^\top H_{xx} H_{xx} x \\
&+ 2\eta b^\top H_{yy} b - \left(\frac{1}{2} - c\right) b^\top \left(\mathrm{Id} + \tilde{M}\right)^{-1} \tilde{M} b + 18 y^\top H_{yy} H_{yy} y \\
&+ 2a^\top \mathcal{R}_x(x,y) + 6\|\mathcal{R}_x(x,y)\|^2 \\
&+ 2b^\top \mathcal{R}_y(x,y) + 6\|\mathcal{R}_y(x,y)\|^2.
\end{aligned}
$$

We now set $c = 1/18$ and define the spectral function $\phi(\lambda) \coloneqq 2\lambda - |\lambda|$ to obtain

$$\begin{aligned}
&\|\nabla_x f(x+x_k, y+y_k)\|^2 + \left\|\nabla_y f(x+x_k, y+y_k)\right\|^2 - \|a\|^2 - \|b\|^2 \\
&\leq -2\eta a^\top h_\pm(H_{xx})\, a - \frac{1}{3} a^\top \left(\mathrm{Id} + \bar{M}\right)^{-1} \bar{M} a \\
&- 2\eta b^\top h_\pm\left(-H_{yy}\right) b - \frac{1}{3} b^\top \left(\mathrm{Id} + \tilde{M}\right)^{-1} \tilde{M} b \\
&+ 2a^\top \mathcal{R}_x(x,y) + 6\|\mathcal{R}_x(x,y)\|^2 \\
&+ 2b^\top \mathcal{R}_y(x,y) + 6\|\mathcal{R}_y(x,y)\|^2.
\end{aligned}$$

To conclude, we need to estimate the $\mathcal{R}$-terms. Using the Lipschitz-continuity of the Hessian, we can estimate

$$\|\mathcal{R}_x(x,y)\|, \|\mathcal{R}_x(x,y)\| \leq (\|x\|+\|y\|)^2 \leq 4\eta^2 L(\|a\|+\|b\|)^2 \leq 8\eta^2 L(\|a\|^2+\|b\|^2). \tag{84}$$

Using this estimate, we obtain

$$\begin{aligned}
&\|\nabla_x f(x+x_k, y+y_k)\|^2 + \left\|\nabla_y f(x+x_k, y+y_k)\right\|^2 - \|a\|^2 - \|b\|^2 \\
&\leq -2\eta a^\top h_\pm(H_{xx})\, a - \frac{1}{3} a^\top \left(\mathrm{Id} + \bar{M}\right)^{-1} \bar{M} a \\
&- 2\eta b^\top h_\pm\left(-H_{yy}\right) b - \frac{1}{3} b^\top \left(\mathrm{Id} + \tilde{M}\right)^{-1} \tilde{M} b \\
&+ 32\eta^2 L\left(\|a\| + \|b\|\right)\left(\|a\|^2 + \|b\|^2\right) + 768\eta^4 L^2(\|a\|^2 + \|b\|^2).
\end{aligned}$$

Rearranging terms, we finally obtain

$$\begin{aligned}
&\|\nabla_x f(x+x_k, y+y_k)\|^2 + \left\|\nabla_y f(x+x_k, y+y_k)\right\|^2 - \|a\|^2 - \|b\|^2 \\
&\leq - a^\top \left(2\eta h_\pm(H_{xx}) + \frac{1}{3}\left(\mathrm{Id} + \bar{M}\right)^{-1} \bar{M} - 32\eta^2 L\left(\|a\| + \|b\|\right) - 768\eta^4 L^2\right) a \\
&\quad - b^\top \left(2\eta h_\pm\left(-H_{yy}\right) + \frac{1}{3}\left(\mathrm{Id} + \tilde{M}\right)^{-1} \tilde{M} - 32\eta^2 L\left(\|a\| + \|b\|\right) - 768\eta^4 L^2\right) b.
\end{aligned}$$

□

Theorem 2.4 follows from Theorem 2.3 by relatively standard arguments:

Proof of Theorem 2.4. Since $\nabla_x f(x^*, y^*), \nabla_x f(x^*, y^*) = 0$ and the gradient and Hessian of f are continuous, there exists a neighborhood $\mathcal{V}$ of (x^*, y^*) such that for all possible starting points $(x_1, y_1) \in \mathcal{V}$, we have $\|(\nabla_x f(x_2, y_2), \nabla_y f(x_2, y_2)\| \leq (1 - \lambda_{\min}/4)\|(\nabla_x f(x_1, y_1), \nabla_y f(x_1, y_1)\|$. Then, by convergence of the geometric series, there exists a closed neighborhood $\mathcal{U} \subset \mathcal{V}$ of (x^*, y^*), such that for $(x_0, y_0) \in \mathcal{U}$, we have $(x_k, y_k) \in \mathcal{V}, \forall k \in \mathbb{N}$ and thus (x_k, y_k) converges at an exponential rate to a point in $\mathcal{U}$. $\square$

.4.2 Details regarding the experiments

Experiment: Estimating a covariance matrix

We consider the problem $-g(V, W) = f(W, V) = \sum_{ijk} W_{ij} \left(\hat{\Sigma}_{ij} - (V\hat{\Sigma}V^\top)_{i,j}\right)$, where the $\hat{\Sigma}$ are empirical covariance matrices obtained from samples distributed according to $\mathcal{N}(0, \Sigma)$. For our experiments, the matrix Σ is created as $\Sigma = UU^T$, where the entries of $U \in \mathbb{R}^{d \times d}$ are distributed i.i.d. standard Gaussian. We consider the algorithms OGDA, SGA, ConOpt, and CGD, with $\gamma = 1.0$, $\epsilon = 10^{-6}$ and let the stepsizes range over $\eta \in \{0.005, 0.025, 0.1, 0.4\}$. We begin with the deterministic case $\hat{\Sigma} = \Sigma$, corresponding to the limit of large sample size. We let $d \in \{20, 40, 60\}$ and evaluate the algorithms according to the trade-off between the number of forward evaluations and the corresponding reduction of the residual $\|W + W^\top\|_{\text{FRO}}/2 + \|UU^\top - VV^\top\|_{\text{FRO}}$, starting with a random initial guess (the same for all algorithms) obtained as $W_1 = \delta W$, $V_1 = U + \delta V$, where the entries of $\delta W, \delta V$ are i.i.d uniformly distributed in $[-0.5, 0.5]$. We count the number of "forward passes" per outer iteration as follows.

- OGDA: 2
- SGA: 4
- ConOpt: 6
- CGD: 4 + 2 $*$ number of CG iterations

The results are summarized in Figure .13. We see consistently that for the same stepsize, CGD has convergence rate comparable to that of OGDA. However, as we increase the stepsize, the other methods start diverging, thus allowing CGD to achieve significantly better convergence rates by using larger stepsizes. For larger dimensions ($d \in \{40, 60\}$), OGDA, SGA, and ConOpt become even more unstable such that OGDA with the smallest stepsize is the only other method that still converges, although at a much slower rate than CGD with larger stepsizes. We

