THE UNIVERSITY OF CHICAGO

THREE WAYS TO TAME STRUCTURED DATA

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

COMMITTEE ON COMPUTATIONAL AND APPLIED MATHEMATICS

BY
PHILLIP LO

CHICAGO, ILLINOIS
AUGUST 2025

S.D.G.

It is the glory of God to conceal things,

but the glory of kings is to search things out.

—*Proverbs 25:2 (NRSV)*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

This dissertation marks the end of a decade-long journey at the University of Chicago, very little of which would have been possible without the support of many mentors, collaborators, and friends.

I have been fortunate enough in graduate school to work closely with three advisors, each of whom have contributed in a unique way to my growth as an applied mathematician. Eric Jonas, you have taught me what it means to do computational science and how to write good code (among a host of many other best practices) in pursuit of it. Yuehaw Khoo, you have shown me how to draw inspiration from real world problems and rigorously mathematize and prove things about them. Aly Azeem Khan, you have introduced me to the world of computational immunology and brought me into a community of biologists, machine learning researchers, and engineers that I never expected I could have the opportunity to work with. Thank you also to Aaron Dinner for serving on my committee and offering your expertise in computational chemistry.

There are many scientific colleagues and collaborators that I am deeply thankful for. Thank you to my close collaborator Sudarshan Babu; HyDiF would not have been possible without your deep expertise in hypernetworks. Thank you to Christopher Stith for answering my differential geometry questions when my work got too abstract for my comfort zone, and thank you to Caitlin Wong Hickernell for answering my biochemistry questions when my work got too concrete. And innumerable thanks go out to all my colleagues in the CAM PhD program; how fortunate I have been to work in an office among some of the most brilliant people I know, always willing to help with questions, mathematical or not. My PhD would have been far lonelier without the company of Adela, Abby, Sue, Nathan, Angela, Mark, Justin, Richard, Hwanwoo, Solomon, *et. al.*

Thank you also to the leadership and administrative team of CAM; Mary Silber and Guillaume Bal, you have cultivated a PhD program that I am proud to have been a part of. Zellencia Harris and Jonathan Rodriguez, you have allowed my six years of graduate school

to be about science, not logistics.

I would be remiss if I did not thank the two co-directors of undergraduate studies in the Department of Mathematics. My very first college class back in Autumn of 2015 was MATH 16100 with Jitka Stehnova; this was the class that inspired me to major in mathematics and what set off my entire academic trajectory. Thank you also to John Boller for the opportunity to serve as an instructor of record as a graduate student. Within the Department of Mathematics, I am perhaps most grateful for the many undergraduates I have had the pleasure of teaching in various calculus courses over the years; the greatest moments of my professional life have come not from proving theorems or computing benchmarks, but rather seeing your faces light up when you finally understand how integration or the KKT theorem works. I hope you all take seriously my charge that anyone can do math.

Outside of mathematics, I have had the unusual privilege of having many musical mentors, colleagues, and collaborators to thank for as rich of a musical life as any amateur musician with a day job could hope for. Thank you to Barbara Schubert for your unimpeachable leadership of the Music Performance Program and the University Symphony Orchestra. Thank you to my piano teachers Eugenia Jeong, Melina Lee Masur, and Lam Wong, who have managed to help me grow as a pianist even throughout the busyness of graduate school. Thank you to the organizers of the New Music Ensemble for cultivating opportunities to work with composers from all over the world to make and hear sounds that I could have never imagined. And thank you to my many stand partners and chamber music collaborators throughout the years; it has been an absolute blast making music with you all.

Thank you to pastor Bing Nieh for your wise spiritual guidance and the rest of Christ Church Chicago for being my faith home for so many years. Thank you also to my friends in Graduate Christian Fellowship; what a blessing it is to have a consistent group of graduate students to meet with weekly and talk about anything other than graduate school.

Thank you to my parents, who were my first math teachers and have always been supportive of my academic journey. I started off with endless arithmetic worksheets that my

parents made for me. Now I am getting a PhD, also in endless arithmetic, just done with GPUs instead of by hand.

Most of all, I am profoundly grateful to have had many dear friends support me throughout my journey and fill my life with immense happiness. Thank you to Larry and Chelsea for always coming out to visit from the Bay Area so often; I promise I will return the favor more once I graduate. Thank you to John and Caitlin for spontaneous good meals and good tea. Thank you to Andrew for your acerbic wit, and Maggie for mellowing it out. Thank you to Caleb for being the ideal roommate. Thank you to Emma, Victor, Harin, Kelsey, Joseph, and the rest of the folks at UT Southwestern for giving me perspective and showing me that graduate school isn't actually so bad; medical school is far, far worse. Thank you to Kevin for being an endless fountain of fun. Thank you to Emily for being such an extraordinary friend from the start; we met on the very first day of college, our dorm rooms down the hall from each other. How serendipitous that ten years later, we still live down the street from another, and once again we are graduating at the same time from UChicago. Thank you all for making my twenties quite lovely, so joyful.

Finally, my deepest acknowledgments go to my wife, Jenny Kim; I cannot possibly thank you enough for your unwavering support. What pride I have had in you, seeing you transition from pre-med to medical school to otolaryngology residency. What inspiration I have felt in your work, hearing about the life-changing surgeries you perform in the operating room. What joys I have experienced with you, growing together ever since we first met ten years ago. How blessed I am to have you in my life!

ABSTRACT

This dissertation presents three different problems in recovering underlying structure from data in diverse scientific fields, each also elucidating a different flavor of applied mathematical approaches.

The first is a problem in manifold learning inspired by cryo-electron microscopy. We consider the problem of recovering a one-dimensional curve embedded in high dimensional space from a high noise observations centered around the curve. We prove a sample complexity lower bound to demonstrate an inherent difficulty in recovering the underlying curve from data, then provide an asymptotically optimal algorithm based on the method of moments. This work demonstrates how analytic and algebraic techniques can be applied to summary statistics to navigate the challenges of high dimensional, noisy data.

The second problem we study is an optimal transport problem on discrete spin systems. We formulate a relaxation of the Glauber dynamics on the Ising model as an optimal transport problem in discrete time and space with an exponentially large number of variables. We then devise numerical techniques for tractably solving this algorithm at the scale of over one billion variables. This work highlights numerical and algorithmic approaches in applied mathematics.

Finally, we move to the world of computational chemistry and develop a machine learning framework that simultaneously serves as a generative and foundational model for 3D conformers of small molecules. Our model is able to synthesize information across a large library of molecules and effectively learn what the manifold of plausible molecular conformers looks like, as well as learn mappings from conformers to molecular properties of interest. At the core of our method is a novel molecular representation that models conformers not as discrete point clouds but as dense fields. This work exemplifies the role of data-driven techniques in complex scientific domains.

We hope this dissertation represents the rich diversity of domains and techniques in applied mathematics.

CHAPTER 1

INTRODUCTION

Throughout the history of science, a central theme has been the search for underlying mathematical models that explain data. From the 8th to 11th centuries, Islamic astronomers fit trigonometric models to celestial observations to determine the qibla, or direction to Mecca, from any geographic location with remarkable accuracy [1]. In the early 17th century, Johannes Kepler formulated his laws of planetary motion from analysis of Tycho Brahe's astronomical observations [2]. In the 1920s, observations by Umberta D'Acona of populations of cartilaginous fish in the Adriatic sea were studied by Volterra to formulate what is now known as the Lotka-Volterra predator-prey model, a foundational model in mathematical ecology [3].

In the 21st century, data no longer take the form of thousands of manual observations made by humans and analog instruments, but rather millions or billions of measurements taken by sophisticated instruments or massive arrays of measurement devices (satellites, personal electronic devices, weather stations, etc.). In this setting, constructing tractable mathematical models that can deduce structure from an incredibly large amount of data is a critical endeavor. Coupled with this explosion in data in the last half-century has been the exponential growth in computing power available to parse through and tame it. In this work, we will consider three problems in recovering structure from data in diverse fields, each with unique theoretical and computational challenges whose solutions elucidate a different flavor of applied mathematical approaches. We term these three flavors *statistical*, *computational*, and *machine learning* approaches.

1.1 A Statistical Approach: A Problem Arising from Imaging Science

The first problem we will consider is a manifold learning problem inspired by cryo-electron microscopy (cryo-EM). First developed in 1981 [4] and the subject of the 2017 Nobel Prize in Chemistry, cryo-EM is an imaging modality for biological macromolecules and the most modern of the three primary imaging methods for proteins, the other two being nuclear magnetic resonance spectroscopy (subject of the 1944 and 1952 Nobel Prize in Physics) and X-ray crystallography (subject of the 1914 Nobel Prize in Physics and many more, including the 1962 Prize in Medicine for elucidation of the structure of DNA).

The basic principle of cryo-EM is as follows: a sample containing many copies of a single macromolecule of interest is spread into a thin sheet and flash frozen. An electron microscope "photograph" is taken of the entire sheet, resulting in 2D projections of many samples of the same molecule, each one in some random unknown rotation. The task is to deduce the 3D structure of the molecule from these 2D projections of unknown rotations. This is made significantly more challenging by the fact that the images obtained from cryo-EM often have an extremely low signal-to-noise ratio (SNR). This low SNR is the reason why thousands or millions of samples are required for cryo-EM to be successful.

From Figure 1.1, it is clear that cryo-EM is a difficult problem, and a wonder that it works at all. In Chapter 2, we are interested in a rigorous study of just how difficult the cryo-EM problem is. To make the cryo-EM problem more tractable, we study a mathematical model that captures the fundamental difficulties of the cryo-EM problem. Specifically, we consider the problem on recovering a 1D curve from high noise data. To briefly describe the problem, let $C^*(t) : [0, 1] \to \mathbb{R}^n$ be some ground truth constant speed curve in space. Suppose we observe N independent samples of the form

$$y^{(j)} = C^*(\tau^{(j)}) + \sigma\xi^{(j)}, \qquad j = 1, \dots, N,$$

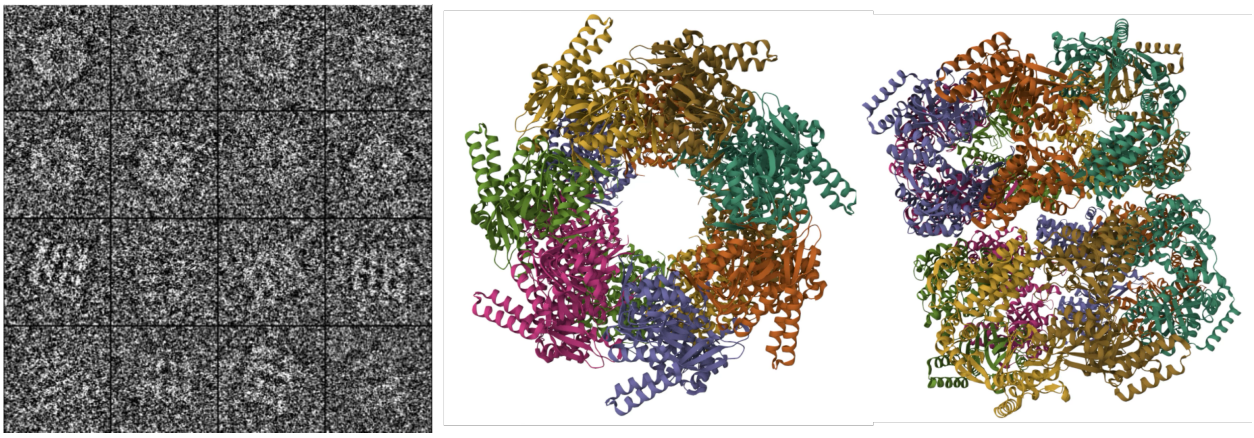

Figure 1.1: Left: sixteen cryo-EM images of the GroEL protein in various rotations (reproduced with permission from [5]); note the extremely low SNR. Center and right: two different views of the actual structure of GroEL, obtained from the Protein Data Bank [6].

where $\tau \sim U[0,1]$ is a uniform time point along the curve and $\xi \sim \mathcal{N}(0, I_d)$ is isotropic Gaussian noise; we can thus view the observations as a Gaussian point cloud centered along the curve. Our goal is to recover C^* from these observations; we call this the noisy curve recovery problem. We add on the caveat that σ is large compared to the scale of C^*; like the cryo-EM problem, we wish to study how difficult the problem is when σ is large enough to "wash out" the underlying signal C^*. We quantify the difficulty of this problem by the *sample complexity*, i.e., a measure on how many observations N are required to recover C^* from the data up to some fidelity. We show a pair of results: (a) when SNR is sufficiently low, the sample complexity scales with order *at least* $1/\text{SNR}^3$, and (b) $1/\text{SNR}^3$ samples is indeed *enough* to recover C^*, making the estimate in (a) tight.

For small SNR, $1/\text{SNR}^3$ is a very large amount of data; we consider regimes where $N \approx 10^8$. Our approach for this problem demonstrates what we call a *statistical approach* to dealing with large amounts of data, i.e., reducing the analysis of large amounts of data to the study of much more managable *summary statistics*. In particular, our analysis of this problem is based on the *method of moments*, which reduces empirical data to its moments (i.e., expected values of tensor powers of the data). We will show that miraculously, compact summary statistics of millions of extremely noisy data points are still enough to solve the

noisy curve recovery problem. In our analysis, we will reveal the terseness of our summary statistics belies some deep structure that allows us to solve the problem.

1.2 A Computational Approach: A Problem Arising from Statistical Mechanics

The second problem we study is an optimal transport problem inspired by the Ising model in statistical mechanics. One of the most celebrated models in statistical mechanics, the Ising model is a discrete model of ferromagnetism consisting of n interacting spins that can assume one of two states $\{\pm 1\}$. This leads to a configuration space $\Omega_n = \{\pm 1\}^n$ of size 2^n, which makes studying these spin configurations a computationally expensive task even for modest n. The importance of the Ising model in statistical mechanics is equalled by the importance of optimal transport in applied mathematics. In brief, optimal transport studies ways of mapping an initial probability distribution μ to a final distribution ν in as efficient a way as possible under some prescribed "moving cost". In Chapter 3, we combine these two notions to formulate an optimal transport problem over distributions on this exponentially large discrete configuration space; in short, given prescribed initial and final probability distributions μ and ν on Ω_n, rules about how these distributions can evolve in discrete time steps, and prescribed costs to these evolution steps, we wish to seek the most efficient path from μ to ν.

Nearly all existing work on optimal transport is between *continuous* probability distributions; this is in part because the *theoretical* aspects of optimal transport between discrete distributions is not particularly rich, since optimal transport reduces to a linear programming problem for discrete distributions. However, the exponential size of our configuration space gives rise to an exponentially large linear program, which poses significant *computational* challenges; in solving large scale linear programs, writing an algorithm that works in theory is easy, but running that algorithm on an actual computer is hard. Our contribution

in Chapter 3 is in presenting a tractable algorithm for exactly solving these large linear programs.

Our approach elucidates what we call a *computational approach* to dealing with large amounts of data, wherein we reformulate a mathematical problem in a way that is amenable to existing computing hardware; in the current decade, this means designing algorithms to be as parallel as possible to take advantage of the parallel nature of GPUs and associated low-level platforms for accelerated matrix computations. In particular, we solve the linear program using the primal-dual gradient method, which reduces solving linear programs to iterative matrix-vector multiplications. While matrix-vector multiplication is generally well-suited for acceleration on modern GPUs, this is not enough for our purposes, as the exponential size of our problem makes even storing these matrices in memory impossible. Rather, we go further by exploiting a particular structure in the relevant matrices that allows us to compute the necessary matrix-vector products in a parallel and *matrix-free* way that prevents us from ever actually writing down large matrices. These computational optimizations allow us to solve linear programs with over a billion variables. We parallelize our solver even further by approximating our linear program with a collection of parallel subproblems and demonstrate that this approach improves runtime by several orders of magnitude while remaining close to optimal. This work highlights an important theme in applied mathematics: seemingly intractable problems can often be made tractable by the right reformulation, with only minor sacrifices in optimality.