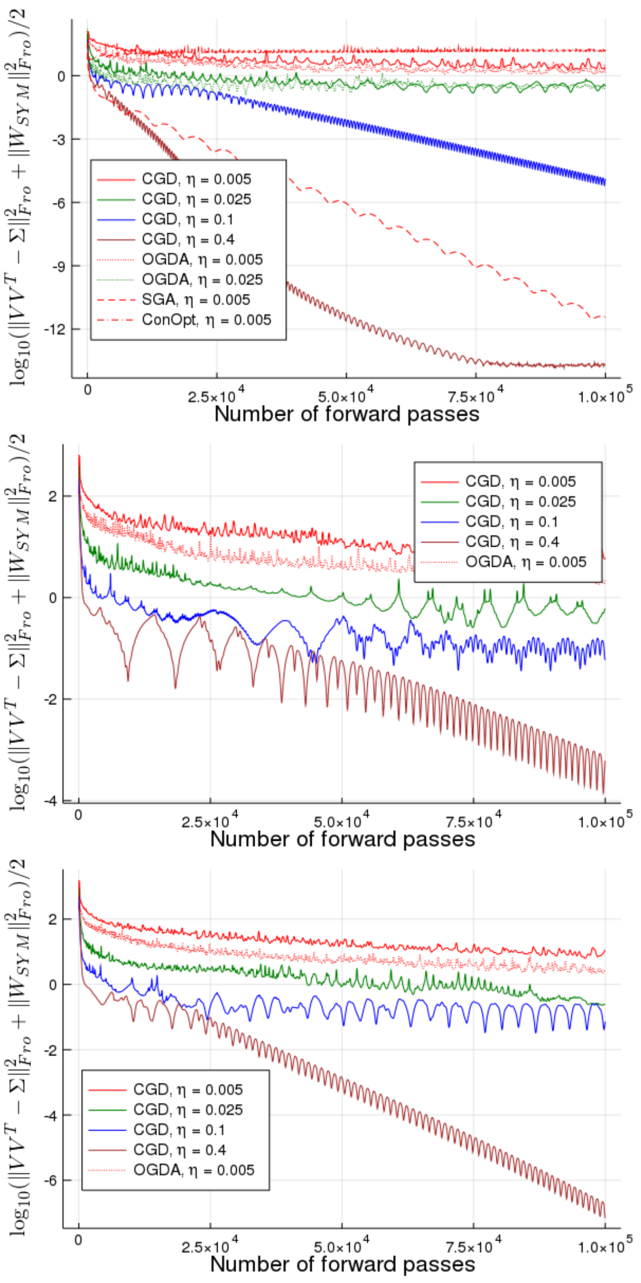


Figure .13: **Comparison of convergence speed.** The decay of the residual as a function of the number of forward iterations ($d = 20, 40, 60$, from top to bottom). **Note that missing combinations of algorithms and stepsizes correspond to divergent experiments**. While the exact behavior of the different methods is subject to some stochasticity, results as above were typical during our experiments.

now consider the stochastic setting, where at each iteration a new $\hat{\Sigma}$ is obtained as the empirical covariance matrix of N samples of $\mathcal{N}(0, \Sigma)$, for $N \in \{100, 1000, 10000\}$. In this setting, the stochastic noise very quickly dominates the error, preventing CGD from achieving significantly better approximations than the other algorithms, while other algorihtms decrease the error more rapidly, initially. It might be possible to improve the performance of our algorithm by lowering the accuracy of the inner linea system solve, following the intuition that in a noisy environment, a very accurate solve is not worth the cost. However, even without tweaking ϵ, it is noticeable that the trajectories of CGD are less noisy than those of the other algorithms, and it is furthermore the only algorithm that does not diverge for any of the stepsizes. It is interesting to note that the trajectories of CGD are consistently more regular than those of the other algorithms, for comparable stepsizes.

Experiment: Fitting a bimodal distribution

We use a GAN to fit a Gaussian mixture of two Gaussian random variables with means $\mu_1 = (0, 1)^\top$ and $\mu_2 = (2^{-1/2}, 2^{-1/2})^\top$, and standard deviation $\sigma = 0.1$. Generator and discriminator are represented by dense neural nets with four hidden layers of 128 units each that are initialized as orthonormal matrices, and ReLU as nonlinearities after each hidden layer. The generator uses 512-variate standard Gaussian noise as input, and both networks use a linear projection as their final layer. At each step, the discriminator is shown 256 real and 256 fake examples. We interpret the output of the discriminator as a logit and use sigmoidal crossentropy as a loss function. We tried stepsizes $\eta \in \{0.4, 0.1, 0.025, 0.005\}$ together with RMSProp ($\rho = 0.9$) and applied SGA, ConOpt ($\gamma = 1.0$), OGDA, and CGD. Note that the RMSProp version of CGD with diagonal scaling given by the matrices S_x, S_y is obtained by replacing the quadratic penalties $x^\top x/(2\eta)$ and $y^\top y/(2\eta)$ in the local game by $x^\top S_x^{-1} x/(2\eta)$ and $y^\top S_x^{-1} y/(2\eta)$, and carrying out the remaining derivation as before. This also allows to apply other adaptive methods like Adam. On all methods, the generator and discriminator are initially chasing each other across the strategy space, producing the typical cycling pattern. When using SGA, ConOpt, or OGDA, however, eventually the algorithm diverges with the generator either mapping all the mass far away from the mode, or collapsing the generating map to become zero. Therefore, we also tried decreasing the stepsize to 0.001, which however did not prevent the divergence. For CGD, after some initial cycles, the generator starts splitting the mass and distributes is roughly evenly among the two modes. During our experiments, this configuration appeared to be robust.

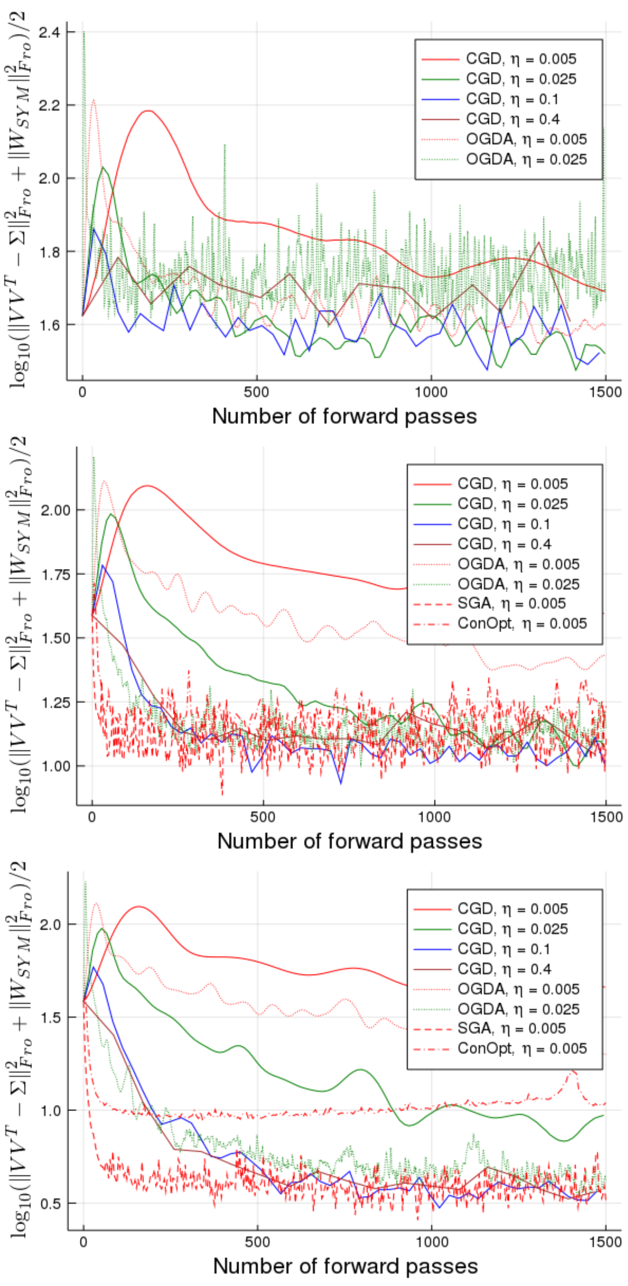


Figure .14: **Convergence speed in the stochastic case.** The decay of the residual as a function of the number of forward iterations in the stochastic case with $d = 20$ and batch sizes of 100, 1000, 10000, from top to bottom).

.5 Appendix to Chapter 8

.5.1 Simplifying constraints by duality

Constrained competitive optimization. The most general class of problems that we are concerned with is of the form

$$\min_{\substack{x \in \mathcal{C},\\ \tilde{f}(x) \in \tilde{\mathcal{C}}}} f(x, y), \quad \min_{\substack{y \in \mathcal{G},\\ \tilde{g}(y) \in \tilde{\mathcal{K}}}} g(x, y), \tag{85}$$

where $\mathcal{C} \subset \mathbb{R}^m, \mathcal{G} \in \mathbb{R}^n$ are convex sets, $\tilde{f} : \mathcal{C} \longrightarrow \mathbb{R}^{\tilde{n}}$ and $\tilde{g} : \mathcal{G} \longrightarrow \mathbb{R}^{\tilde{m}}$ are continuous and piecewise differentiable multivariate functions of the two agents' decision variables x and y, and $\tilde{\mathcal{C}}, \tilde{\mathcal{K}}$ are closed convex cones. This framework is extremely general and by choosing suitable functions $\tilde{f}, \tilde{g}$ and convex cones $\tilde{\mathcal{C}}, \tilde{\mathcal{K}}$, it can implement a variety of nonlinear equality, inequality, and positive-definiteness constraints. While there are many ways in which a problem can be cast into the above form, we are interested in the case where the $f, \tilde{f}, g, \tilde{g}$ are allowed to be *complicated*, for instance given in terms of neural networks, while the $\tilde{\mathcal{C}}, \tilde{\mathcal{K}}$ are simple and well-understood. For convex constraints and objectives f and g, the canonical solution concept is a *Nash equilibrium*.

Definition 21. *A* Nash equilibrium *of Problem 85 is a pair of feasible strategies $(\bar{x}, \bar{y})$ such that $\bar{x}$ $(\bar{y})$ is the optimal strategy for x (y) given $y = \bar{y}$ $(x = \bar{x})$.*

In the non-convex case, it is less clear what should constitute a solution and in Chapter 9 we will argue that meaningful solutions need not even be local Nash equilibria.

Lagrange multipliers for linear constraints. Using the classical technique of Lagrangian duality, the complicated parameterization $f, \tilde{f}, g, \tilde{g}$ and the simple constraints given by the $\tilde{\mathcal{C}}, \tilde{\mathcal{K}}$ can be further decoupled. The polar of a convex cone $\mathcal{G}$ is defined as $\mathcal{G}^{\circ} := \left\{ y : \sup_{x \in \mathcal{G}} x^{\top} y \leq 0 \right\}$. Using this definition, we can rewrite Problem (85) as

$$\min_{\substack{x \in \mathcal{C},\\ \mu \in \tilde{\mathcal{K}}^{\circ}}} f(x, y) + \max_{\nu \in \tilde{\mathcal{C}}^{\circ}} \nu^{\top} \tilde{f}(x), \quad \min_{\substack{y \in \mathcal{G},\\ \nu \in \tilde{\mathcal{C}}^{\circ}}} g(x, y) + \max_{\mu \in \tilde{\mathcal{K}}^{\circ}} \mu^{\top} g(y). \tag{86}$$

Here we used the fact that the maxima are infinity if any constraint is violated, and zero otherwise.

Watchmen watching watchmen. We can now attempt to simplify the problem by making μ_j (ν_i) decision variables of the y (x) player and adding a zero sum objective to the game that incentivizes both players to enforce each other's compliance with the constraints, resulting in

$$\min_{\substack{x \in \mathcal{C}, \\ \mu \in \tilde{\mathcal{K}}^\circ}} f(x,y) + \nu^\top \tilde{f}(x) - \mu^\top \tilde{g}(y), \quad \min_{\substack{y \in \mathcal{G}, \\ \nu \in \mathcal{C}^\circ}} g(x,y) + \mu^\top \tilde{g}(y) - \nu^\top \tilde{f}(x). \tag{87}$$

Is there a relationship between the Nash equilibria of Problems (85) and (87)? It turns out that this question can be studied in terms of two *decoupled* zero-sum games.

Lemma 19. *A pair of points $\bar{x}$, $\bar{y}$ is a Nash equilibrium of Problem 85 if and only if $\bar{x}$ and $\bar{y}$ are minimizers of*

$$\min_{\substack{x \in \mathcal{C}, \\ \tilde{f}(x) \in \tilde{\mathcal{C}}}} F(x), \quad \min_{\substack{y \in \mathcal{G}, \\ \tilde{g}(y) \in \tilde{\mathcal{K}}}} G(y), \tag{88}$$

for $F(x) \coloneqq f(x, \bar{y})$ and $G(y) \coloneqq g(\bar{x}, y)$. Similarly, a pair of strategies $(\bar{x}, \bar{\mu})$, $(\bar{y}, \bar{\nu})$ is a Nash equilibrium of Problem 87 if and only if $(\bar{x}, \bar{\nu})$, $(\bar{y}, \bar{\mu})$ are Nash equilibria of the decoupled zero sum games

$$\min_{x \in \mathcal{C}} \max_{\nu \in \tilde{\mathcal{C}}^\circ} F(x) + \nu^\top \tilde{f}(x), \quad \min_{y \in \mathcal{G}} \max_{\mu \in \tilde{\mathcal{K}}^\circ} G(y) + \mu^\top \tilde{g}(y). \tag{89}$$

Proof. The first part of the Lemma follows directly from the Definition 21 of a Nash equilibrium. For the second part, we observe that $(\bar{x}, \bar{\mu})$ ($(\bar{y}, \bar{\nu})$) is an optimal strategy against $(\bar{y}, \bar{\nu})$ ($(\bar{x}, \bar{\mu})$) if and only if $\bar{x}$ ($\bar{y}$) minimizes F (G) over $\mathcal{C}$ ($\mathcal{G}$) and $\bar{\mu}$ ($\bar{\nu}$) maximizes $\mu \mapsto \mu^\top g(\bar{y})$ ($\nu \mapsto \nu^\top f(\bar{x})$). But this is exactly the definition of $(\bar{x}, \bar{\nu})$ ($(\bar{y}, \bar{\mu})$) being a Nash equilibrium of Problem 89. $\square$

While this result follows directly from the definition, it reduces the question to that of constrained single-agent optimization, which has been studied extensively, allowing us to deduce the following theorem. For convex, strictly feasible problems (*"Slater's condition"*), we can show the equivalence of Problems (85) and (87). In order to formulate these results in full generality, we need the following definition.

Definition 22. *We call a function $f : \mathcal{G} \longrightarrow \mathbb{R}^n$* convex with respect to the cone *$\mathcal{C} \subset \mathbb{R}^n$ if*

$$\tau f(x) + (1-\tau) f(y) - f(\tau x + (1-\tau) y) \in \mathcal{C}, \ \forall \tau \in [0,1], \ x, y \in \mathcal{G}.$$

With this definition, we can formulate the following theorem:

Theorem 34. *Assume that the following holds:*

(i): f, $\tilde{f}$, g, and $\tilde{g}$ are continuous.

(ii): $\tilde{f}$ ($\tilde{g}$) is convex with respect to $-\tilde{\mathcal{C}}$ ($-\tilde{\mathcal{K}}$).

(iii): For all $\bar{y} \in \mathcal{G}$ ($\bar{x} \in \mathcal{C}$), F (G) as defined in Lemma 19 is convex.

(iv): For all $\bar{x} \in \mathcal{C}$ and $\bar{y} \in \mathcal{G}$, the minimal values of Problem 88 are finite (not $-\infty$).

(v): There exist (x, y) such that $x \in \operatorname{int} \mathcal{C}$, $\tilde{f}(x) \in \operatorname{int} \tilde{\mathcal{C}}$, $y \in \operatorname{int} \mathcal{G}$, $\tilde{g}(y) \in \operatorname{int} \tilde{\mathcal{K}}$.

Then, $\bar{x}$ and $\bar{y}$ are a Nash equilibrium of Problem (85) if and only if there exist $\bar{\nu}$ and $\bar{\mu}$ such that $(\bar{x}, \bar{\mu})$ and $(\bar{y}, \bar{\nu})$ are a Nash equilibrium of Problem (87).

Proof. By Lemma 19, it is enough to show that $\bar{x}$ and $\bar{y}$ are minimizers of Problem 88 if and only if they can be complemented with Lagrange multipliers $\bar{\nu}$ and $\bar{\mu}$ to obtain Nash equilibria of Problem 89. This result is shown, for instance, in [73, Chapter 3, Theorem 5.1]. □

A simplified problem. In the general non-convex setting, the relationship between Problems (85) and (87) is difficult to characterize. In this case, Problem (87) serves as an approximation to Problem 85 that might be easier to solve. Techniques for closing the duality gap in single agent optimization, such as the addition of redundant constraints, can also serve to improve the approximation of Problem (85) by Problem 87.

.5.2 Numerical implementation and experiments

Dual coordinate system for improved stability. In principle, either the primal, or the dual coordinate system can be used to keep track of the running iterate. However, we observe that storing the iterates in CGD in the dual coordinate system improves the numerical stability of the algorithm.