1.3 A Machine Learning Approach: A Problem Arising from Medicinal Chemistry

Finally, we move to the world of computational chemistry, specifically generative molecules for small, drug-like molecules. Small molecule drugs are molecules with low molecular weight (on the order of hundreds of daltons) that interact with biological processes; over 90% of

existing drugs on the market are small molecules (as opposed to biologics) [7]. The space of small molecules is astronomically vast; [8] estimates that the space of reasonable small molecule drugs is at least 10^{60}. Yet only a miniscule proportion of these drugs have ever been synthesized, indexed, or otherwise written down; the largest chemical libraries (both academic and commercial) have "only" billions or trillions of compounds [9]. Enumerating over the space of potential drug-like molecules is thus an impossible task①. Therefore, for drug discovery applications, it is critical to develop powerful tools to be able to sample diverse, novel compounds from this vast universe.

With such a rich diversity of potential drug candidates, it is important not only to be able to sample diverse compounds efficiently, but also to perform large scale *in silico* screening of compounds for key pharmacological properties. The ability to perform cheap, virtual high throughput screening for compounds before testing a much smaller number of compounds *in vitro* or *in vivo* can massively improve the efficiency of the drug discovery pipeline [7].

In this era of generative artificial intelligence, the computational medicinal chemistry community has sought out denoising diffusion models as a way to sample from the space of small molecule drugs. At a high level, denoising diffusion models train on a large corpus of data (text, images, or in our case, small molecules) to learn what the manifold of plausible data samples look like. These trained models are then able to map Gaussian noise to this manifold, giving a way to generate novel samples from randomness. Unlike with text or images, where the data itself is its own representation (text *is* a string of characters, an image *is* a pixel array), it is not as obvious how one should represent molecules in a computer-parsable way. Current approaches use a variety of different representations, including string, graph, and point cloud based representations of molecules, each with distinct advantages and disadvantages.

①. Consider a miraculous server farm on Earth that is able to evaluate whether a given small molecule is a good candidate for a particular biological application at a rate of one femtosecond per molecule. By the time the Sun has expanded enough to engulf the earth [10], only 10^{32} out of 10^{60} molecules will have been evaluated.

In Chapter 4, we propose a novel representation for 3D molecular conformers. Rather than modeling a molecule as a discrete string, graph, or point cloud, we model a molecule as a *continuously defined vector field*; we argue that such a continuous representation more richly captures the geometry of 3D conformers at high resolution. Using the fact that multilayer perceptrons (MLPs) are universal function approximators, we design a neural network architecture that is designed to map our vector field representation of molecules to MLPs approximating these vector fields. We then train a denoising diffusion model on a library of molecules which we convert to this vector field representation and find that we are successfully able to generate molecules from noise. Furthermore, our approach is able to achieve state-of-the-art performance on a *molecular inpainting* task, wherein our model is tasked with completing a partial molecule; we posit that this is a more relevant task to drug discovery, as we will argue further in Chapter 4. In addition to these *generative* capabilities, we harness the rich internal representations of molecules learned by our model to endow it with *property prediction* capabilities; we demonstrate that our model achieves state-of-the-art performance on a set of benchmark molecular property prediction tasks, making it a promising tool for virtual drug candidate screening.

The approaches in this chapter demonstrate what we call a *machine learning* approach to dealing with large amounts of data, where we take advantage of the miraculous ability of machine learning methods to model complex, high dimensional data distributions from an enormous number of samples. Central to our approach is the careful construction of a neural network architecture that is able to effectively synthesize information across millions of molecular conformers.

CHAPTER 2

METHOD OF MOMENTS FOR ESTIMATION OF NOISY CURVES

Manifold learning is a central problem in applied mathematics that seeks to find simple geometric structure in noisy data. In this chapter, we study the problem of recovering a ground truth high dimensional piecewise linear curve $C^*(t) : [0, 1] \to \mathbb{R}^d$ from a high noise Gaussian point cloud with covariance $\sigma^2 I$ centered around the curve. We first establish that the sample complexity of recovering C^* from data scales with order at least σ^6. We then show that recovery of a piecewise linear curve from the third moment is locally well-posed, and hence $O(\sigma^6)$ samples is also sufficient for recovery. We propose methods to recover a curve from data based on a fitting to the third moment tensor with a careful initialization strategy and conduct some numerical experiments verifying the ability of our methods to recover curves. This chapter is adapted from the preprint [11].

2.1 Introduction

2.1.1 Motivation and Related Work

Manifold learning is a widely-studied problem in statistics in which a low dimensional manifold is fit to high dimensional data in an effort to understand the geometry of the data and circumvent the curse of dimensionality. A vast amount of literature has been dedicated to the noise-free case, where the data lie exactly on some lower dimensional manifold; see, for instance, [12] for a review of more classical methods, or [13, 14] for more recent results. Work has also been done in the case of low noise, such as in [15–19]. Less work has been done for the more difficult high noise case, in which each data point is completely dominated by noise. In [20], the authors establish an algorithm for recovering manifolds with particular smoothness properties corrupted by Gaussian noise with covariance σ^2 for arbitrary σ, but

their algorithm requires a number of samples exponential in σ^2.

Another field in which recovering a signal from high noise has been studied is in orbit recovery problems. In these problems, an underlying signal is corrupted by some group action and a large amount of additive noise, and the goal is to recover the original signal. Perhaps the most famous such problem is the reconstruction of the 3D structure of macromolecules from cryo-electron microscopy (cryo-EM) images. In cryo-EM, a large number of extremely noisy images are taken of a molecule in various 3D orientations. We can think of these images as projections of an unknown rigid rotation of the Coulomb potential $f : \mathbb{R}^3 \to \mathbb{R}$ along an axis to a two-dimensional image, which is then corrupted by a large amount of additive Gaussian noise. The task of recovering f thus involves both denoising the Gaussian noise and undoing the rotation. See [21] for a comprehensive review of computational aspects of cryo-EM. A simpler orbit recovery problem that has been well-studied is the multireference alignment (MRA) problem. In the MRA model, the ground truth is a one-dimensional signal $f \in \mathbb{R}^d$ and the measurements consist of a random cyclic permutation of f corrupted with additive Gaussian noise. The sample complexity of MRA is established in [22], where the authors prove that for large σ, any algorithm hoping to recover f must require $O(\sigma^6)$ samples.

We are interested in recovering one-dimensional manifolds (i.e., curves) in high dimensions in the presence of large Gaussian noise. For a particular class of curves, we will establish the sample complexity of the problem to be $O(\sigma^6)$ and provide a recovery algorithm that meets that lower bound. The setting of one-dimensional manifolds corresponds to temporal data, and we can think of noisy curves as noisy measurements of some system evolving in time; recovering the underlying curve thus corresponds to recovering the underlying dynamics. In the one-dimensional setting, a great deal of work has been done in the *seriation* problem, where noisy observations of time-dependent data are labeled with ordered timestamps. For instance, in cryo-EM, one might have noisy observations of a biological macromolecule undergoing some conformational change over time; in this context, the seriation problem

involves estimating the temporal order of these conformers [23, 24]. In archaeology, seriation techniques are used to date archeological finds [25]. Various spectral methods have been developed for the seriation problem [26–28], but often require strong structural assumptions on the data. More recently, the authors in [29] use the Fiedler vector of the graph Laplacian of the data to approach the seriation problem in a more general case. Note that our problem is not only to recover the time ordering of noisy points coming from a curve, but to recover the underlying curve itself. If the distribution of the noise is known, then the time ordering of noisy points can be computed from the underlying curve simply by associating each noisy data point with a point in the underlying curve that maximizes the likelihood.

2.1.2 Problem Formulation: Noisy Curve Recovery

Consider a parametric, non self-intersecting ground truth piecewise linear (PWL) curve $C^*(t)$ in $\mathbb{R}^d$, where $t \in [0, 1]$. We assume that the curve consists of a finite number of segments. In order to fix a scale, we assume that $\|C^*(t)\| < 1$ for all $t \in [0, 1]$. The *noisy curve model* consists of N independent observations

$$y^{(j)} = C^*(\tau^{(j)}) + \sigma\xi^{(j)}, \qquad j = 1, \ldots, N. \tag{2.1.1}$$

Here $\tau \sim U[0, 1]$ is drawn uniformly randomly and $\xi \sim \mathcal{N}(0, I_d)$ denotes isotropic Gaussian noise independent of τ; we assume σ is known. We can imagine this as a Gaussian "fattening" of the curve C^*. Let $P_{C^*,\sigma}$ denote the random variable given by $C^*(\tau) + \sigma\xi$, and let P_{C^*} denote the case where $\sigma = 0$. We formulate the *noisy curve recovery* problem as the estimation of C^* from observations $\{y^{(j)}\}_{j=1}^N$, as depicted in Figure 2.1. Note that we can think of this model as a continuous Gaussian mixture model, where the mixture centers are drawn continuously from $C^*(t)$.

In the low noise regime, the salient features of the underlying curve are still visible in the presence of noise, and the curve can be estimated by "tracing through" the point cloud coming

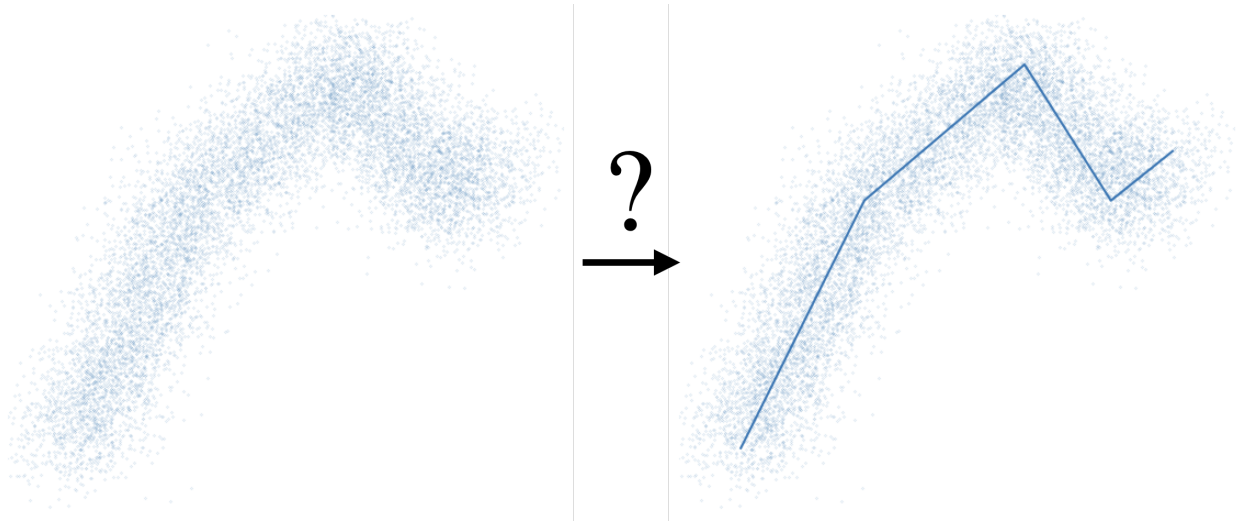

Figure 2.1: An illustration of the noisy curve recovery problem.

from (2.1.1). We provide a more detailed account of an algorithm in §A.1, but essentially, much like the expectation-maximization step in estimating finite Gaussian mixture models, we can assume that nearby points in the cloud come from nearby points on the underlying curve. We can thus use the local largest principal component of subsets of the point cloud to approximate the tangent space of the curve near that point. However, in the case where noise is large, we can no longer reliably assign points in the point cloud to a nearby point on the underlying curve. This is illustrated in Figure 2.2, where we demonstrate the results of tracing through a point cloud coming from a PWL curve in both the low and high noise case. We are henceforth interested in studying the high noise case.

In this chapter, we will first prove a result on the sample complexity of the high noise curve recovery problem, that is, the number of samples needed to recover the ground truth underlying curve from data up to some error. In particular, we will show that for sufficiently high noise, the number of samples from (2.1.1) needed to recover a curve scales with order σ^6. The proof relies on studying the moments of the distribution $P_{C^*,\sigma}$, particularly the third moment tensor. Our approach is based largely on the approach in the proof of the sample complexity of the MRA problem in [22]; the proof hinges on showing that the first two moments alone do not uniquely determine a curve in a neighborhood. We then prove

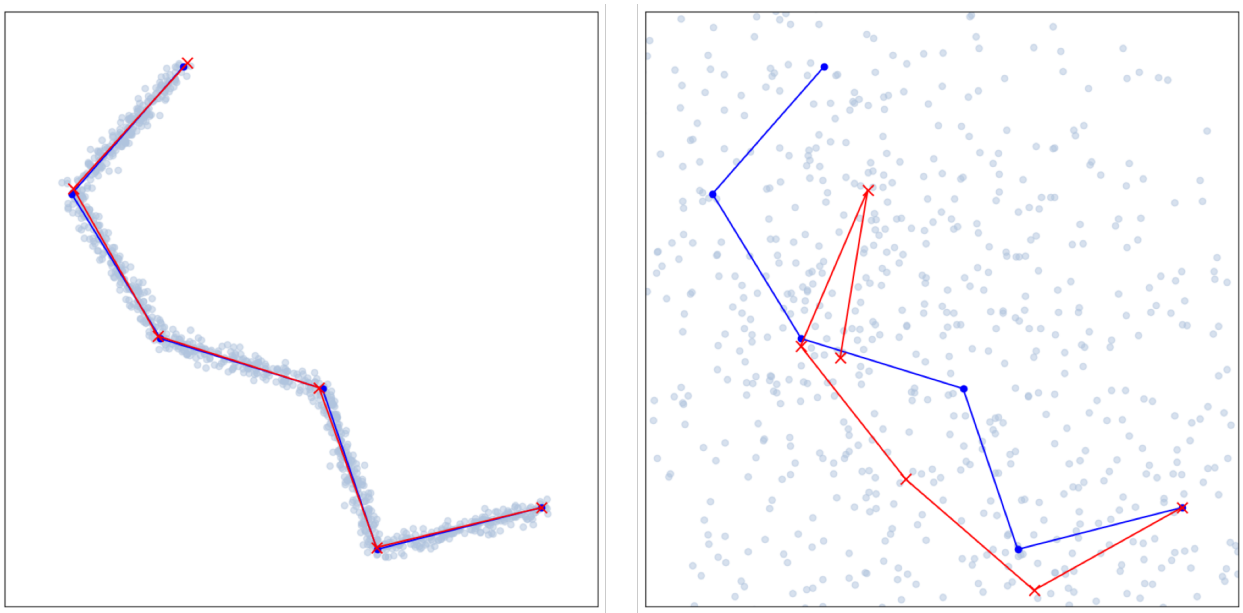

Figure 2.2: Results of a cloud tracing algorithm (detailed in §A.1) on a point cloud coming from a piecewise linear curve with low noise (left) and high noise (right). In both cases, a ground truth curve in blue is shown with an accompanying point cloud. When the noise is low (i.e., the scale of the noise is smaller than the scale of the geometric features of the curve), the structure of the underlying curve is still visible in the point cloud, and the curve can be successfully traced through (in red). When the noise is large, the local geometric structure of the curve is destroyed and the tracing approach fails.

that the problem of recovering the underlying curve from the *third* moment of the data is in fact locally well-posed, and provide an algorithm to obtain an initial rough estimate of the curve based on a decomposition of the third moment tensor of the curve. The idea of the algorithm is to use the tensor power method to approximate the subspaces in which vertices of the curve lie, then use this as an initialization to a gradient descent scheme to fit a curve defined by the vertices to the third moment.