When expressing the update direction in the dual coordinate system, the local problem reads

$$\min_{x^* \in \mathbb{R}^m} [D_x f] \left[D^2\psi\right]^{-1} x^* + x^{*,\top} \left[D^2\psi\right]^{-1} \left[D^2_{xy} f\right] \left[D^2\phi\right]^{-1} y^* + \frac{1}{2} x^{*,\top} \left[D^2\psi\right]^{-1} x^*$$
$$\min_{y^* \in \mathbb{R}^n} \left[D_y g\right] \left[D^2\phi\right]^{-1} y^* + x^{*,\top} \left[D^2\psi\right]^{-1} \left[D^2_{xy} g\right] \left[D^2\phi\right]^{-1} y^* + \frac{1}{2} y^{*,\top} \left[D^2\phi\right]^{-1} y^*,$$

where all derivatives are computed in the last iterate (x_k, y_k) Setting the derivatives with respect to $x*$ (y^*) to zero, we obtain

$$[D_x f] \left[D^2\psi\right]^{-1} + \left(\left[D^2\psi\right]^{-1} \left[D^2_{xy} f\right] \left[D^2\phi\right]^{-1} y^*\right)^\top + x^{*,\top} \left[D^2\psi\right]^{-1} = 0 \quad (90)$$
$$\left[D_y g\right] \left[D^2\phi\right]^{-1} + x^{*,\top} \left[D^2\psi\right]^{-1} \left[D^2_{xy} g\right] \left[D^2\phi\right]^{-1} + y^{*,\top} \left[D^2\phi\right]^{-1} = 0. \quad (91)$$

We plug these equations into each other to obtain

$$[D_x f] \left[D^2\psi\right]^{-1} - \left(\left[D_y g\right] + x^{*,\top} \left[D^2\psi\right]^{-1} \left[D^2_{xy} g\right]\right) \left[D^2\phi\right]^{-1} \left[D^2_{yx} f\right] \left[D^2\psi\right]^{-1} + x^{*,\top} \left[D^2\psi\right]^{-1} = 0$$
$$\left[D_y g\right] \left[D^2\phi\right]^{-1} - \left([D_x f] + y^{*,\top} \left[D^2\phi\right]^{-1} \left[D^2_{yx} f\right]\right) \left[D^2\psi\right]^{-1} \left[D^2_{xy} g\right] \left[D^2\phi\right]^{-1} + y^{*,\top} \left[D^2\phi\right]^{-1} = 0.$$

Solving the above for x^* and y^*, we obtain

$$x^* = -\left(\left[D^2\psi\right]^{-1} - \left[D^2\psi\right]^{-1} \left[D^2_{xy} f\right] \left[D^2\phi\right]^{-1} \left[D^2_{yx} g\right] \left[D^2\psi\right]^{-1}\right)^{-1} \left(\left[D^2\psi\right]^{-1} [D_x f]^\top - \left[D^2\psi\right]^{-1} \left[D^2_{xy} f\right] \left[D^2\phi\right]^{-1} \left[D_y g\right]^\top\right)$$
$$y^* = -\left(\left[D^2\phi\right]^{-1} - \left[D^2\phi\right]^{-1} \left[D^2_{yx} g\right] \left[D^2\psi\right]^{-1} \left[D^2_{xy} f\right] \left[D^2\phi\right]^{-1}\right)^{-1} \left(\left[D^2\phi\right]^{-1} [D_x g]^\top - \left[D^2\phi\right]^{-1} \left[D^2_{yx} g\right] \left[D^2\psi\right]^{-1} [D_x f]^\top\right).$$

Once x^* and y^* have been computed, we can update the dual variables as $x_{k+1} = x_k + x^*$ and $y_{k+1} = y_k + y^*$.

Computing the updates in practice. Just like CGD, CMD requires the solution of a system of linear equations at each step. While this may seem prohibitively expensive at first, we will show in Chapter 9 that CGD can be scaled to problems with millions of degrees of freedom. This is achieved by using Krylov subspace methods such as the conjugate gradient or GMRES algorithms [210] combined with mixed-mode automatic differentiation that allows to compute Hessian-vector products almost as cheaply as gradients [195]. By using $\left[D^2\psi\right]$ and $\left[D^2\phi\right]$ as a preconditioner, the matrices that have to be inverted at each step are perturbations of the identity

$$\left(\mathrm{Id} - \left[D^2\psi\right]^{-\frac{1}{2}} \left[D^2_{xy} f\right] \left[D^2\phi\right]^{-1} \left[D^2_{yx} g\right] \left[D^2\psi\right]^{-\frac{1}{2}}\right) \quad (92)$$
$$\left(\mathrm{Id} - \left[D^2\phi\right]^{-\frac{1}{2}} \left[D^2_{yx} g\right] \left[D^2\psi\right]^{-1} \left[D^2_{xy} f\right] \left[D^2\phi\right]^{-\frac{1}{2}}\right). \quad (93)$$

As discusssed in Chapter 7, competing methods become unstable if the perturbation is large. If the perturbation is small, conjugate gradient or GMRES algorithms converge quickly, resulting in minimal overhead. In Chapter 7, we showed that this adaptivity, together with the fact that CGD can use larger step sizes, allows it to outperform competing methods even when fairly accounting for the cost of the matrix inversion. The computational cost can be reduced further by using the solution of the linear system in the last iteration as a warm start for the solution in the present iteration. Finally, we can see from Equations 90 and 91 that once x^* (y^*) has been computed, y^* (x^*) can be computed by only inverting $\left[D^2\phi\right]$ ($\left[D^2\psi\right]$), which can often be done in closed form. Therefore we can alternatingly invert the matrices in Equations 92 and 93, one at each iteration.

Numerical experiments. We will now provide numerical evidence for the practical performance of CMD. The Julia-code for the numerical experiments presented here can be found under https://github.com/f-t-s/CMD. As discussed in Section 3 of the paper, naively combining cgd with a projection step can result in convergence to spurious stable points even in convex problems, due to the *empty threats phenomenon*. The same argument applies to all other methods described in Section 7.3 as including a *"competitive term,"* with the exception of OGDA and the closely related extragradient method. We believe that the empty threats phenomenon rules out projected versions of algorithms affected by it and therefore focus our numerical comparison on the projected extragradient method (PX) of [143]. We also focus on the special case of CMW in this section, leaving a more thorough exploration of other constraint sets and Bregman potentials to future work.

A first benefit of CMD is that it allows us to extend the robustness properties of CGD to conically constrained problems. As discussed in Chapter 7, CGD is robust to strong interactions, without adjusting its step size.

To showcase this property, we consider the simple bilinear zero-sum game $f(x, y) = \alpha(x - 0.1)(y - 0.1) = -g(x, y)$ with x and y constrained to lie in $\mathbb{R}_+$. While the projected extragradient method converges faster for small α than CMD, we observe that the latter converges over the entire range of values for α, whereas the extragradient method diverges as α gets too large (see Figure .15).

.5.3 Additional experimental results on the prisoner's dilemma

In Figures .16, .17, .18, .19 we display the convergence behavior of CMD, extramirror, PCGD, and projected extragradient on the prisoner's dilemma.

.6 Appendix to Chapter 9

.6.1 Euclidean distance on images

In Figure .20, we provide a larger reproduction of Figure 2 from the main chapter. We see that also on the larger resolution, the third pair of images is visually indistinguishable, despite having the largest Euclidean distance of all pairs. The textures of this image are very rough, with a rapid alternation of bright and dark pixels. Therefore, a slight warping of the image will lead to dark pixels taking the place of bright ones and vice versa, leading to a large Euclidean distance. A similar effect could be achieved by the wind slightly moving the foliage between, for instance, successive frames of a video. Thus, this phenomenon could be observed in real images.

.6.2 ICR as projection

For the experiments in Figure 5 of the main paper, we used two tiny neural networks with 28 parameters and three layers each as generator and discriminator.

The generator $\mathcal{G}$ is composed as follows:

1. Use first four parameters as input, apply arctan nonlinearity.

2. Apply four times four dense layer, followed by arctan.

3. Apply two times four dense layer, followed by the nonlinearity

$$\begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} \exp\left(\arctan\left(y\right)/\pi + x\right) \\ \exp\left(\arctan\left(y\right)/\pi - x\right) \end{pmatrix}.$$

The form of the last nonlinearity ensures that the output is restricted to the set

$$\mathcal{S} := \left\{ (e^{s+t}, e^{s-t}) \middle| s \in \left[-\frac{1}{2}, \frac{1}{2}\right], t \in \mathbb{R} \right\} \subset \mathbb{R}^2$$

that does not include the target $P_{\text{data}} := (2, 2)$. Note that in this simple example, the generator does not take any input, but directly maps the weights $w_{\mathcal{G}} \in \mathbb{R}^{28}$ to a pair of real numbers.

The discriminator $\mathcal{D}_\eta$ is composed as follows:

1. Rescale input by the diagonal matrix η, apply four times two dense layer, followed by arctan.

2. Apply four times four dense layer, followed by arctan.

3. Apply one times four dense layer, followed by arctan.

While we did not observe the metastable projection behavior on all runs, we observed it in 13 out of 20 independent runs when using SimGD. When using CGD, we observed the projection behavior in 17 out of 20 independent runs (with the same initialization as in the SimGD cases). Furthermore, the number of iterations spent in the projection states was larger when using CGD.

.6.3 ICR on MNIST

In our experiments on MNIST, we use the network architectures detailed in Table .1 and Table .2. We train using stochastic SimGD with a learning rate of 0.01. First, we train the GAN for 9,000 iterations with a batch size of 128. We refer to the resulting generator and discriminator as the checkpoint generator and discriminator.

Details about Figure 4 of the main paper

We then create a test set that has a real set X_{real} that has 500 images sampled from MNIST training set, and a fake set X_{fake} that has 500 images generated by the checkpoint generator, as illustrated in Figure .21.

Let D_t, D_c denote, respectively, the discriminator at time step t and the checkpoint discriminator. The Euclidean distance between predictions of D_t and D_c over set X_{real} and X_{fake} in Figure 4 of the main paper is given by

$$\boldsymbol{d}_{\text{set}}(D_t, D_c) = \sqrt{\sum_{x \in X_{\text{set}}} (D_t(x) - D_c(x))^2} \tag{94}$$

where set $\in \{\text{real}, \text{fake}\}$.

Module	Kernel	Stride	Output shape
Gaussian distribution	N/A	N/A	96
Linear, BN, ReLU	N/A	N/A	1024
Linear, BN, ReLU	N/A	N/A	$128 \times 7 \times 7$
ConvT2d, BN, ReLU	4×4	2	$64 \times 14 \times 14$
ConvT2d, Tanh	4×4	2	$1 \times 28 \times 28$

Table .1: Generator architecture for MNIST experiments.

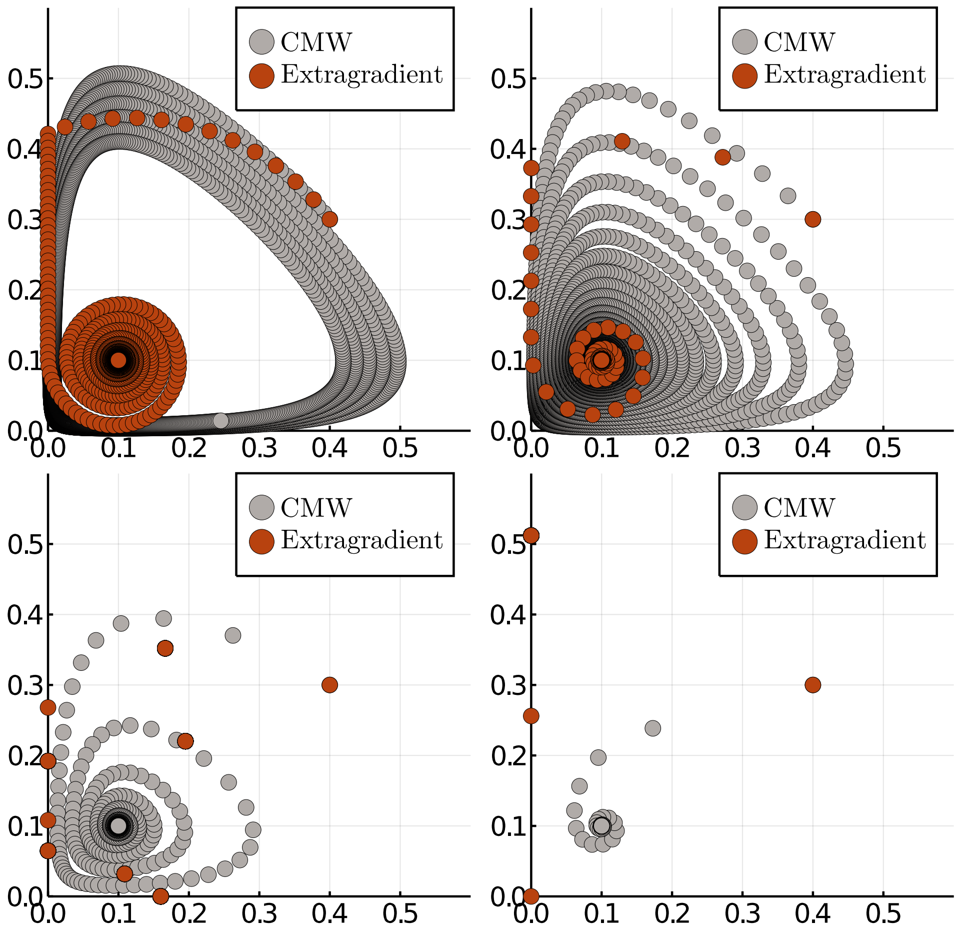


Figure .15: CMW and PX applied to $f(x, y) = \alpha(x - 0.1)(y - 0.1) = -g(x, y)$ ($\alpha \in \{0.1, 0.3, 0.9, 2.7\}$). For small α, PX converges faster, but for large α, it diverges.

Module	Kernel	Stride	Output shape
Input	N/A	N/A	$1 \times 28 \times 28$
Conv2d, LeakyReLU	5×5	1	$32 \times 24 \times 24$
MaxPool	2×2	N/A	$32 \times 12 \times 12$
Conv2d, LeakyReLu	5×5	1	$64 \times 8 \times 8$
MaxPool	2×2	N/A	$64 \times 4 \times 4$
Linear, LeakyReLU	N/A	N/A	1024
Linear	N/A	N/A	1

Table .2: Discriminator architecture for MNIST experiments.

.6.4 CIFAR10 experiments

Architecture

For our experiments on CIFAR10, we use the same DCGAN network architecture as in Wasserstein GAN with gradient penalty [106], which is reported in Table .4 and Table .5.

Hyperparameters

We compare the stability and performance of Adam and ACGD by varying the loss functions and regularization methods.

Loss:

1. Original GAN loss [96]

$$\mathcal{L}_o = \mathbb{E}_{x \sim P_{\text{data}}} \left[\log \mathcal{D}(x)\right] + \mathbb{E}_{x \sim P_{\mathcal{G}}} \left[\log\left(1 - \mathcal{D}(x)\right)\right].$$

2. Wasserstein GAN loss [16]

$$\mathcal{L}_w = \mathbb{E}_{x \sim P_{\text{data}}} \left[\mathcal{D}(x)\right] - \mathbb{E}_{x \sim P_{\mathcal{G}}} \left[\mathcal{D}(x)\right].$$

Regularization:

1. L_2 weight penalty on the discriminator parameters $\lambda \in \{10^{-2}, 10^{-3}, 10^{-4}\}$.

2. Gradient penalty on the discriminator proposed by WGAN-GP paper [106].

3. Spectral normalization on the discriminator proposed by SNGAN paper [176].

Each experiment is trained with a batch size of 64. When using Adam and the original GAN loss, we adopt the log-trick as recommended in GAN paper [96]. When using ACGD, the generator and discriminator share the same loss function. For the training of WGAN-GP, we use the same training strategy and hyperparameters as WGAN-GP [106]. Hyperparameter setting for each experiment is reported in Table .3.