2.1.3 Chapter Outline

This chapter is organized as follows. In §2.2, we introduce notation for piecewise linear curves and compute their moments. In §2.3, we state and outline the proof of the sample complexity result. In §2.4, we show that locally, the third moment uniquely determines piecewise linear curves in high dimensions. In §2.5, we use this local well-posedness to devise an algorithm that recovers piecewise linear curves. We show results from numerical experiments demonstrating the performance of our algorithm in §2.6.

2.2 Piecewise Linear Curves and their Moments

We restrict our analysis to piecewise linear curves in order to have a simple explicit representation of a curve. We also require that our curves have domain $[0, 1]$, are open (i.e., the two end points are not the same), and have constant speed (i.e., $\|C'(t)\|$ is constant).

Let $0 = t_0 < \cdots < t_M = 1$ be points in time and let $c_0 = c(t_0), \ldots, c_M = c(t_M) \in \mathbb{R}^d$ be the vertices of a PWL curve. Throughout this chapter, we use d to denote the ambient dimension of the curve and M to denote the number of segments. We assume in general that our dimension is high enough that $d \geq M$. We also assume that $c_i \neq c_j$ for all $i \neq j$. In particular $c_0 \neq c_M$, so our curve is open.

We have the following explicit expression for $C(t)$:

$$C(t) = \frac{t - t_{i-1}}{t_i - t_{i-1}} c_i - \frac{t - t_i}{t_i - t_{i-1}} c_{i-1}, \text{ when } t \in [t_{i-1}, t_i]_{i=1,\ldots,M}. \tag{2.2.1}$$

Since we require $C(t)$ to have constant speed, we must have

$$t_i = \frac{\sum_{k=1}^{i} \|c_k - c_{k-1}\|}{Z} \quad \text{for } i = 1, \ldots, M \tag{2.2.2}$$

where $Z := \sum_{k=1}^{M} \|c_k - c_{i-1}\|$ is the total length of the curve. The benefit of requiring constant speed is that our curves are completely characterized by the vertices $c_0, \ldots, c_M$. In a slight abuse of notation, we thus identify a PWL curve with the $(M+1, d)$-shaped matrix of its row-stacked vertices:

$$C = \begin{bmatrix} -c_0- \\ -c_1- \\ \vdots \\ -c_M- \end{bmatrix}. \tag{2.2.3}$$

We assume that the c_i are in generic position, so that the vectors $\{c_i - c_{i-1}\}_{i=1}^{M}$ are linearly independent.

Our approach to proving lower bounds on the sample complexity of noisy curve recovery is based on studying the moments of a curve, defined by

$$m_k(C) = \int_0^1 C(t)^{\otimes k} \, dt. \tag{2.2.4}$$

Here, $\cdot^{\otimes k}$ denotes the k-fold tensor power of a vector. For instance, for $v \in \mathbb{R}^d$, $v^{\otimes 3}$ is the (d, d, d)-shaped tensor whose jkl entry is given by $v_j v_k v_l$. The use of moment tensors for inverse problems has been studied in many other contexts, such as finite Gaussian mixture models in [30].

We now derive expressions for the first three moments of C in terms of the vertices

$c_0, \ldots, c_M$.

Proposition 2.2.5 (Moments of Piecewise Linear Curves)**.** *Let C be the constant speed PWL curve with domain $[0,1]$ with vertices $c_0, \ldots, c_M$. Let Z be the length of C. Then the first three moments of C are given by the formulas*

$$\begin{aligned} m_1(C) &= \frac{1}{2}\sum_{i=1}^{M} \frac{\|c_i - c_{i-1}\|}{Z}(c_{i-1} + c_i) \\ m_2(C) &= \frac{1}{6}\sum_{i=1}^{M} \frac{\|c_i - c_{i-1}\|}{Z}\left((c_{i-1} + c_i)^{\otimes 2} + c_{i-1}^{\otimes 2} + c_i^{\otimes 2}\right) \\ m_3(C) &= \frac{1}{12}\sum_{i=1}^{M} \frac{\|c_i - c_{i-1}\|}{Z}\left((c_{i-1} + c_i)^{\otimes 3} + 2c_{i-1}^{\otimes 3} + 2c_i^{\otimes 3}\right). \end{aligned} \tag{2.2.6}$$

Proof. Consider the change of variables

$$\frac{t - t_{i-1}}{t_i - t_{i-1}} \mapsto t, \qquad dt \mapsto (t_i - t_{i-1})dt \tag{2.2.7}$$

For any k, we can apply this change of variables and (2.2.2) to get an expression for $m_k(C)$ in terms of the vertices c_i:

$$\begin{aligned} m_k(C) &= \int_0^1 C(t)^{\otimes k}\, dt \\ &= \sum_{i=1}^{M} \int_{t_{i-1}}^{t_i} \left(\frac{t - t_{i-1}}{t_i - t_{i-1}} c_i - \frac{t - t_i}{t_i - t_{i-1}} c_{i-1}\right)^{\otimes k} dt \\ &= \sum_{i=1}^{M} \int_0^1 \left(tc_i + (1-t)c_{i-1}\right)^{\otimes k} (t_i - t_{i-1})\, dt \\ &= \sum_{i=1}^{M} \frac{\|c_i - c_{i-1}\|}{Z} \int_0^1 \left((1-t)c_{i-1} + tc_i\right)^{\otimes k} dt. \end{aligned} \tag{2.2.8}$$

The desired result can be obtained by expanding the tensor power in the integral for $k = 1, 2, 3$ and direct computation of the integral. ■

Recall our assumption that $d \geq M$ and observe that a PWL curve C with M segments

naturally lives in an M-dimensional subspace V of $\mathbb{R}^n$ spanned by the vectors $\{c_i - c_{i-1}\}_{i=1}^M$. We can in fact compute an orthonormal basis for this subspace by taking the top M eigenvectors of the second moment $m_2(C)$; indeed, the second moment is a rank M matrix for generic C. Therefore we will often assume $M = d$, since if $d > M$, we can project the coordinates of C down to V. Nonetheless, for semantic clarity, we will often use M when referring to the number of segments and d when referring to the dimension.

Later on, we will show two important results involving moment matching. We will show that matching a PWL curve to a third moment $m_3(C)$ is locally well-posed, while matching a PWL curve to the first two moments $m_1(C)$ and $m_2(C)$ is locally ill-posed. A careful analysis of the moments is difficult owing to the $\|c_i - c_{i-1}\|/Z$ terms (note that Z also depends on C). To aid in our analysis and make things easier to compute, we introduce the following *relaxed* curve moments that replace the $\|c_i - c_{i-1}\|/Z$ terms with new variables. We will show that it is sufficient to consider moment matching in this relaxed setting.

Definition 2.2.9 (Relaxed Curve Moments)**.** Let $C \in \mathbb{R}^{(M+1)\times d}$ and $p \in \mathbb{R}^M$. Then we define the *relaxed moments* of C to be

$$\begin{aligned}
\mu_1(C,p) &= \frac{1}{2}\sum_{i=1}^{M} p_i(c_{i-1} + c_i) \\
\mu_2(C,p) &= \frac{1}{6}\sum_{i=1}^{M} p_i\left((c_{i-1} + c_i)^{\otimes 2} + c_{i-1}^{\otimes 2} + c_i^{\otimes 2}\right) \\
\mu_3(C,p) &= \frac{1}{12}\sum_{i=1}^{M} p_i\left((c_{i-1} + c_i)^{\otimes 3} + 2c_{i-1}^{\otimes 3} + 2c_i^{\otimes 3}\right).
\end{aligned} \tag{2.2.10}$$

We can rewrite these in a more vectorized way that makes our analysis more straightforward.

$$\begin{aligned}
\mu_1(C,p) &= \frac{1}{2}C^T A^s p \\
\mu_2(C,p) &= C^T \overline{A}(p) C \\
\mu_3(C,p) &= C^{\otimes 3} \star \alpha(p).
\end{aligned} \tag{2.2.11}$$

Here, A^s is an $(M+1, M)$-shaped matrix constant and $\overline{A}(p)$ is a symmetric $(M+1, M+1)$-shaped matrix that is a linear function of p. The six-way tensor $C^{\otimes 3}$ has shape $(M+1, d, M+1, d, M+1, d)$ and has $mjnkol$ entry given by $C_{mj}C_{nk}C_{ol}$. The three-way tensor $\alpha(p)$ is a linear function of p and is symmetric with shape $(M+1, M+1, M+1)$. The operation $\star$ denotes the natural tensor contraction between tensors of shape $(M+1, d, M+1, d, M+1, d)$ and $(M+1, M+1, M+1)$:

$$[X \star y]_{jkl} = \sum_{m,n,o=0}^{M} X_{mjnkol} y_{jkl}. \tag{2.2.12}$$

The exact forms of A^s, $\overline{A}(p)$, and $\alpha(p)$ are given in §A.4. Note that in the case where p is the vector of proportional segment lengths of C, i.e., $p_i = \|c_i - c_{i-1}\|/Z$ for all $i = 1, \ldots, M$, the relaxed moments are precisely equal to the actual moments.

We now present expressions for the Jacobians of the error between a predicted relaxed third moment and a ground truth third moment. The proof is by standard matrix calculus techniques, which we omit.

Lemma 2.2.13 (Jacobians of Relaxed Moments)**.** *Let $C \in \mathbb{R}^{(M+1)\times d}$ be the matrix representation of a PWL curve and let $p \in \mathbb{R}^M$; note that we do not require p to be equal to the proportional segment lengths of C. Let $m_3 \in \mathbb{R}^{d\times d\times d}$ be some third moment tensor. Define the third moment loss*

$$L_{m_3}(C, p) = \|\mu_3(C, p) - m_3\|_F^2, \tag{2.2.14}$$

where $\|\cdot\|_F^2$ is the squared Frobenius norm on tensors, i.e., the sum of the squared entries. Let $y \in \mathbb{R}^{(M+1)\times d}$ and $z \in \mathbb{R}^M$. Let $\alpha(p)$ be as in (A.4.3) and $\star$ be as in (2.2.12). Then the derivatives of the third moment loss with respect to C and p applied to y and z respectively

are given by

$$\langle \nabla_C L_{m_3}(C,p), y\rangle = \Big\langle 2\left[C^{\otimes 3}\star\alpha(p) - m_3\right], \\ (C\otimes C\otimes y + C\otimes y\otimes C + y\otimes C\otimes C)\star\alpha(p)\Big\rangle \tag{2.2.15}$$

$$\left\langle \nabla_p L_{m_3}(C,p), z\right\rangle = \Big\langle 2\left[C^{\otimes 3}\star\alpha(p) - m_3\right], C^{\otimes 3}\star\alpha(z)\Big\rangle.$$

2.3 The Sample Complexity Lower Bound

To talk about the sample complexity of recovering PWL curves, we need a measure of the distance between two curves. We choose the following natural distance.

Notation 2.3.1. Let $C(t)$ and $\Gamma(t)$ be two open parametric curves on $[0,1]$. Then we define the distance between the two curves to be the average squared Euclidean distance between the curves:

$$\rho(C,\Gamma) = \int_0^1 \|C(t)-\Gamma(t)\|^2\, dt. \tag{2.3.2}$$

We assume that $\Gamma(t)$ is parameterized so that $\rho(C(t),\Gamma(t)) \le \rho(C(t),\Gamma(1-t))$; i.e., Γ is oriented in the direction that minimizes the distance.

We first prove that curve recovery from the first and second moment alone is not locally well-posed. In other words, we can find two curves that are arbitrarily close together with exactly the same first and second moment. As discussed in §2.2, we assume that the number of segments M is equal to the ambient dimension d.

Proposition 2.3.3. *Let $C^*(t) : [0,1] \to \mathbb{R}^d$ be a PWL constant speed curve with $M = d$ segments. For almost all such C^*, for sufficiently small $\epsilon > 0$, there exists a PWL curve $\Gamma(t) \neq C^*(t)$ with $m_1(C^*) = m_1(\Gamma)$ and $m_2(C^*) = m_2(\Gamma)$ such that $\|C^* - \Gamma\|_F = \epsilon$, where the norm here is on the matrix representation of the curves.*

The detailed proof is provided in §A.2; the idea is that matching the first and second moment of a curve can be expressed as an underdetermined system of polynomials whose solutions form a manifold of positive dimension.

Since C^* and Γ being close as matrices implies that $C^*(t)$ and $\Gamma(t)$ are close as curves, we have the following immediate corollary of Proposition 2.3.3.

Corollary 2.3.4. *As in Proposition 2.3.3, let $C^*(t) : [0,1] \to \mathbb{R}^d$ be a PWL constant speed curve with $M = d$ segments. For almost all such C^*, for sufficiently small $\epsilon > 0$, there exists a PWL curve $\Gamma(t) \neq C^*(t)$ with $m_1(C^*) = m_1(\Gamma)$ and $m_2(C^*) = m_2(\Gamma)$ such that $\|C^*(t) - \Gamma(t)\| \leq \epsilon$ for all $t \in [0,1]$.*

The application of Corollary 2.3.4 is to show that the hypotheses of the following result are not vacuous for $\ell = 3$. The proof is rather technical and left to §A.3.

Proposition 2.3.5. *Let C, Γ be mean zero parametric curves such that $\|C(t) - \Gamma(t)\| \leq 1/3$ and $\|C(t)\|, \|\Gamma(t)\| < 1$ for all $t \in [0,1]$. Suppose $m_k(C) = m_k(\Gamma)$ for all $k \leq \ell - 1$. Then for $\sigma \geq 1$, we have the following bound on the χ^2 divergence between $P_{C,\sigma}$ and $P_{\Gamma,\sigma}$:*

$$\chi^2(P_{C,\sigma} \| P_{\Gamma,\sigma}) \leq \mathcal{K}_\ell \sigma^{-2\ell} \rho(C, \Gamma), \tag{2.3.6}$$

where $\mathcal{K}_\ell$ is a constant that depends on ℓ only.

This result allows us to bound the χ^2 divergence between two noisy curve distributions by $\sigma^{-2\ell}$ if the moments of the underlying curves match up to (but not including) order ℓ. This is the primary ingredient in the proof of the following main sample complexity result. Note that we use $\lesssim$ to denote inequality up to a universal constant.

Theorem 2.3.7. *Let C be a curve as in be as in Proposition 2.3.5. Given any $\Delta > 0$ sufficiently small, there exists another curve $\Gamma \neq C$ satisfying the hypotheses of Proposition 2.3.5 such that $\rho(C, \Gamma) = \Delta$, where Γ is indistinguishable from C in the following sense. Consider N samples $Y := \{y^{(1)}, \ldots, y^{(N)}\}$ and the task of deciding if Y came from $P_{C,\sigma}$*

or $P_{\Gamma,\sigma}$. Assume a uniform prior on whether the ground truth curve is C or Γ. Then if $N \lesssim \sigma^6 \Delta^{-1}$, the probability of making an error is bounded below by $1/4$.