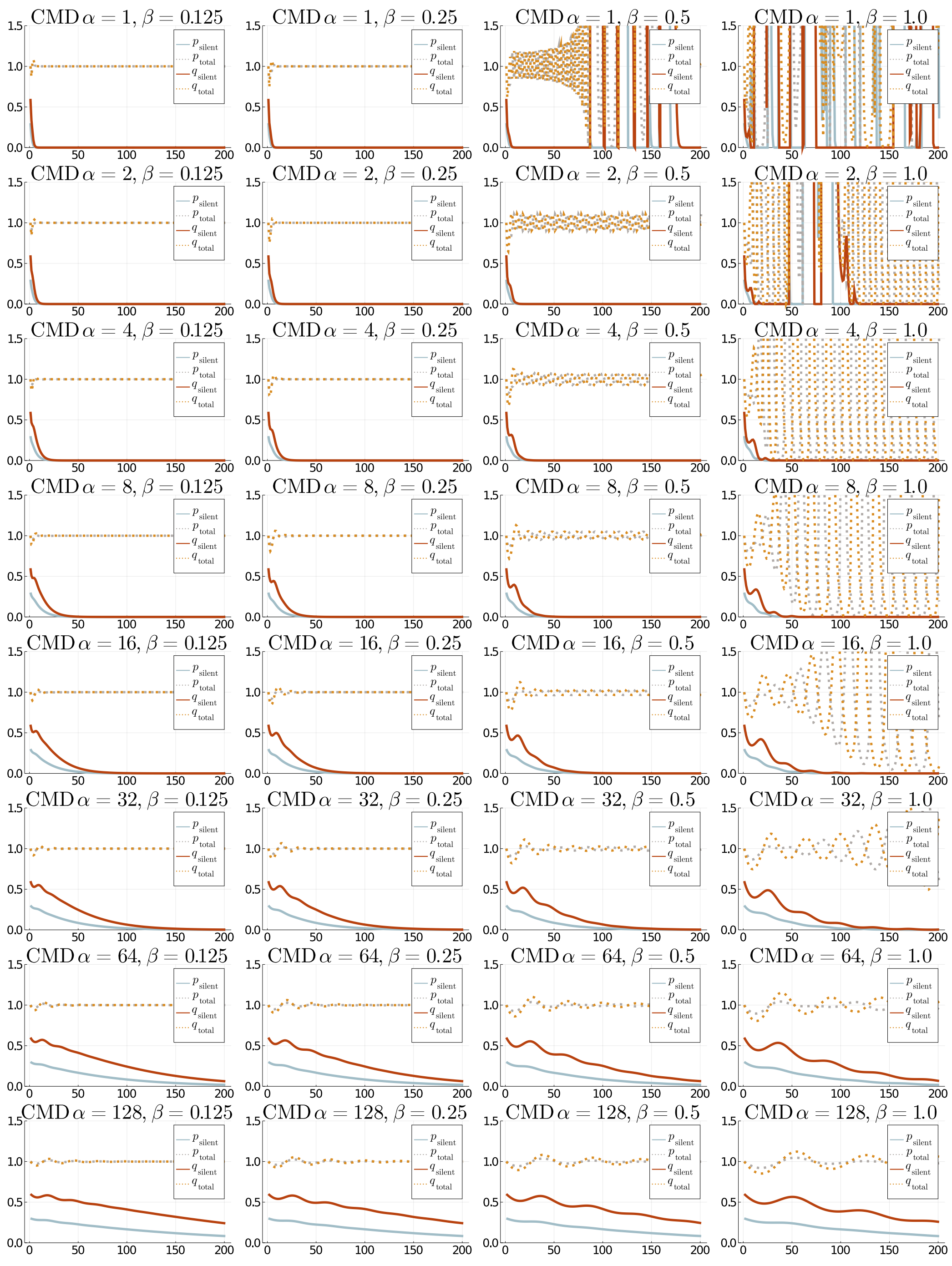


Figure .16: **Convergence of CMD on the prisoner's dilemma.**

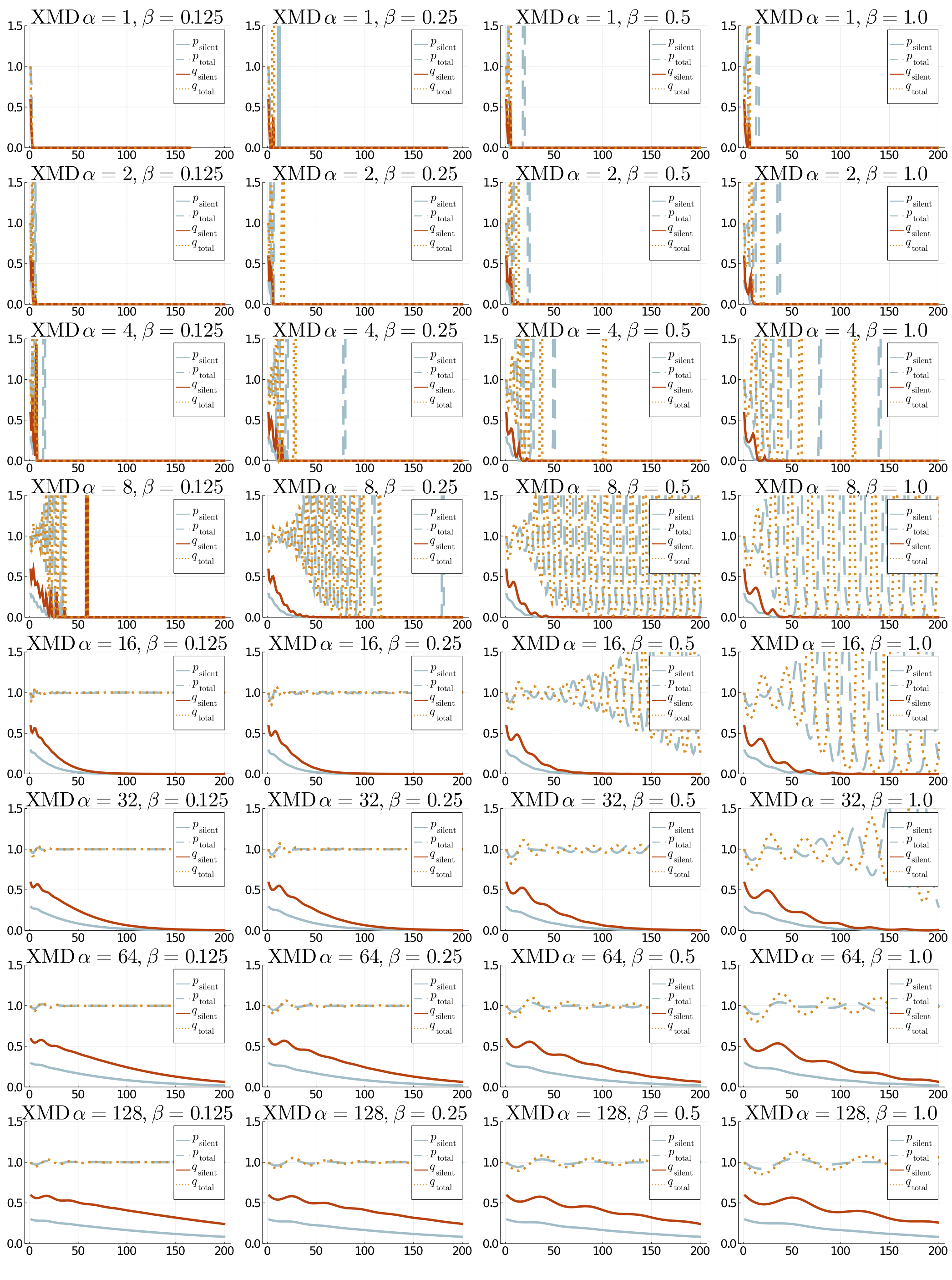


Figure .17: **Convergence of extramirror on the prisoner's dilemma.**

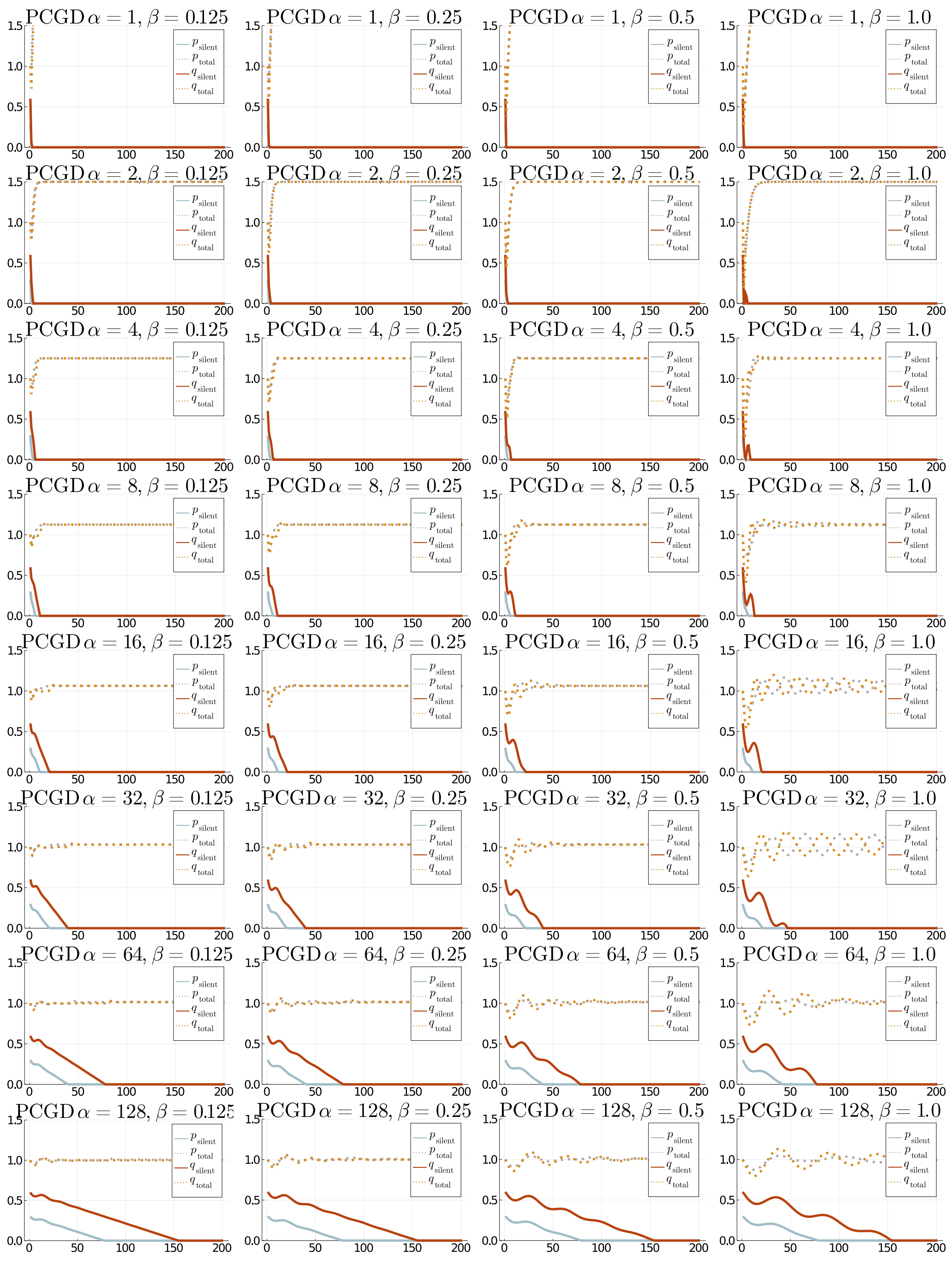


Figure .18: **Convergence of PCGD on the prisoner's dilemma.**

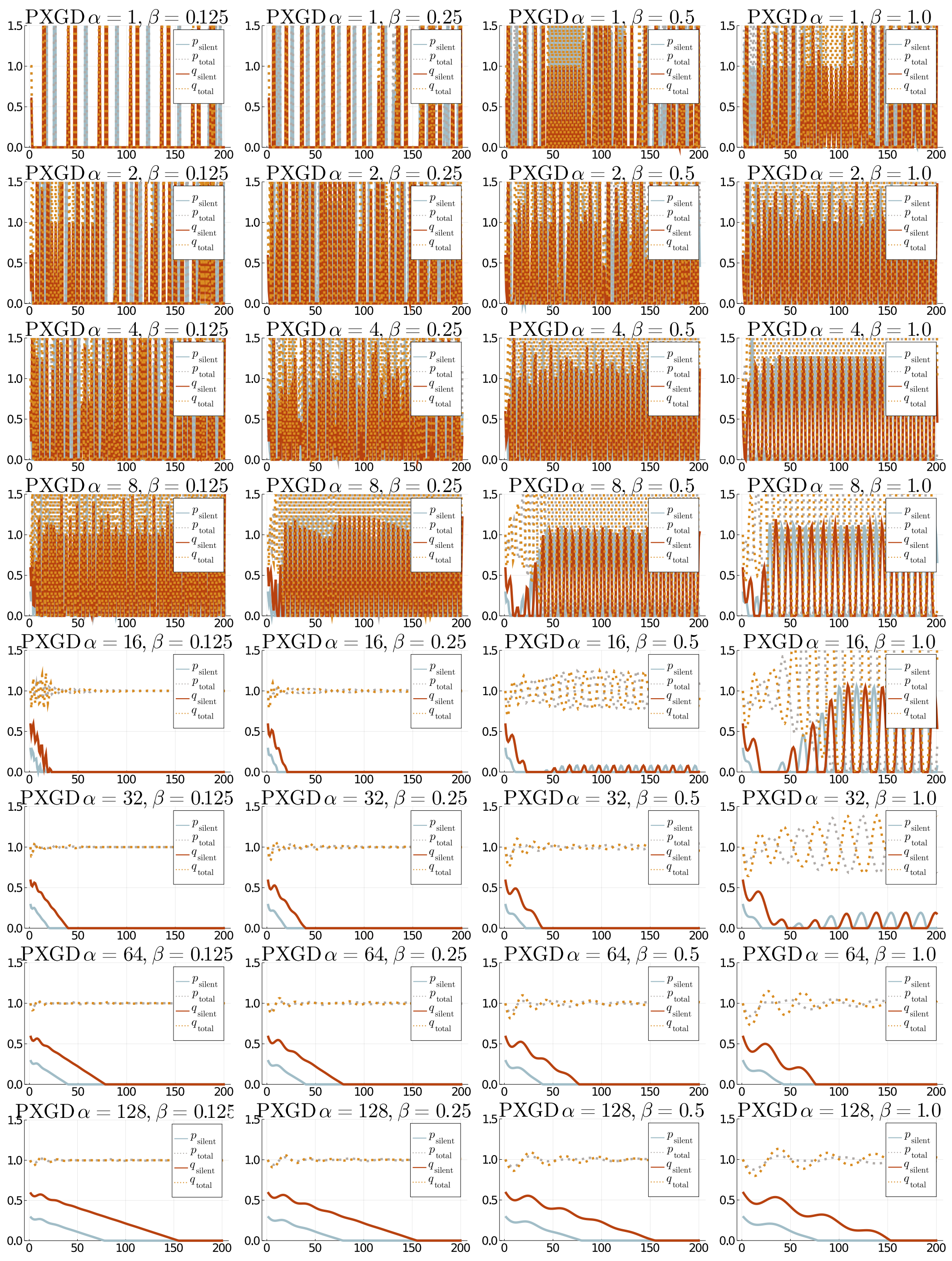


Figure .19: **Convergence of projected extragradient on the prisoner's dilemma.**

Figure .20: A larger reproduction of Figure 2 of the main paper. The first pair is based on an image by Matt Artz, the second pair on an image by Yanny Mishchuk, and the third pair on an image by Tim Mossholder. All images were obtained from https://unsplash.com/.

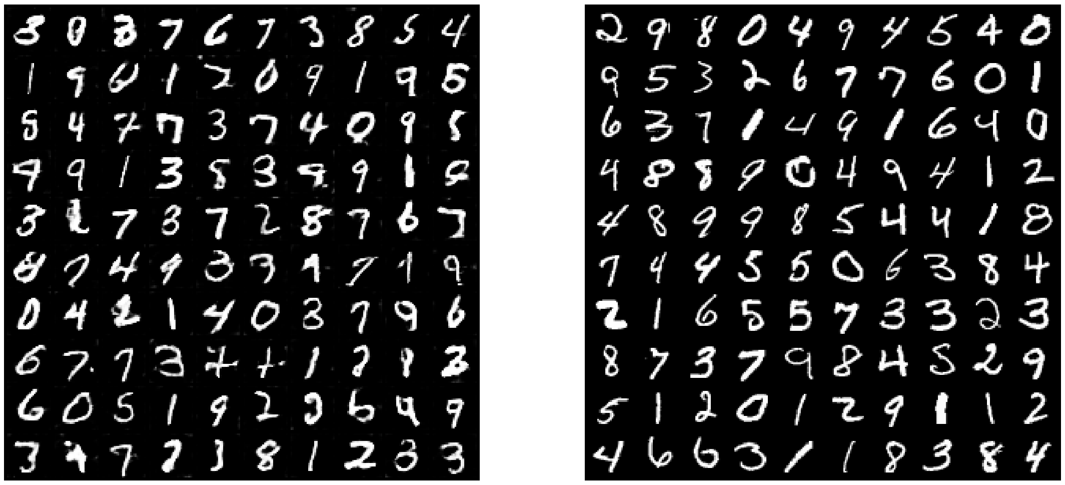


Figure .21: Test set for Figure 4 of the main paper. We show a set of fake images on the left, and real images on the right.