Proof. Given $C(t)$, construct $\Gamma(t)$ such that the first two moments match and $\rho(C, \Gamma) = \Delta$ as in Proposition 2.3.3. Then by Theorem 2.3.5, we have the bound $\chi^2(P_{C,\sigma} \| P_{\Gamma,\sigma}) \leq \mathcal{K}_3 \sigma^{-6} \rho(C, \Gamma)$. Let $P_{C,\sigma}^{\otimes N}$ and $P_{\Gamma,\sigma}^{\otimes N}$ be the distribution of N independent draws from $P_{C,\sigma}$ and $P_{\Gamma,\sigma}$, respectively. By Lemma A.5.4, Lemma A.5.6, and Lemma A.5.8,

$$\mathrm{TV}(P_{C,\sigma}^{\otimes N}, P_{\Gamma,\sigma}^{\otimes N})^2 \leq \frac{1}{2} \mathrm{KL}(P_{C,\sigma}^{\otimes N} \| P_{\Gamma,\sigma}^{\otimes N}) \leq \frac{1}{2} \mathcal{K}_3 \sigma^{-6} \Delta N. \tag{2.3.8}$$

Therefore, for $N \leq \frac{1}{2} \mathcal{K}_3^{-1} \sigma^6 \Delta^{-1}$, we have

$$\mathrm{TV}(P_{C,\sigma}^{\otimes N}, P_{\Gamma,\sigma}^{\otimes N}) \leq \frac{1}{2}. \tag{2.3.9}$$

Now let $\psi : \mathbb{R}^{d \times N} \to \{C, \Gamma\}$ be any measurable function of the data Y. Let A denote the set of all Y such that $\psi(Y) = \Gamma$. Then

$$\begin{aligned} P_{C,\sigma}^{\otimes N}(\psi(Y) = \Gamma) + P_{\Gamma,\sigma}^{\otimes N}(\psi(Y) = C) &= P_{C,\sigma}^{\otimes N}(A) + P_{\Gamma,\sigma}^{\otimes N}(A^c) \\ &= 1 - (P_{\Gamma,\sigma}^{\otimes N}(A) - P_{C,\sigma}^{\otimes N}(A)) \\ &\geq 1 - \sup_A (P_{\Gamma,\sigma}^{\otimes N}(A) - P_{C,\sigma}^{\otimes N}(A)) \\ &= 1 - \mathrm{TV}(P_{\Gamma,\sigma}^{\otimes N}, P_{C,\sigma}^{\otimes N}) \\ &\geq 1/2. \end{aligned} \tag{2.3.10}$$

Let X denote the event that the ground truth curve is C and X^c denote the event that the ground truth curve is Γ. By our uniform prior assumption, we have $\mathbb{P}(X) = \mathbb{P}(X^c) = 1/2$.

Then the probability of the hypothesis test ψ making an error is

$$\begin{aligned}\mathbb{P}(\psi(Y) = \Gamma|X)\mathbb{P}(X) + \mathbb{P}(\psi(Y) = C|X^c)\mathbb{P}(X^c) \\ &= \frac{1}{2}P_{C,\sigma}^{\otimes N}(\psi(Y) = \Gamma) + \frac{1}{2}P_{\Gamma,\sigma}^{\otimes N}(\psi(Y) = C) \\ &\geq \frac{1}{4}.\end{aligned} \tag{2.3.11}$$

■

This result reveals a fundamental limitation of the curve recovery problem in the high noise regime; regardless of the method used, it is impossible to discern with high probability which of two Δ-separated curves the data Y come from without at least $O(\sigma^6)$ samples.

2.4 Local Well-Posedness of Third Moment-Based Recovery of Piecewise Linear Curves

Theorem 2.3.7 gives us a lower bound of $O(\sigma^6)$ on the sample complexity of recovering a curve from data. We now show that for PWL curves with constant speed, this lower bound is asymptotically tight by showing that $O(\sigma^6)$ is indeed enough samples for recovery. We accomplish this by showing that the third moment of a noise-free PWL curve uniquely determines a PWL curve in a neighborhood. This is contrast to the result of Proposition 2.3.3, which says that determining a curve from the first two moments alone is locally ill-posed. As before, in this section we assume $M = d$.

Our ultimate goal in this section is to show the following.

Theorem 2.4.1. *Let $C^* \in \mathbb{R}^{(M+1)\times d}$ be a ground truth PWL curve with proportional segment lengths $p^* \in \mathbb{R}^M$, where $M = d \geq 4$. Let $m_3^* = \mu_3(C^*, p^*)$ be the ground truth third moment. Let $L_{m_3^*}$ be the third moment squared Frobenius loss as defined in (2.2.14) Then for almost all C^*, if C is sufficiently close to C^* and p is sufficiently close to p^*, we have*

$$\left\langle \nabla_C L_{m_3^*}(C, p), C^* - C \right\rangle < 0 \tag{2.4.2}$$

$$\left\langle \nabla_p L_{m_3^*}(C,p), p^* - p \right\rangle < 0. \tag{2.4.3}$$

Local well-posedness of recovery from C is an immediate corollary of this.

Corollary 2.4.4. *Let C^*, p^*, and m_3^* as in Theorem 2.4.1. Then for almost all C^*, there exists a neighborhood U of (C^*, p^*) such that C^* is the only PWL curve in U with third moment m_3^*.*

Proof. The result of Theorem 2.4.1 implies that (C^*, p^*) is an isolated minimum of $L_{m_3^*}(C,p)$. ■

To prove Theorem 2.4.1, we will first need the following weaker result.

Lemma 2.4.5. *Let $C^* \in \mathbb{R}^{(M+1)\times d}$ be a ground truth PWL curve with proportional segment lengths $p^* \in \mathbb{R}^M$, where $M = d \geq 4$. Let $m_3^* = \mu_3(C^*, p^*)$ be the ground truth third moment. Then for almost all C^*, for C sufficiently close to C^* and p sufficiently close to p^*, we have*

$$\left\langle \nabla_C L_{m_3^*}(C,p^*), C^* - C \right\rangle < 0 \tag{2.4.6}$$

$$\left\langle \nabla_p L_{m_3^*}(C^*,p), p^* - p \right\rangle < 0. \tag{2.4.7}$$

Equation (2.4.6) differs from (2.4.2) in that we have replaced the p in $L_{m_3^*}(C,p)$ with p^*, and similarly for (2.4.7) and (2.4.3). In other words, (2.4.6) says that recovery of C from m_3^* and p^* is locally well-posed, while (2.4.7) says that recovery of p from m_3^* and C^* is locally well-posed. On the other hand, the stronger statements (2.4.2) and (2.4.3) say that recovery of C and p from m_3^* is locally well-posed. We prove (2.4.6) in §2.4.1, (2.4.7) in §2.4.2, and (2.4.2) and (2.4.3) in §2.4.3.

2.4.1 Recovery of Curve Vertices from Proportional Segment Lengths

Let $\delta \in \mathbb{R}^{(M+1)\times d}$ be sufficiently small and $C = C^* + \delta$ be some other curve in a neighborhood of C^*. Then we wish to show that for C sufficiently close to C^*, we have

$$\left\langle \nabla_C L_{m_3^*}(C, p^*), C^* - C \right\rangle < 0$$

Using (2.2.15), the left hand side of (2.4.6) becomes

$$\begin{aligned} 2\Big\langle & (C^{\otimes 3} - C^{*\otimes 3}) \star \alpha(p^*), \\ & (C \otimes C \otimes C^* + C \otimes C^* \otimes C + C^* \otimes C \otimes C - 3C^{\otimes 3}) \star \alpha(p^*) \Big\rangle \end{aligned} \tag{2.4.8}$$

We perform a Taylor expansion, replacing C with $C^* + \delta$, expanding the nonlinear terms and only keeping terms with one δ. Then (2.4.8) has first order approximation given by

$$-2 \left\| (C^* \otimes C^* \otimes \delta + C^* \otimes \delta \otimes C^* + \delta \otimes C^* \otimes C^*) \star \alpha(p^*) \right\|_F^2 . \tag{2.4.9}$$

Let $g(\delta)$ be the term inside the norm:

$$g(\delta) := (C^* \otimes C^* \otimes \delta + C^* \otimes \delta \otimes C^* + \delta \otimes C^* \otimes C^*) \star \alpha(p^*). \tag{2.4.10}$$

This is a linear operator from $\mathbb{R}^{(M+1)\times d} \to \mathbb{R}^{d\times d\times d}$. Therefore, to show (2.4.6), it suffices to show that the kernel of g is trivial.

Let $\mathrm{Sym} : \mathbb{R}^{d\times d\times d} \to \mathbb{R}^{d\times d\times d}$ denote the symmetrization operator on three-way tensors, i.e.,

$$\mathrm{Sym}(X)_{jkl} = \frac{1}{6}\left(X_{jkl} + X_{klj} + X_{ljk} + X_{lkj} + X_{kjl} + X_{jlk}\right) \tag{2.4.11}$$

Note that Sym is linear. Then by the definition of the $\star$ operation and the symmetry of

$\alpha(p^*)$, we can rewrite (2.4.10) as

$$g(\delta) = \mathrm{Sym}\left(3(\delta \otimes C^* \otimes C^*) \star \alpha(p^*)\right). \tag{2.4.12}$$

Let $\tilde{g}$ denote the term inside the Sym:

$$\tilde{g}(\delta) := 3(\delta \otimes C^* \otimes C^*) \star \alpha(p^*). \tag{2.4.13}$$

To show that $g(\delta)$ has trivial kernel, it suffices to show that (a) $\tilde{g}$ has trivial kernel and (b) $\mathrm{Im}(\tilde{g}) \cap \mathrm{Ker}(\mathrm{Sym}) = \{0\}$. We prove (a) in Lemma 2.4.14 and (b) in Lemma 2.4.17.

Lemma 2.4.14. *Let $M = d$. Let $C^* = [c_0^*, \ldots, c_M^*]^T$ and assume that $c_0^*, \ldots, c_M^* \in \mathbb{R}^d$ are in generic position so that any collection of M of the c_i^*'s is linearly independent. Let $\tilde{g}$ be the linear operator given in (2.4.13). Then $\tilde{g}$ has trivial kernel.*

Proof. Note that $\tilde{g}(\delta)$ is a linear operator from $\mathbb{R}^{(M+1)\times d} \to \mathbb{R}^{d\times d\times d}$. If we think of this as a matrix operating on the flattened spaces, then with our assumption of $M = d$, the matrix representation of $\tilde{g}$ is a tall and thin matrix. To show that $\tilde{g}$ has trivial kernel, it suffices to show that the $(M+1)d$ "columns" of $\tilde{g}$ are linearly independent.

Let $E^{m,j}$ be the standard basis matrix in $\mathbb{R}^{(M+1)\times d}$ with all zeros except a 1 in the m, j entry; note that m ranges from 0 to M while j ranges from 1 to d, i.e. we zero-index the rows but one-index the columns. Then the (m, j) "column" of $\tilde{g}$ is given by $\tilde{g}(E^{m,j})$. Let $E_i^{m,j}$ denote the ith row of $E^{m,j}$, which is equal to e_j (the jth standard basis vector in $\mathbb{R}^d$) if $i = m$ and zero otherwise. Then

$$\begin{aligned}\tilde{g}(E^{m,j}) &= \sum_{i=1}^{M} p_i^* \Bigg[\frac{1}{4}(E_{i-1}^{m,j} + E_i^{m,j}) \otimes (c_{i-1}^* + c_i^*) \otimes (c_{i-1}^* + c_i^*) \\ &\qquad + \frac{1}{2}E_{i-1}^{m,j} \otimes c_{i-1}^* \otimes c_{i-1}^* + \frac{1}{2}E_i^{m,j} \otimes c_i^* \otimes c_i^*\Bigg] \\ &= \frac{1}{4}p_m^* e_j \otimes (c_{m-1}^* + c_m^*) \otimes (c_{m-1}^* + c_m^*) + \frac{1}{2}p_m^* e_j \otimes c_m^* \otimes c_m^*\end{aligned}$$

$$+\frac{1}{4}p^*_{m+1}e_j\otimes(c^*_m+c^*_{m+1})\otimes(c^*_m+c^*_{m+1})+\frac{1}{2}p^*_{m+1}e_j\otimes c^*_m\otimes c^*_m$$
$$=[0,\ldots,0,F^m,0,\ldots,0]\,.$$

In the last line, we are notating a $(M+1,M+1,M+1)$-shaped tensor as a size $M+1$ vector consisting of size $(M+1,M+1)$ matrices, where the matrix F^m is in the jth slot of the vector and is given by

$$F^m=\begin{cases}\frac{1}{4}p^*_1(c^*_0+c^*_1)^{\otimes 2}+\frac{1}{2}p^*_1{c^*_0}^{\otimes 2} & \text{if } m=0\\ \frac{1}{4}p^*_m(c^*_{m-1}+c^*_m)^{\otimes 2}+\frac{1}{4}p^*_{m+1}(c^*_m+c^*_{m+1})^{\otimes 2} & \\ \qquad+\frac{1}{2}p^*_m{c^*_m}^{\otimes 2}+\frac{1}{2}p^*_{m+1}{c^*_m}^{\otimes 2} & \text{if } 1\le m\le M-1\\ \frac{1}{4}p^*_M(c^*_{M-1}+c^*_M)^{\otimes 2}+\frac{1}{2}p^*_M{c^*_M}^{\otimes 2} & \text{if } m=M\end{cases}\tag{2.4.15}$$

From the position of the nonzero entries, it is clear that $\tilde{g}(E^{m,j})$ are linearly independent for fixed m and varying j. It remains to show that there is also linear independence for fixed j and varying m; in other words, we need to show that $\{F^m\}_{m=0}^M$ is a linearly independent collection of matrices; we verify this in Lemma A.5.9. ■

Before we show $\mathrm{Im}(\tilde{g})\cap\mathrm{Ker}(\mathrm{Sym})=\{0\}$, we first need the following result, which we have adapted from Lemma 5.6 in [31].

Lemma 2.4.16. *Let $\mathcal{M}(x)$ be a matrix whose entries are analytic functions of a variable $x\in U\subset\mathbb{R}^n$ for some connected open domain U. Let r be the rank of $\mathcal{M}(x_0)$ for some particular $x_0\in\mathbb{R}^n$. Then $\mathrm{rank}(\mathcal{M}(x))\ge r$ for almost all $x\in U$.*

Proof. Note that $\mathrm{rank}(\mathcal{M}(x))<r$ if and only if the determinant of every (r,r)-shaped minor of $\mathcal{M}(x)$ vanishes. Since the rank of $\mathcal{M}(x_0)$ is r, there exists some (r,r)-shaped minor $\hat{\mathcal{M}}(x)$ such that $\det(\hat{\mathcal{M}}(x_0))\neq 0$. Note that $\det(\hat{\mathcal{M}}(x))$ is an analytic function of x, and the witness x_0 shows that $\det(\hat{\mathcal{M}}(x))$ is not identically zero. An analytic function that is not identically zero has a measure zero vanishing set [32]. Therefore, $\det(\hat{\mathcal{M}}(x))$ has a vanishing

set of measure zero, and hence rank($\mathcal{M}(x)$) is $< r$ on a measure zero set. ■

The utility of this lemma is that if a matrix $\mathcal{M}(x)$ has entries that are analytic and we find a single witness x_0 such that $\mathcal{M}(x_0)$ is full rank, then we know $\mathcal{M}(x)$ is full rank for almost every x. We use this lemma in the proof of the following result.

Lemma 2.4.17. *Let $d \geq 4$ and $M = d$. Then for almost all $C^* \in \mathbb{R}^{(M+1)\times d}$, the image of $\tilde{g}$ and the kernel of* Sym *are linear subspaces that intersect only at* 0.

Proof. In the proof of Lemma 2.4.14, we showed that $\left\{\tilde{g}(E^{m,j})\right\}_{0\leq m\leq M, 1\leq j\leq d}$ forms a basis for $\mathrm{Im}(\tilde{g})$. Since the action of $\tilde{g}$ depends on C^*, in this proof we will use the notation $\tilde{g}_{C^*}$ to denote the $\tilde{g}$ operator induced by a particular choice of C^*. For coefficients $\beta_{m,j}$, we wish to show that if

$$\mathrm{Sym}\left(\sum_{\substack{0\leq m\leq M\\ 1\leq j\leq d}} \beta_{m,j}\tilde{g}_{C^*}(E^{m,j})\right) = 0, \tag{2.4.18}$$

then all the coefficients must be zero. By linearity of the Sym operator, it suffices to show that

$$\left\{\mathrm{Sym}(\tilde{g}_{C^*}(E^{m,j}))\right\}_{0\leq m\leq M, 1\leq j\leq d} \tag{2.4.19}$$

forms a linearly independent set of (M, M, M)-shaped tensors.

Let $\mathcal{M}(C^*)$ denote the $(M^3, (M+1)M)$-shaped matrix of column-stacked flattenings of the tensors in (2.4.19). Then we wish to show that this matrix has full column rank for almost all C^*. The entries of $\mathcal{M}(C^*)$ are either zeros or linear combinations of the entries of F^m as defined in (2.4.15). Recall that the values p_i^* are a function of C^* in the following way:

$$p_i^* = p_i^*(C^*) = \frac{\|c_i^* - c_{i-1}^*\|}{\sum_{j=1}^{M}\|c_j^* - c_{j-1}^*\|}. \tag{2.4.20}$$

On the connected open domain $U := \{C^* \in \mathbb{R}^{(M+1)\times d} : c_j^* - c_{j-1}^* \neq 0,\ j = 1, \ldots M\}$, this is an analytic function of the entries of C^*. Since the entries of terms of the form $(c_{m-1}^* + c_m^*)^{\otimes 2}$ and ${c_m^*}^{\otimes 2}$ are also analytic, it follows from the form of F^m that the entries of $\mathcal{M}(C^*)$ are

analytic functions of the entries of C^*. By Lemma 2.4.16, it suffices then to find a single witness C^* for each dimension M where $\mathcal{M}(C^*)$ has full rank.

For $M \geq 4$, we define the following witness C^M (for notational clarity, we show the case $M = 4$ here, with C^M for greater M being defined in the obvious way):

$$C^M := \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} + 1 \end{bmatrix}. \tag{2.4.21}$$

We choose this particular C^M because it induces matrices F^m that are sparse and have a particular recursive structure, and the proportional segment lengths of C^M are all $1/M$. Define

$$\begin{aligned} w_M^{m,j} &:= M \operatorname{Sym}(\tilde{g}_{C^M}(E^{m,j})) \\ W^M &= \{w_M^{m,j}\}_{0 \leq m \leq M, 1 \leq j \leq M}. \end{aligned} \tag{2.4.22}$$

To show that C^M is a witness for a full rank $\mathcal{M}(C^*)$, it suffices to show that for all $M \geq 4$, W^M is a linearly independent collection of tensors. We proceed by induction. The base case $M = 4$ can be verified by straightforward computation. We verify this numerically by computing the singular value decomposition of the matrix $\mathcal{M}(C^4)$ and observing that the smallest singular value is positive; the smallest singular value is approximately 3.29024×10^{-4}.

Now suppose W^M is a linearly independent collection; we wish to show that W^{M+1} also is a linearly independent collection. Note that $w_{M+1}^{m,j}$ is an $(M+1, M+1, M+1)$-shaped tensor. Let $\hat{w}_{M+1}^{m,j}$ denote the (M, M, M)-shaped "minor" of $w_{M+1}^{m,j}$ given by the $(1:M, 1:M, 1:M)$ subtensor of $w_M^{m,j}$ (here we are 1-indexing). By straightforward explicit computation of the entries of $M \operatorname{Sym}(\tilde{g}_{C^M}(E^{m,j}))$ and $(M+1)\operatorname{Sym}(\tilde{g}_{C^{M+1}}(E^{m,j}))$, it can

be shown that the minors of $w_{M+1}^{m,j}$ have the following properties:

(a) For $0 \le m \le M-2$, $1 \le j \le M$: $\hat{w}_{M+1}^{m,j} = w_M^{m,j}$.

(b) For $1 \le j \le M$: $\hat{w}_{M+1}^{M-1,j} = w_M^{M-1,j} - \left(\frac{1}{\sqrt{2}} - \frac{1}{4}\right) w_M^{M,j}$.

(c) For $1 \le j \le M$: $\hat{w}_{M+1}^{M,j} = \frac{1}{4(1+\sqrt{2})^2} w_M^{M,j}$.

(d) For $1 \le j \le M$: $\hat{w}_{M+1}^{M+1,j} = 0$.

(e) For $0 \le m \le M+1$: $\hat{w}_{M+1}^{m,M+1} = 0$.

Now given coefficients $\alpha_{m,j}$, suppose

$$\sum_{\substack{0 \le m \le M+1 \\ 1 \le j \le M+1}} \alpha_{m,j} w_{M+1}^{m,j} = 0. \tag{2.4.23}$$

To complete the inductive step, it suffices to show that all coefficients $\alpha_{m,j}$ must be zero. Observe that the $(1:M, 1:M, 1:M)$ subtensor of the linear combination above must also be zero, and hence using the properties above we have

$$\begin{aligned}
0 &= \sum_{\substack{0 \le m \le M+1 \\ 1 \le j \le M+1}} \alpha_{m,j} \hat{w}_{M+1}^{m,j} \\
&= \sum_{\substack{0 \le m \le M-2 \\ 1 \le j \le M}} \alpha_{m,j} \hat{w}_{M+1}^{m,j} + \sum_{1 \le j \le M} \alpha_{M-1,j} \hat{w}_{M+1}^{M-1,j} + \sum_{1 \le j \le M} \alpha_{M,j} \hat{w}_{M+1}^{M,j} \\
&\qquad + \sum_{1 \le j \le M} \alpha_{M+1,j} \hat{w}_{M+1}^{M+1,j} + \sum_{0 \le m \le M+1} \alpha_{m,M+1} \hat{w}_{M+1}^{m,M+1} \\
&= \sum_{\substack{0 \le m \le M-2 \\ 1 \le j \le M}} \alpha_{m,j} w_M^{m,j} + \sum_{1 \le j \le M} \alpha_{M-1,j} \left(w_M^{M-1,j} - \left(\frac{1}{\sqrt{2}} - \frac{1}{4}\right) w_M^{M,j} \right) \\
&\qquad + \sum_{1 \le j \le M} \alpha_{M,j} \frac{1}{4(1+\sqrt{2})^2} w_M^{M,j} \\
&= \sum_{\substack{0 \le m \le M-2 \\ 1 \le j \le M}} \alpha_{m,j} w_M^{m,j} + \sum_{1 \le j \le M} \alpha_{M-1,j} w_M^{M-1,j}
\end{aligned}$$

$$+ \sum_{1 \le j \le M} \left(\alpha_{M,j} \frac{1}{4(1+\sqrt{2})^2} - \alpha_{M-1,j} \left(\frac{1}{\sqrt{2}} - \frac{1}{4} \right) \right) w_M^{M,j}.$$

This is a linear combination of the terms $\{w_M^{m,j}\}_{0 \le m \le M, 1 \le j \le M} = W^M$; by the inductive hypothesis, this is a set of linearly independent tensors. In the last line above, we see that all $\alpha_{m,j}$ in the first sum must be zero and all $\alpha_{M-1,j}$ in the second sum must be zero. Therefore all $\alpha_{M-1,j}$ in the third sum of the last line must be zero, and hence all $\alpha_{M,j}$ in the third sum must be zero as well. We can thus conclude that in (2.4.23), all $\{\alpha_{m,j}\}_{0 \le m \le M, 1 \le j \le M}$ must be zero. Therefore, most of the terms in (2.4.23) vanish, and we are left with

$$\begin{aligned} \sum_{\substack{0 \le m \le M+1 \\ 1 \le j \le M+1}} \alpha_{m,j} w_{M+1}^{m,j} &= \sum_{0 \le m \le M} \alpha_{m,M+1} w_{M+1}^{m,M+1} + \sum_{1 \le j \le M+1} \alpha_{M+1,j} w_{M+1}^{M+1,j} \\ &= 0. \end{aligned} \tag{2.4.24}$$

Now consider the $(M+1,:,:)$ slice of the above tensor equation, which must equal the zero matrix. For $M+1=5$, the $(M+1,:,:)$ slices of $\{w_{M+1}^{m,M+1}\}_{0 \le m \le M}$ and $\{w_{M+1}^{M+1,j}\}_{1 \le j \le M+1}$ are as follows, with the slices for greater values of $M+1$ being the obvious extension of the

below (with the nonzero values being the same and more zero padding).

$$
\begin{aligned}
w_{M+1}^{0,M+1}[M+1,:,:] &= \begin{bmatrix} 1/8 & 1/24 & 0 & 0 & 0 \\ 1/24 & 1/24 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} & w_{M+1}^{M+1,1}[M+1,:,:] &= \begin{bmatrix} 0 & 0 & 0 & 0 & \frac{(1+\sqrt{2})^2}{6} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \frac{(1+\sqrt{2})^2}{6} & 0 & 0 & 0 & 0 \end{bmatrix} \\
w_{M+1}^{1,M+1}[M+1,:,:] &= \begin{bmatrix} 1/24 & 1/24 & 0 & 0 & 0 \\ 1/24 & 1/4 & 1/24 & 0 & 0 \\ 0 & 1/24 & 1/24 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} & w_{M+1}^{M+1,2}[M+1,:,:] &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{(1+\sqrt{2})^2}{6} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{(1+\sqrt{2})^2}{6} & 0 & 0 & 0 \end{bmatrix} \\
w_{M+1}^{2,M+1}[M+1,:,:] &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1/24 & 1/24 & 0 & 0 \\ 0 & 1/24 & 1/4 & 1/24 & 0 \\ 0 & 0 & 1/24 & 1/24 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} & w_{M+1}^{M+1,3}[M+1,:,:] &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{(1+\sqrt{2})^2}{6} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{(1+\sqrt{2})^2}{6} & 0 & 0 \end{bmatrix} \\
w_{M+1}^{3,M+1}[M+1,:,:] &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/24 & 1/24 & 0 \\ 0 & 0 & 1/24 & 1/4 & 1/12 \\ 0 & 0 & 0 & 1/12 & 1/8 \end{bmatrix} & w_{M+1}^{M+1,4}[M+1,:,:] &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{(1+\sqrt{2})^2}{6} \\ 0 & 0 & 0 & \frac{(1+\sqrt{2})^2}{6} & 0 \end{bmatrix} \\
w_{M+1}^{4,M+1}[M+1,:,:] &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/24 & 1/12 \\ 0 & 0 & 0 & 1/12 & \frac{11}{8}+\frac{1}{\sqrt{2}} \end{bmatrix} & w_{M+1}^{M+1,5}[M+1,:,:] &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{(1+\sqrt{2})^2}{2} \end{bmatrix}
\end{aligned}
\tag{2.4.25}
$$

One can show that for all dimensions M, these matrices are all linearly independent, and hence the remaining coefficients in (2.4.24) must also all be zero. Therefore, all coefficients in (2.4.23) must be zero, and hence W^{M+1} is a linearly independent collection, as desired. ■

This completes the proof of (2.4.6) holding for almost all C^*.

2.4.2 Recovery of Segment Lengths from Curves

Now we wish to show that for p sufficiently close to p^*, we have (2.4.7), which we repeat here for convenience:

$$\left\langle \nabla_p L_{m_3^*}(C^*, p), p^* - p \right\rangle < 0.$$

We can actually show a stronger statement, that the above is true for *all* $p \neq p^*$. Using (2.2.15) and the fact that α is linear in p, we have

$$\begin{aligned}\left\langle \nabla_p L_{m_3^*}(C^*, p), p^* - p \right\rangle &= \left\langle 2{C^*}^{\otimes 3} \star \alpha(p - p^*), {C^*}^{\otimes 3} \star \alpha(p^* - p) \right\rangle \\ &= -2\|{C^*}^{\otimes 3} \star \alpha(p - p^*)\|^2.\end{aligned}$$

Unpacking the definition of α and the $\star$ operator, the term inside the norm on the last line is equal to

$$\sum_{i=1}^{M} (p_i - p_i^*) \left(\frac{1}{12}(c_{i-1}^* + c_i^*)^{\otimes 3} + \frac{1}{6}(c_{i-1}^*)^{\otimes 3} + \frac{1}{6}(c_i^*)^{\otimes 3} \right). \tag{2.4.26}$$

It suffices to show that this is equal to zero if and only if $p_i - p_i^* = 0$ for all i. In other words, we need to show that

$$\left\{ \frac{1}{12}(c_{i-1}^* + c_i^*)^{\otimes 3} + \frac{1}{6}(c_{i-1}^*)^{\otimes 3} + \frac{1}{6}(c_i^*)^{\otimes 3} \right\}_{i=1}^{M} \tag{2.4.27}$$

is a collection of M linearly independent three-way tensors for c_i^* in generic position. We omit the proof of this because it uses techniques very similar to that of the proof of Lemma A.5.9. This completes the proof of (2.4.7) and hence the proof of Lemma 2.4.5.

2.4.3 Recovery of Curves and Segment Lengths from the Third Moment

Now we combine (2.4.6) and (2.4.7) to complete the proof of Theorem 2.4.1.

We first prove (2.4.2), which we repeat here; we wish to show that for C sufficiently close to C^* and p sufficiently close to p^*, we have

$$\left\langle \nabla_C L_{m_3^*}(C, p), C^* - C \right\rangle < 0.$$

For notational convenience, let us introduce the shorthand

$$[[C, C^*]] := \left[C \otimes C \otimes C^* + C \otimes C^* \otimes C + C^* \otimes C \otimes C - 3C^{\otimes 3}\right]. \tag{2.4.28}$$

We wish to write the expression in (2.4.2) in terms of (2.4.6) plus an error term. Using (2.2.15) and the fact $m_3^* = C^{*\otimes 3} \star \alpha(p^*)$ to compute the difference and regrouping terms, we get

$$\begin{aligned}\left\langle \nabla_C L_{m_3^*}(C,p), C^* - C \right\rangle = &\left\langle \nabla_C L_{m_3^*}(C,p^*), C^* - C \right\rangle \\ &+ 2\left\langle (C^{\otimes 3} - C^{*\otimes 3}) \star \alpha(p^*), [[C, C^*]] \star \alpha(p - p^*) \right\rangle \\ &+ 2\left\langle C^{\otimes 3} \star \alpha(p - p^*), [[C, C^*]] \star \alpha(p^*) \right\rangle \\ &+ 2\left\langle C^{\otimes 3} \star \alpha(p - p^*), [[C, C^*]] \star \alpha(p - p^*) \right\rangle.\end{aligned} \tag{2.4.29}$$

The first term on the right hand side above is precisely the term in (2.4.6), which we showed was strictly negative for C close to C^* in §2.4.1. By linearity of α, the remaining three terms on the right hand side go to zero as $p \to p^*$, and hence can be made arbitrarily small for p close to p^* by continuity. Hence, for (C, p) sufficiently close to (C^*, p^*), the entire right hand side is strictly negative, and we have (2.4.2); we denote the neighborhood around (C^*, p^*) for which (2.4.2) holds with U_C.