Experiment	Loss	Optimizer	Learning rate	Spectral Normalization	L_2 penalty	GP weight	Critic iterations
OGAN-0.0001L2+Adam	L_o	Adam	10^{-4}	N/A	10^{-4}	N/A	1
OGAN-0.0001L2+ACGD	L_o	ACGD	10^{-4}	N/A	10^{-4}	N/A	1
OGAN-NOREG+Adam	L_o	Adam	10^{-4}	N/A	N/A	N/A	1
OGAN-NOREG+ACGD	L_o	ACGD	10^{-4}	N/A	N/A	N/A	1
WGAN-GP+Adam	L_w	Adam	10^{-4}	N/A	N/A	10	5
WGAN-SN+Adam	L_w	Adam	10^{-4}	Yes	N/A	N/A	1
WGAN-NOREG+ACGD	L_w	ACGD	10^{-4}	N/A	N/A	N/A	1
WGAN-GP+ACGD	L_w	ACGD	10^{-4}	N/A	N/A	10	5
WGAN-0.01L2+Adam	L_w	Adam	10^{-4}	N/A	10^{-2}	N/A	1
WGAN-0.001L2+Adam	L_w	Adam	10^{-4}	N/A	10^{-3}	N/A	1
WGAN-0.001L2+ACGD	L_w	ACGD	10^{-4}	N/A	10^{-3}	N/A	1

Table .3: Settings for all the experiments that occurred in Figure 7 of the main paper.

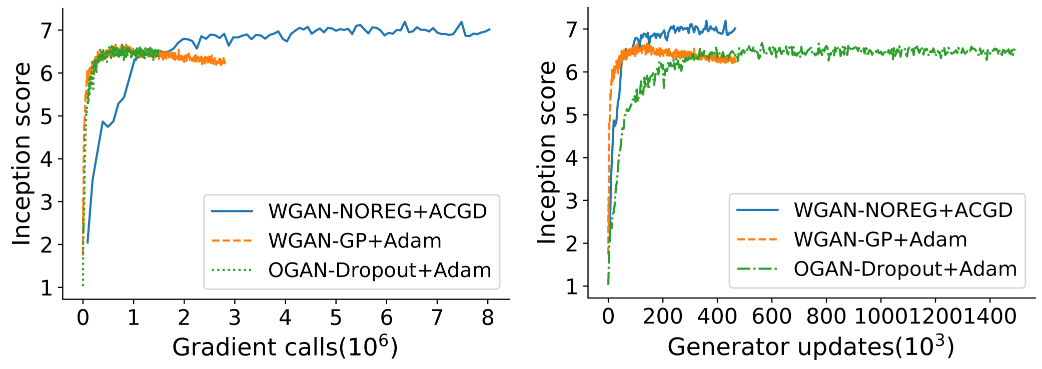


Figure .22: Tensorflow inception scores for important runs, plotted against the number of gradient calls (left) and the number of generator updates (right).

Tensorflow inception score

We compute the Tensorflow version of the inception scores for important runs of our experiments to show that the relative performance of the different models is largely the same. As reported in Figure .22, our results match the ones reported in the literature (Figure 3 in [106]) with ACGD still outperforming WGAN-GP trained with Adam by around 10%.

Module	Kernel	Stride	Output shape
Gaussian distribution	N/A	N/A	128
Linear, BN, ReLU	N/A	N/A	$256 \times 4 \times 4$
ConvT2d, BN, ReLU	4×4	2	$128 \times 8 \times 8$
ConvT2d, BN, ReLU	4×4	2	$64 \times 16 \times 16$
ConvT2d, Tanh	4×4	2	$3 \times 32 \times 32$

Table .4: Generator architecture for CIFAR10 experiments

Module	Kernel	Stride	Output shape
Input	N/A	N/A	$3 \times 32 \times 32$
Conv2d, LeakyReLU	4×4	2	$64 \times 16 \times 16$
Conv2d, LeakyReLU	4×4	2	$128 \times 8 \times 8$
Conv2d, LeakyReLu	4×4	2	$256 \times 4 \times 4$
Linear	N/A	N/A	1

Table .5: Discriminator architecture for CIFAR10 experiments

.6.5 Details on ACGD

In order to make a fair comparison with Adam, we run our experiments with ACGD, a variant of CGD that adaptively adjusts CGD's step size. The algorithm is described in Algorithm 28. ACGD computes individual step sizes for the different parameters. Let $A_{x,t}$ and $A_{y,t}$ denote the diagonal matrices containing the step sizes of x and y at time step t as elements. If $A_{x,t}$ and $A_{y,t}$ are multiples of the identity, the algorithm reduces to CGD with the corresponding step size. The reason we rearrange the terms as shown in Algorithm 28 is that we want the matrix inverse to contain an additive identity (to decrease the condition number) and be symmetric positive definite (so that we can use conjugate gradient [72] for its computation). ACGD adjusts CGD's step size adaptively during training with second moment estimate of the gradients. The update rules are derived from the local game in the same way as for CGD.

$$\begin{aligned}
&\min_x x^\top \nabla_x f(x_t, y_t) + x^\top [D_{xy} f(x_t, y_t))] y + \frac{1}{2} x^T A_{x,k}^{-1} x, \\
&\max_y y^\top \nabla_y f(x_t, y_t) + y^\top [D_{yx} f(x_t, y_t))] x - \frac{1}{2} y^T A_{y,k}^{-1} y.
\end{aligned}$$

Algorithm 28 ACGD, a variant of CGD with RMSProp-type heuristic to adjust learning rates. All operations on vectors are element wise. $D^2_{xy}f$, $D^2_{yx}f$ denote the mixed Hessian matrix $\frac{\partial^2 f}{\partial x \partial y}$ and $\frac{\partial^2 f}{\partial y \partial x}$. β_2^t denotes β_2 to the power t. $\phi(\eta)$ denotes a diagonal matrix with η on the diagonal. Hyperparameter settings for the tested GANs training problems are $\alpha = 10^{-4}, \beta_2 = 0.99$, and $\epsilon = 10^{-5}$.

```
Require: $\alpha$: Step size
Require: $\beta_2$: Exponential decay rates for the second moment estimates
Require: $\max_y \min_x f(x, y)$: zero-sum game objective function with parameters
  $x, y$
Require: $x_0, y_0$ Initial parameter vectors
  $t \leftarrow 0$ Initialize timestep
  $v_{x,0}, v_{y,0} \leftarrow 0$ (Initialize the $2^{nd}$ moment estimate)
  repeat
    $t \leftarrow t + 1$
    $v_{x,t} \leftarrow \beta_2 \cdot v_{x,t-1} + (1 - \beta_2) \cdot g^2_{x,t}$
    $v_{y,t} \leftarrow \beta_2 \cdot v_{y,t-1} + (1 - \beta_2) \cdot g^2_{y,t}$
    $v_{x,t} \leftarrow v_{x,t} / (1 - \beta_2^t)$
    $v_{y,t} \leftarrow v_{y,t} / (1 - \beta_2^t)$ (Initialization bias correction )
    $\eta_{x,t} \leftarrow \alpha / (\sqrt{v_{x,t}} + \epsilon)$
    $\eta_{y,t} \leftarrow \alpha / (\sqrt{v_{y,t}} + \epsilon)$
    $A_{x,t} = \phi(\eta_{x,t})$
    $A_{y,t} = \phi(\eta_{y,t})$
    $\Delta x_t \leftarrow -A^{\frac{1}{2}}_{x,t}(I + A^{\frac{1}{2}}_{x,t} D^2_{xy} f A_{y,t} D^2_{yx} f A^{\frac{1}{2}}_{x,t})^{-1} A^{\frac{1}{2}}_{x,t}$
    $(\nabla_x f + D^2_{xy} f A_{y,t} \nabla_y f)$
    $\Delta y_t \leftarrow A^{\frac{1}{2}}_{y,t}(I + A^{\frac{1}{2}}_{y,t} D^2_{yx} f A_{x,t} D^2_{xy} f A^{\frac{1}{2}}_{y,t})^{-1} A^{\frac{1}{2}}_{y,t}$
    $(\nabla_y f - D^2_{yx} f A_{x,t} \nabla_x f)$
    $x_t \leftarrow x_{t-1} + \Delta x_t$
    $y_t \leftarrow y_{t-1} + \Delta y_t$
  until $x_t, y_t$ converged
```