Now we prove (2.4.3), which we also repeat; we wish to show that for C sufficiently close to C^* and p sufficiently close to p^*, we have

$$\left\langle \nabla_p L_{m_3^*}(C,p), p^* - p \right\rangle < 0.$$

Similarly to before, now we wish to write the expression in (2.4.3) in terms of (2.4.6) plus

an error term. Again using (2.2.15) and regrouping terms, we get

$$\begin{aligned}\left\langle \nabla_p L_{m_3^*}(C,p), p^* - p \right\rangle = \left\langle \nabla_p L_{m_3^*}(C^*,p), p^* - p \right\rangle \\ + 2\left\langle (C^{\otimes 3} - C^{*\otimes 3}) \star \alpha(p), C^{\otimes 3} \star \alpha(p^*) \right\rangle \\ + 2\left\langle (C^{\otimes 3} - C^{*\otimes 3}) \star \alpha(p^*), C^{*\otimes 3} \star \alpha(p) \right\rangle \\ + 2\left\langle (C^{*\otimes 3} - C^{\otimes 3}) \star \alpha(p^*), C^{*\otimes 3} \star \alpha(p^*) \right\rangle \\ + 2\left\langle (C^{*\otimes 3} - C^{\otimes 3}) \star \alpha(p), C^{*\otimes 3} \star \alpha(p) \right\rangle \\ + 2\left\langle (C^{*\otimes 3} - C^{\otimes 3}) \star \alpha(p), C^{\otimes 3} \star \alpha(p) \right\rangle \\ + 2\left\langle (C^{\otimes 3} - C^{*\otimes 3}) \star \alpha(p), C^{*\otimes 3} \star \alpha(p^*) \right\rangle .\end{aligned} \tag{2.4.30}$$

Once again, the first term on the right hand side is precisely the term in (2.4.7), which we showed was strictly negative for all $p \neq p^*$ in §2.4.2. By continuity of the function $\cdot^{\otimes 3}$, the remaining six terms on the right hand side go to zero as $C \to C^*$, and hence can be made arbitrarily small for C close to C^*. Therefore there exists a neighborhood U_p of (C^*, p^*) where (2.4.3) holds.

To complete the proof of Theorem 2.4.1, let $U = U_C \cap U_p$; in this neighborhood, (2.4.2) and (2.4.3) hold. This completes the proof of Theorem 2.4.1, and thus the proof of local well-posedness of third moment-based curve recovery for piecewise linear curves.

2.5 A Third Moment-Based Recovery Algorithm

Now we turn our attention to devising an algorithm to recover PWL curves from data. We assume that we have noisy observations coming from (2.1.1) of a ground truth PWL curve C^* in $\mathbb{R}^d$ with M segments. We assume that $d \geq M \geq 4$. We also assume that C^* is mean zero and that the points $c_0^*, \ldots, c_M^*$ are in generic position.

The local well-posedness result in Theorem 2.4.1 suggests that a reasonable approach is to first obtain a rough estimate $\hat{C}_{\text{init}}$ with proportional segment lengths $\hat{p}_{\text{init}}$ in a neighborhood

near the ground truth C^*, then perform gradient descent in C and p on the relaxed third moment loss (2.2.14), using $\hat{C}_{\text{init}}$ and $\hat{p}_{\text{init}}$ as initializations. We propose an algorithm that does exactly this, where the initial rough estimate is obtained via the tensor power method. Note however that applying Theorem 2.4.1 requires knowledge of the noise-free third moment of a curve. We therefore need a method of estimating noise-free moments $m_k(C)$ from data.

In §2.5.1, we derive unbiased estimators of the noise-free curve moments from data. In §2.5.2, §2.5.3, and §2.5.4, we describe our recovery algorithm: in §2.5.2, we discuss how to use the tensor power method to estimate an unordered list of subspaces that the vertices of C^* live on; in §2.5.3, we discuss how to reorder these subspaces; in §2.5.4, we use these reordered subspaces to obtain initial estimates $\hat{C}_{\text{init}}$ and $\hat{p}_{\text{init}}$ as initializations for a gradient descent scheme.

2.5.1 Unbiased Estimators of Noise-Free Moments

Our task is to recover a ground truth curve C^* from observations $y^{(j)} = C^*(\tau^{(j)}) + \sigma\xi^{(j)}$, $j = 1, \ldots, N$, where $\tau \sim U[0,1]$ and $\xi \sim \mathcal{N}(0, I)$. We want a way of recovering the noise-free moments of the curve $m_k^* := m_k(C^*)$ from data. With $Y := \{y^{(1)}, \ldots, y^{(N)}\}$, we define the kth empirical moment of the data Y to be

$$\widehat{m}_k = \widehat{m}_k(Y) := \frac{1}{N}\sum_{j=1}^{N}(y^{(j)})^{\otimes k}. \tag{2.5.1}$$

First we recall expressions for the first three uncentered moments of a multivariate Gaussian.

Lemma 2.5.2. *(adapted from [33]) Let $X \sim \mathcal{N}(\mu, \Sigma)$ be a multivariate normal. Then the*

first through third uncentered moments of X are given by

$$\begin{aligned} m_1(X) &= \mu \\ m_2(X) &= \mu^{\otimes 2} + \Sigma \\ m_3(X) &= \mathrm{Sym}(\mu^{\otimes 3} + 3\mu \otimes \Sigma) \end{aligned} \tag{2.5.3}$$

Here, we use $\mu \otimes \Sigma$ to denote the three-way tensor whose jkl entry is $\mu_j \Sigma_{kl}$.

From Lemma 2.5.2, we can compute unbiased estimators for the noise-free moments.

Lemma 2.5.4. *Let C be such that $m_1(C) = 0$ and let $y \sim C(\tau) + \sigma\xi$, where $\tau \sim U[0,1]$ and $\xi \sim \mathcal{N}(0, I)$. Then*

1. *y is an unbiased estimator of $m_1(C)$;*
2. *$y^{\otimes 2} - \sigma^2 I_d$ is an unbiased estimator of $m_2(C)$;*
3. *$y^{\otimes 3}$ is an unbiased estimator of $m_3(C)$.*

Proof. For the first moment:

$$\begin{aligned} \mathbb{E}_{\tau \sim U[0,1], \xi \sim \mathcal{N}(0, I_d)}[y] &= \mathbb{E}_{\tau \sim U[0,1]}[C(\tau)] + \mathbb{E}_{\xi \sim \mathcal{N}(0, I_d)}[\sigma\xi] \\ &= \mathbb{E}_{\tau \sim U[0,1]}[C(\tau)] \\ &= m_1(C). \end{aligned} \tag{2.5.5}$$

For the second moment, eliminating cross terms by independence and the fact that C and ξ are mean zero, we can compute

$$\begin{aligned} &\mathbb{E}_{\tau \sim U[0,1], \xi \sim \mathcal{N}(0, I_d)}\left[y^{\otimes 2}\right] \\ &\quad = \mathbb{E}_{\tau}[C(\tau)^{\otimes 2}] + \mathbb{E}_{\tau,\xi}[C(\tau) \otimes \sigma\xi] + \mathbb{E}_{\tau,\xi}[\sigma\xi \otimes C(\tau)] + \mathbb{E}_{\xi}[\sigma\xi \otimes \sigma\xi] \\ &\quad = \mathbb{E}_{\tau}[C(\tau)^{\otimes 2}] + \mathbb{E}_{\xi}[\sigma\xi \otimes \sigma\xi] \\ &\quad = m_2(C) + \sigma^2 I_d. \end{aligned} \tag{2.5.6}$$

For the third moment, eliminating terms with the same idea, we have

$$\begin{aligned}
&\mathbb{E}_{\tau\sim U[0,1],\xi\sim\mathcal{N}(0,I_d)}\left[y^{\otimes 3}\right] \\
&\quad= \mathbb{E}_{\tau}[C(\tau)^{\otimes 3}] + \mathbb{E}_{\tau,\xi}[C(\tau)\otimes\sigma\xi\otimes\sigma\xi] + \mathbb{E}_{\tau,\xi}[C(\tau)\otimes C(\tau)\otimes\sigma\xi] \\
&\qquad\quad + \mathbb{E}_{\tau,\xi}[\sigma\xi\otimes C(\tau)\otimes C(\tau)] + \mathbb{E}_{\tau,\xi}[C(\tau)\otimes C(\tau)\otimes\sigma\xi] \\
&\qquad\quad + \mathbb{E}_{\tau,\xi}[C(\tau)\otimes\sigma\xi\otimes C(\tau)] + \mathbb{E}_{\xi}[(\sigma\xi)^{\otimes 3}] \\
&\quad= \mathbb{E}_{\tau}[C(\tau)^{\otimes 3}] + \mathbb{E}_{\xi}[(\sigma\xi)^{\otimes 3}] \\
&\quad= m_3(C).
\end{aligned} \tag{2.5.7}$$

■

Lemma 2.5.4 and the law of large numbers imply that the empirical moments $\widehat{m}_1$, $\widehat{m}_2 - \sigma_2 I$, and $\widehat{m}_3$ can estimate the noise-free moments m_1^*, m_2^*, and m_3^* arbitrarily well (in probability) with a sufficient number of points as long as the noise level σ is known. Therefore, we will design our algorithm for curve recovery from data assuming that we have estimates $\widehat{m}_1$, $\widehat{m}_2$, and $\widehat{m}_3$ from (2.5.1) that are arbitrarily close to the ground truth moments m_1^*, m_2^*, and m_3^* of C^*.

Note that for y coming from a noisy curve with noise level σ, the entries of $y^{\otimes k}$ have variance on the order of σ^{2k}, so by Chebyshev's inequality, the number of points needed to estimate the first three moments is order σ^6. A third moment-based recovery algorithm thus shows that the sample complexity lower bound in Theorem 2.3.7 is asymptotically tight.

2.5.2 Recovering the Subspaces of the Curve with the Tensor Power Method

Let T^3 be a three-way tensor of the form

$$T^3 = \sum_{i=1}^{r} \lambda_i u_i^{\otimes 3}, \tag{2.5.8}$$

where the vectors u_i are orthonormal and the scalars λ_i are all positive; such tensors are called *orthogonally decomposable*. The tensor power method (TPM) is a robust iterative algorithm that is able to recover $\{\lambda_i, u_i\}_{i=1}^r$, up to some permutation of the indices. For u_i that are unit length and independent (but not necessarily orthonormal), a whitening preprocessing step can be applied to reduce the problem to orthogonally decomposable case through an invertible linear transformation. A thorough introduction to the tensor power method as well as robustness guarantees can be found in [34]; we will take the TPM for granted in this chapter. We summarize the TPM with whitening as presented in [34] in Algorithm 1. Note that for a (d, d, d)-shaped tensor T^3 and (d, r) matrix W, we use $T^3(W, W, W)$ to denote the tensor contraction of the three indices of T^3 with the first index of W, i.e.,

$$[T^3(W, W, W)]_{mno} = \sum_{jkl} T^3_{jkl} W_{jm} W_{kn} W_{lo}. \tag{2.5.9}$$

We use the TPM in our algorithm in the following way. Recall from (2.2.6) that the third moment tensor of a ground truth PWL curve C^* is given by

$$m_3^* = \frac{1}{12} \sum_{i=1}^{M} \frac{\|c_i^* - c_{i-1}^*\|}{Z} \left((c_{i-1}^* + c_i^*)^{\otimes 3} + 2{c_{i-1}^*}^{\otimes 3} + 2{c_i^*}^{\otimes 3} \right). \tag{2.5.10}$$

If we heuristically assume we can throw away the cross terms from $(c_{i-1}^* + c_i^*)^{\otimes 3}$, then this becomes a tensor of the form

$$\sum_{i=0}^{M} \lambda_i {c_i^*}^{\otimes 3}. \tag{2.5.11}$$

We conjecture that the TPM with whitening applied to $\widehat{m}_3$ will recover unit vectors $\{\tilde{c}_j\}_{j=0}^M$, such that after reordering, $\tilde{c}_{j_i}$ approximately spans the one-dimensional subspace that c_i^* lives in, that is, $\tilde{c}_{j_i}$ is close to c_i^* in cosine similarity.① Throughout this subsection, we will use $\tilde{c}_j$ to refer to the unsorted unit vectors and $\tilde{c}_{j_i}$ to refer to the sorted unit vectors.

Note that (2.5.8) is invariant to a reordering of the λ_i's and u_i's, so the fact that the

① The *cosine similarity* of two vectors a and b is $\langle a, b \rangle / (\|a\| \|b\|)$.

Algorithm 1: Tensor Power Method with Whitening: `TPM`

Input: Symmetric three-way tensor $T^3 = \sum_{i=1}^{r} \lambda_i u_i^{\otimes 3}$ with $u_i \in \mathbb{R}^d$ unit length and linearly independent and $\lambda_i > 0$; rank r symmetric matrix $T^2 = \sum_{i=1}^{r} \lambda_i u_i^{\otimes 2}$; desired number of additive factors $r \leq d$.

Output: Estimates $\{\hat{u}_i\}_{i=1}^{r}$ of $\{u_i\}_{i=1}^{r}$.

`/* WHITENING STEP */`

Set U to be the $\mathbb{R}^{d \times r}$-shaped matrix whose columns are orthonormal eigenvectors of T_2 corresponding to nonzero eigenvectors.

Set D to be the $\mathbb{R}^{r \times r}$-shaped diagonal matrix of positive eigenvalues of T_2.

Compute the whitening matrix $W = UD^{-1/2} \in \mathbb{R}^{d \times r}$.

Whiten the T^3 to an orthogonally decomposable tensor by setting $\tilde{T}^3 = T^3(W, W, W) \in \mathbb{R}^{r \times r \times r}$.

`/* TENSOR POWER ITERATION */`

for $i = 1, \ldots, r$ **do**

  Randomly initialize $x \in \mathbb{R}^d$.

  while *not converged* **do**

    Iterate $x \leftarrow \sum_{jk} \tilde{T}^3_{jkl} x_k x_l / \| \sum_{jk} \tilde{T}^3_{jkl} x_k x_l \|$.

  end

  Set $\tilde{\lambda}_i = \sum_{jkl} \tilde{T}^3_{jkl} x_j x_k x_l$.

  Set $\tilde{u}_i = x$.

  Set $\hat{u}_i = \tilde{\lambda}_i (W^T)^{\dagger} \tilde{u}_i$, where $\dagger$ denotes the Moore-Penrose pseudoinverse.

  Deflate the whitened tensor: $\tilde{T}^3 \leftarrow \tilde{T}^3 - \tilde{\lambda}_i \tilde{u}_i^{\otimes 3}$.

end

TPM recovers u_i up to permutation is not important to the tensor decomposition problem. However, to recover the vertices of a curve, we do in fact need to recover the ordering of the c_i^*'s. Therefore, we need some way to reorder the putative subspaces returned by the TPM.

2.5.3 Reordering the Predicted Subspaces

The tensor power moment returns unit vectors $\{\tilde{c}_j\}_{j=0}^M$ that are ostensibly close in alignment to the ground truth points c_i^*; we need to reorder them so that $\tilde{c}_{j_i}$ is close to c_i^*, where we use the term "close" here (and in the rest of this subsection) in the sense of cosine similarity. The seriation problem of ordering time dependent data is well-studied and there are many existing approaches (see, for instance, [27, 29]), but here we outline a heuristic approach that empirically performs well for our purposes.

A reasonable prior is that our curve is sufficiently smooth such that if $i \neq 0, M$, then the two points closest to c_i^* are c_{i-1}^* and c_{i+1}^*. This is the inspiration behind Algorithm 2, where we order the subspaces (given an initial choice) by iteratively choosing the closest subspace from $\{\tilde{c}_j\}_{j=0}^M$ that has not been sorted yet.

Algorithm 2 can output up to $M+1$ different orderings depending on the choice of initial index j_0. To see which ordering to choose, consider the simple case illustrated in Figure 2.3. Here we assume that we have five unit vectors along some arc. Then the five possible orderings output by Algorithm 2, depending on the choice of initial index, are

1. c_0, c_1, c_2, c_3, c_4
2. c_1, c_0, c_2, c_3, c_4
3. c_2, c_3, c_4, c_1, c_0
4. c_3, c_2, c_1, c_0, c_4
5. c_4, c_3, c_2, c_1, c_0.

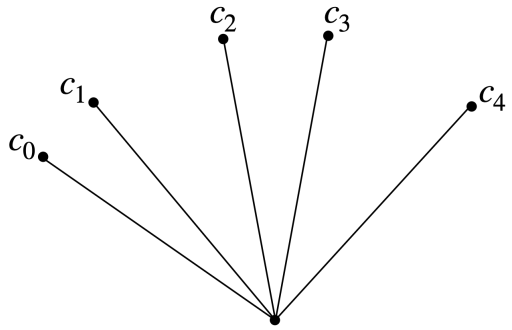

Figure 2.3: An illustration of a toy ideal case for Algorithm 2.

Notice that the two correct sortings are (1) and (5), and that these two sortings are palindromes of another. A reasonable heuristic then is to assume that C^* does not deviate too much from this ideal situation and $\{\tilde{c}_j\}$ do not deviate too much from the true subspaces, build up the $M+1$ chains starting from each of the predicted subspaces, and choose the chain that has a palindrome.

Of course, in general we are not guaranteed that a palindrome pair will exist among the $M+1$ sortings generated by Algorithm 2, so we instead need a way to measure how "palindromic" a particular sorting is given a list of sortings.

Definition 2.5.12. Let $\Pi = \{\pi_i\}$ be a set of permutations of $M+1$ elements, where we identify permutations with ordered tuples of length $M+1$ with exactly one of $\{0, \dots M\}$ in each entry. Let $\mathrm{Rev}(\pi)$ be the reversed tuple of π. Given $\pi_1, \pi_2 \in \Pi$, we define the palindrome similarity $\mathrm{psim}(\pi_1, \pi_2)$ between them as the number of entries in which π_1 and $\mathrm{Rev}(\pi_2)$ agree. For example, $\mathrm{psim}((0,1,2,3,4),(2,3,4,1,0)) = 3$. The *palindromicity* $\mathscr{P}_\Pi(\pi)$ of a permutation $\pi \in \Pi$ is defined as the maximum palindrome similarity between π and every other permutation in Π:

$$\mathscr{P}_\Pi(\pi) = \max_{\nu \in \Pi} \mathrm{psim}(\pi, \nu). \tag{2.5.13}$$

Note that if the palindromicity of a permutation in a list is equal to $M+1$, that means the palindrome of the permutation exists in the list. From our list of $M+1$ orderings generated by the chain-building algorithm, we then choose the one with the highest palindromicity,

breaking ties arbitrarily. In the ideal case, there should be exactly one pair of orderings with palindromicity equal to $M + 1$.

2.5.4 Final Optimization Steps

From the output of Algorithm 1 sorted by Algorithm 2, we now have a row-stacked matrix $\tilde{C} = [\tilde{c}_{j_0}, \ldots \tilde{c}_{j_M}]^T$ of unit vectors that approximately span the subspaces in which c_i^* lie. Since $\tilde{c}_{j_i}$ are unit length, we now wish to find scalars λ_i such that the curve with vertices $\lambda_i \tilde{c}_{j_i}$ has moments close to the estimated moments $\widehat{m}_1, \widehat{m}_2, \widehat{m}_3$. Observe that $\mathrm{diag}(\lambda)\tilde{C}$ is the matrix whose ith row is given by $\lambda_i \tilde{c}_{j_i}$. Then we choose λ to minimize the loss of the first three moments of $\mathrm{diag}(\lambda)\tilde{C}$:

$$\hat{\lambda} = \mathrm{argmin}_\lambda \left[\sum_{k=1}^{3} \left\| m_k(\mathrm{diag}(\lambda)\tilde{C}) - \widehat{m}_k \right\|_F^2 \right] \tag{2.5.14}$$

If the $\tilde{c}_{j_i}$ are close in alignment to c_i^*, then $\mathrm{diag}(\hat{\lambda})\tilde{C} =: \hat{C}_{\text{init}}$ should be close to the ground truth C^*. If this is close enough, then it will be within the neighborhood of C^* where the third moment minimizer is unique (the existence of such a neighborhood is precisely the statement of Corollary 2.4.4). We call this first step the initial estimate phase.

With $\hat{\lambda}$ determined, we now use $\hat{C}_{\text{init}}$ as an initialization for gradient descent on $L_{m_3^*}(C, p)$ as defined in (2.2.14). In practice, as mentioned in §2.2, since C^* naturally lives in an M-dimensional subspace V of $\mathbb{R}^d$, we first project $\hat{C}_{\text{init}}$ down to the subspace spanned by the top M eigenvectors of $\widehat{m}_2$; let Q be the (d, M)-shaped matrix whose columns are these eigenvectors. Note that if we replace the estimate $\widehat{m}_2$ with the exact second moment m_2^* of C^*, then m_2^* has exactly M nonzero eigenvectors. Then the projection of $\hat{C}_{\text{init}}$ onto V can be estimated by $\hat{C}_{\text{init}}Q$ (the estimation is exact when Q is computed from m_2^* rather than $\widehat{m}_2$).

Since we wish to match the third moment in the subspace V, we need to estimate the third moment of the projected curve. The exact third moment of the projected curve is

given by $m_3^*(Q,Q,Q)$ (see (2.5.9)), and hence we can estimate the projected third moment from data by $\widehat{m}_3(Q,Q,Q)$. We then use $\hat{C}_{\text{init}}Q$ and its corresponding proportional segment lengths $\hat{p}_{\text{init}}$ as an initial point for minimizing $L_{\widehat{m}_3(Q,Q,Q)}(C,p)$ via alternating gradient descent, then deproject the result back to $\mathbb{R}^d$ to get our final prediction $\hat{C}$. We call this second step consisting of projection, alternating descent, and deprojection the *finetuning phase*.

To summarize, our recovery algorithm is in two phases. The first initial estimate phase uses the TPM on the third moment tensor to approximately recover the unordered subspaces in which the points c_i approximately lie; these points are then sorted using the chain-building algorithm, and the sorting with the best palindromicity is chosen. We then find the best $\hat{c}_i$ along these subspaces that minimize the loss of the first three moments. In the second finetuning phase, we use these $\hat{c}_i^*$ as an initial guess for minimizing the loss of the relaxed third moment in the M-dimensional subspace where C^* naturally lies. The full algorithm is presented as Algorithm 3.

Algorithm 2: Chain-Building Algorithm: `ChainBuild`

Input: Unordered approximate subspaces $\{\tilde{c}_j\}_{j=0}^{M}$; initial index $0 \le j_0 \le M$.

Output: Putative ordering of the approximate subspaces $[j_0, \dots j_M]$.

Initialize the chain with the initial index: `chain` $\leftarrow [j_0]$.

Remove the initial index from the list of indices of unsorted subspaces: `RemainingSubspaces` $\leftarrow [0, \dots, M] \setminus j_0$.

for $k = 1, \dots, M$ **do**

  Set j_k to be the index in `RemainingSubspaces` such that $\tilde{c}_j$ is closest to $\tilde{C}_{j_{k-1}}$ in cosine similarity.

  Append j_k to the chain: `chain` $\leftarrow [j_0, \dots, j_{k-1}, j_k]$.

  Remove j_k from the list of unsorted subspaces: `RemainingSubspaces` $\leftarrow$ `RemainingSubspaces` $\setminus j_k$.

end

Algorithm 3: Full Curve Recovery Algorithm

Input: Number of segments M; estimated moments $\widehat{m}_1, \widehat{m}_2, \widehat{m}_3$ of a PWL curve in $\mathbb{R}^d$ where $d \ge M$.

Output: Vertices of predicted PWL curve $\hat{C}$ in $\mathbb{R}^d$ with M segments.

`/* FIRST PHASE, INITIAL ESTIMATE */`

Use the tensor power method to estimate the unordered subspaces in which the vertices of C^*: $\tilde{C}_{\text{unsorted}} := \{\tilde{c}_0, \dots, \tilde{c}_M\} = \texttt{TPM}(\widehat{m}_3, \widehat{m}_2, M)$.

Build all possible putative sortings of the unsorted subspaces: $\Pi = \{\texttt{ChainBuild}(\{\tilde{c}_j\}_{j=0}^{M}, i)\}_{i=0}^{M}$.

Choose the most palindromic sorting in Π: $\hat{\pi} = \operatorname{argmax}_\pi \mathscr{P}_\Pi(\pi)$ (see (2.5.13)); break ties arbitrarily.

Sort $\tilde{C}_{\text{unsorted}}$ using $\hat{\pi}$: $\tilde{C} = [\tilde{c}_{\hat{\pi}(0)}, \dots, \tilde{c}_{\hat{\pi}(M)}]$.

Find $\hat{\lambda} \in \mathbb{R}^{M+1}$ that minimizes (2.5.14) via gradient descent: $\hat{\lambda} = \operatorname{argmin}_\lambda \left[\sum_{k=1}^{3} \left\| m_k(\operatorname{diag}(\lambda)\tilde{C}) - \widehat{m}_k \right\|_F^2\right]$.

Compute the initial guess for the curve vertices along the subspaces spanned by $\{\tilde{c}_i\}$: $\hat{C}_{\text{init}} = \operatorname{diag}(\hat{\lambda})\tilde{C}$.

`/* SECOND PHASE, FINETUNING */`

Set Q to be the (d, M)-shaped matrix whose columns are the eigenvectors corresponding to the top M eigenvalues of $\widehat{m}_2$.

Project the initial projection to the subspaces spanned by the columns of Q: $\hat{C}_{\text{proj init}} = \hat{C}_{\text{init}} Q$.

Set $\hat{p}_{\text{proj init}}$ as the proportional segment lengths of $\hat{C}_{\text{proj init}}$.

Compute the third moment of the projected curve: $(\widehat{m}_3)_{\text{proj}} = \widehat{m}_3(Q, Q, Q)$.

Solve $(\hat{C}_{\text{proj}}, \hat{p}_{\text{proj}}) = \operatorname{argmin}_{C,p} L_{(\widehat{m}_3)_{\text{proj}}}(C, p)$ via alternating gradient descent on C and p with $(\hat{C}_{\text{proj init}}, \hat{p}_{\text{proj init}})$ as initialization.

Deproject the final projected prediction to the ambient space: $\hat{C} = \hat{C}_{\text{proj}} Q^T$.

2.6 Numerical Results

Here we present some numerical results on using Algorithm 3 to recover curves. Our implementation is primarily built with the the libraries JAX and JAXopt ([35, 36]).

To generate our data, we first prescribe M segment lengths Z_i uniformly randomly from the interval $[1, 2]$. Then we sample a random curve in $\mathbb{R}^d$ with M segments by choosing the origin as the initial point, then iteratively adding points by choosing a random vector on the sphere with radius L_i and adding it to the most recently generated point on the curve. We then subtract away the first moment of the curve to ensure that our curve is mean zero. We do not normalize the segment lengths to fit the curve into the unit ball (as in the hypotheses for Proposition 2.3.5) for numerical stability reasons.

Recall Algorithm 3 relies on having access to the noise-free moments of the ground truth curve C^*. For high noise, a very large number (order σ^6) of data points from a noisy cloud are required to closely estimate the first three moments of the noise-free underlying curve. In high dimensions, this becomes enormously computationally expensive. Therefore, we show results from two different experiments: one where we apply Algorithm 3 directly on ground truth noise-free moments that we obtain from C^* (i.e., we set $\widehat{m}_k = m_k^*$), and another experiment that uses empirical moments computed from a point cloud generated from C^* (i.e. we estimate $\widehat{m}_k$ from data). Due to the large number of points needed to accurately estimate the third moment tensor, we compute the moments in an online manner so that we never have to store an entire point cloud in memory.

Algorithm 3 involves an initial phase that estimates a curve close to the ground truth before a second phase that directly minimizes the relaxed third moment loss. We propose two similar baseline algorithms to show that our initial rough estimation phase is essential: we simply directly minimize either the squared Frobenius loss of the third moment or the sum of the losses of the first, second and third moments from multiple random initializations, as detailed in Algorithm 4. We perform the optimization in an estimation of the natural M-dimensional subspace V where the curve lies, so we minimize against the moments of

the projected curve. As before, let Q be the (d, M)-shaped matrix whose columns are the top M eigenvectors of $\widehat{m}_2$; when $\widehat{m}_2$ equals m_2^* exactly, the columns of Q exactly span V. Then the first three moments of the projected curve can be estimated by $(\widehat{m}_1)_{\text{proj}} = Q^T \widehat{m}_1$, $(\widehat{m}_2)_{\text{proj}} = Q^T \widehat{m}_2 Q$, and $(\widehat{m}_3)_{\text{proj}} = \widehat{m}_3(Q, Q, Q)$. As in Algorithm 3, the minimizations are performed with gradient descent. We perform the minimization with $n = 10$ random initializations and choose the result with the best final moment loss.

Algorithm 4: Baseline Recovery Algorithm with Random Initialization for Moment Matching

Input: Number of segments M; estimated moments $\widehat{m}_1, \widehat{m}_2, \widehat{m}_3$ of a PWL curve in $\mathbb{R}^d$ where $d \geq M$, number of random baseline initializations n.

Output: Vertices of predicted PWL curve $\hat{C}$ in $\mathbb{R}^d$ with M segments.

Use eigenvalue decomposition of $\widehat{m}_2$ to compute the (d, M)-shaped matrix Q whose columns are the top M eigenectors of $\widehat{m}_2$.

Estimate the moments of the projected curve: $(\widehat{m}_1)_{\text{proj}} = Q^T \widehat{m}_1$, $(\widehat{m}_2)_{\text{proj}} = Q^T \widehat{m}_2 Q$, $(\widehat{m}_3)_{\text{proj}} = \widehat{m}_3(Q, Q, Q)$.

Randomly initialize the running best predicted curve $\hat{C}_{\text{proj best}}$ and set the running best loss from n trials: $\text{loss}_{\text{best}} \leftarrow +\infty$.

Define a loss function from matching just the third moment or the all of first three moments: $L(C) := \sum_{k=1}^{3} \|m_k(C) - \widehat{m}_k\|_F^2$ or $L(C) := \|m_3(C) - \widehat{m}_3\|_F^2$ (as in (2.2.14)).

`/* iteratively update best estimate */`

for $k = 1, \ldots, n$ **do**

  Randomly initialize $\hat{C}_{\text{proj init}}$ with shape $(M + 1, d)$.

  Solve $\hat{C}_{\text{proj}} = \text{argmin}_C \, L(C)$ via gradient descent with $\hat{C}_{\text{proj init}}$ as initialization.

  if $L(\hat{C}_{\text{proj}}) < \text{loss}_{\text{best}}$ **then**

    Update the best loss: $\text{loss}_{\text{best}} \leftarrow L(\hat{C}_{\text{proj}})$.

    Update the best projected curve: $\hat{C}_{\text{proj best}} \leftarrow \hat{C}_{\text{proj}}$.

  end

end

Deproject the best projected curve: $\hat{C} = \hat{C}_{\text{proj best}} Q^T$.

In Table 2.1, we present results from the experiment with moments $\widehat{m}_k$ estimated from a point cloud with $N = 10^8$ points, with $M = 16$ segments, ambient dimension $d = 24$, and noise level $\sigma^2 = 1/4$. We report the lower quartile, median, and upper quartile of the square root of the curve loss $\sqrt{\rho(C^*, \hat{C})}$ (see (2.3.2)) and the relative third moment loss

$\|m_3^* - m_3(\hat{C})\|_F / \|m_3^*\|_F$ between the ground truth and predicted curves over 500 trials with different random ground truth curves C. We present the results for the outputs of both phases of Algorithm 3, as well as the performance of both baselines with the best result out of $n = 10$ random initializations. In Table 2.2, we present the same statistics for the experiment with access to ground truth moments with $M = 32$, and $d = 48$ (we are able to perform the experiment in higher dimensions because we no longer have to generate and compute moments of a large point cloud, which is the primary performance bottleneck). We see that in both experiments, Algorithm 3 significantly outperforms both baselines in the curve loss, even without the finetuning phase. Observe also that the actual third moment loss is lower for the baseline algorithms even though the curve loss is higher; this indicates that the baseline algorithms are converging to a curve different from the ground truth C^*. In the experiment with access to ground truth moments, the sorting of subspaces was successful $488/500 = 97.6\%$ times, where we define success as there being exactly one palindromic pair of sorted subspaces. In the experiment with moments estimated from a point cloud, the success rate was $449/500 = 89.8\%$. We show some predicted curves from Algorithm 3 for both experiments in Figures 2.4 and 2.5.

2.7 Conclusion and Future Work

In this chapter, we consider the problem of resolving a PWL constant speed curve in high dimensions from a high noise point cloud around the curve. We show that for sufficiently large noise, any algorithm hoping to recover the underlying curve with high probability will require at least $O(\sigma^6)$ samples, giving an asymptotic lower bound on the sample complexity. We then show that this lower bound is asymptotically tight by providing a recovery algorithm based on the third moment tensor, exploiting the fact that the third moment uniquely determines a curve in some neighborhood. Further work is needed in showing theoretical guarantees for our algorithm, e.g., showing robustness in the presence of error in the estimation of the curve moments $m_k(C)$ from the data, or guarantees on Algorithm 2. There is also work to be done

in recovery of curves other than PWL curves, as well as investigating the phase transition between the low noise and high noise recovery problems.

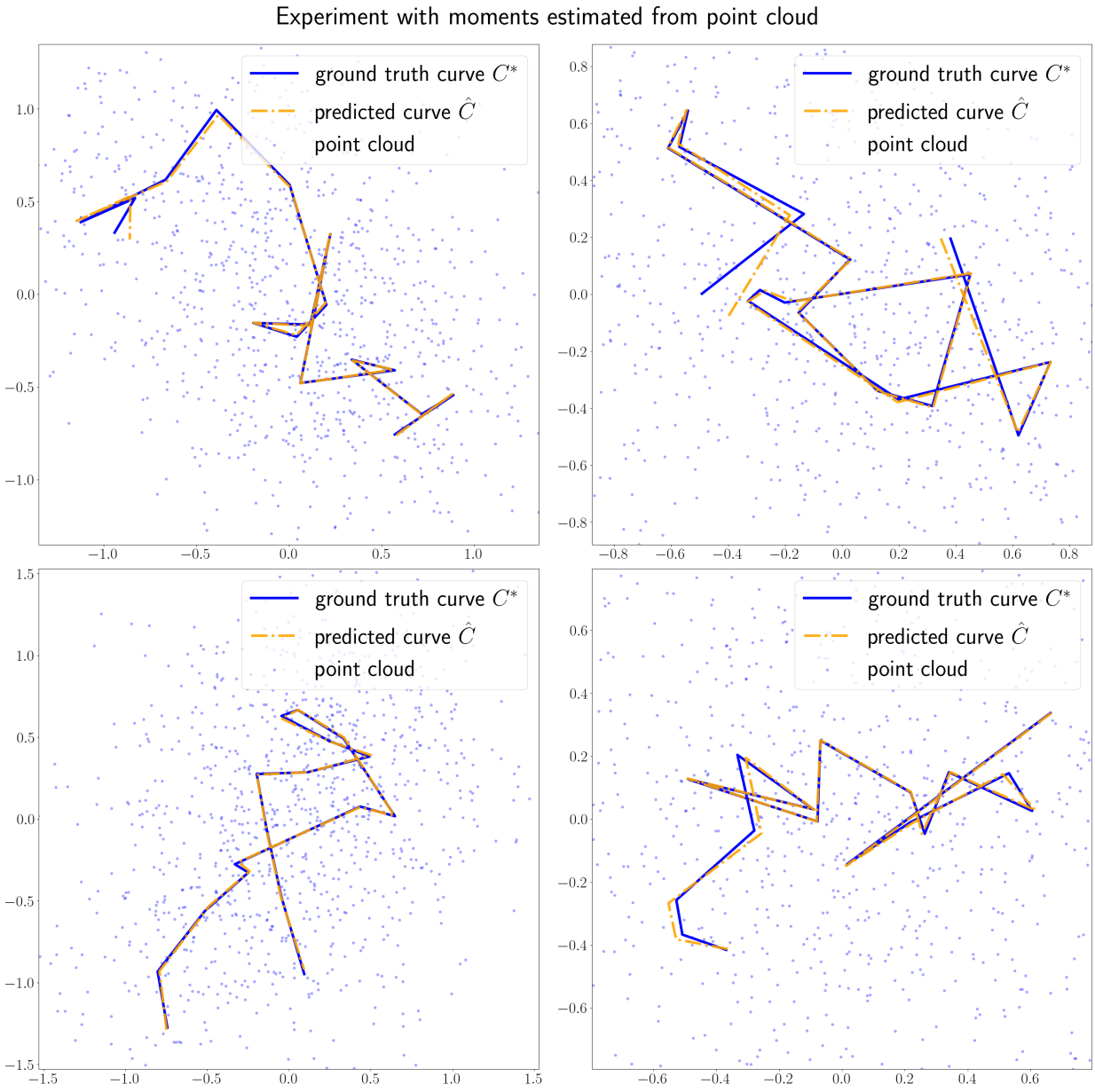

Figure 2.4: Four uncurated experimental results for curves with $M = 16$ segments, ambient dimension $d = 24$, noise level $\sigma^2 = 1/4$ and moments estimated from point cloud with $N = 10^8$ points. We show the ground truth curve and a subsample of 1000 points from the point cloud in blue. The output of Algorithm 3 is given by the dashed orange line. All curves are projected down to the first two dimensions of $\mathbb{R}^{24}$

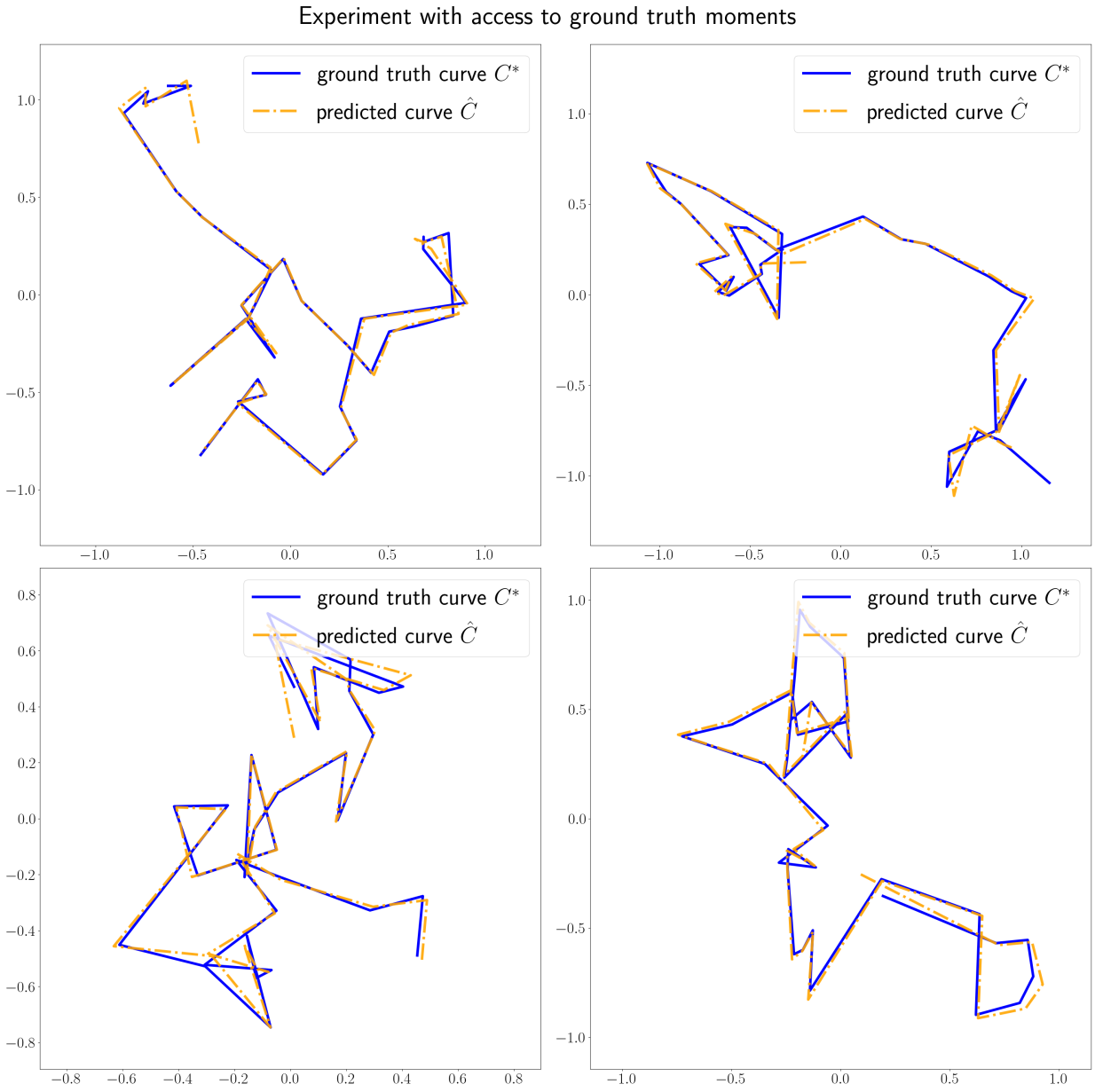

Figure 2.5: Four uncurated experimental results for curves with $M = 32$ segments, ambient dimension $d = 48$, and access to ground truth moments. We show the ground truth curve in blue. The output of Algorithm 3 is given by the dashed orange line. All curves are projected down to the first two dimensions of $\mathbb{R}^{48}$

Table 2.1: Lower quartile, median, and upper quartile root curve losses $\sqrt{\rho(C^*, \hat{C})}$ (see (2.3.2)) and relative third moment losses $\|m_3^* - m_3(\hat{C})\|_F / \|m_3^*\|_F$ over 500 trials with moments estimated from point cloud with $N = 10^8$ points, $M = 16$ segments, ambient dimension $d = 24$, noise level $\sigma^2 = 1/4$. The lowest value in each column is bolded.

	root curve loss			**rel. third moment loss**		
method	**25%**	**50%**	**75%**	**25%**	**50%**	**75%**
Algorithm 3 after first phase	0.4576	0.4866	0.5431	0.0513	0.0574	0.0658
Algorithm 3 after second phase	**0.2271**	**0.2627**	**0.3157**	0.0578	0.0776	0.1141
Baseline algorithm with third moment only	1.5947	1.6740	1.7444	0.0172	0.0201	0.0243
Baseline algorithm with all three moments	1.5681	1.6505	1.7130	**0.0134**	**0.0152**	**0.0171**

Table 2.2: Lower quartile, median, and upper quartile root curve losses $\sqrt{\rho(C^*, \hat{C})}$ (see (2.3.2)) and relative third moment losses $\|m_3^* - m_3(\hat{C})\|_F / \|m_3^*\|_F$ over 500 trials with access to ground truth moments, $M = 32$ segments, ambient dimension $d = 48$. The lowest value in each column is bolded.

	root curve loss			**rel. third moment loss**		
method	**25%**	**50%**	**75%**	**25%**	**50%**	**75%**
Algorithm 3 after first phase	0.6784	0.7977	**0.9874**	0.0415	0.0630	0.1091
Algorithm 3 after second phase	**0.4532**	**0.6389**	1.0519	0.0672	0.1023	0.2073
Baseline algorithm with third moment only	1.9862	2.0539	2.1163	0.0152	0.0176	0.0204
Baseline algorithm with all three moments	1.9850	2.0462	2.1079	**0.0115**	**0.0128**	**0.0138**

CHAPTER 3

FAST DISCRETE OPTIMAL TRANSPORT ON SPIN SYSTEMS

Optimal transport is one of the most well-studied problems in applied mathematics; the following very brief historical sketch is taken from [37]. The problem was first studied in Gaspard Monge's 1781 work *Sur la théorie des déblais et des remblais*, in which he studied the problem of optimally transporting earth from mines (or other sources in the ground) to some other location. In the 1940s, Kantorovich rediscovered the problem, albeit with a somewhat different formulation, and furthermore framed it as a linear programming problem, thus giving a method to actually compute solutions to the optimal transport problem. The 21st century has seen a massive acceleration in interest in optimal transport, as evidenced by Cédric Villani's monumental reference [37], as well as Filippo Santambrogio more applied perspective [38]. While most work on optimal transport has been for continuous domains, we are interested in optimal transport on a discrete domain; see [39] for a brief survey on techniques in discrete optimal transport.

In search of a discrete setting in which to do optimal transport, we turn to the Ising model. The Ising model is one of the most foundational and well-studied models in statistical physics [40], consisting of a lattice of interacting spins that can assume one of two states. The Glauber model, first introduced in [41], is a dynamical system on the Ising model, where at each step, a transition rate determines the probabilities of spins changing state. In the Glauber model, the transition rates for a particular site have a particular form that depends on the spin states of the site's neighbors. We instead consider a more relaxed problem with very few restrictions on the transition rates. From these Glauber-like dynamics on spin systems, we will derive a high dimensional optimal transport problem, then implement a fast solver that is able to solve problem instances up to dimension one billion.

This chapter is organized as follows. In §3.1.1, we introduce and set up the spin system in which we will be doing optimal transport. In §3.1.2, we describe the Glauber-like dynamics we impose on the spin system and connect this to optimal transport. In §3.1.3, we give

a formal statement of the problem. We formulate this as a standard, albeit exponentially large, linear program in §3.2.1 and discuss a practical method to solve large instances of this linear program in §3.2.2, followed by an accelerated mean-field approximation approach in §3.2.3. We show some results of numerical experiments in §3.3.

3.1 Introduction

3.1.1 Notation and Introduction to the Problem

Consider a system of n spins, each of which may assume a state of ± 1; this gives rise to a state space $\Omega = \Omega_n := \{\pm 1\}^n$ of size 2^n; we call elements $\sigma \in \Omega$ *configurations*. Observe that by assigning an arbitrary ordering to the spins, there exists a natural bijection between Ω and the binary representations of all integers $0, 1, \ldots, 2^n - 1$; for example, for the case $n = 5$, we can associate the decimal integer 22, with binary representation 10110, with the configuration where sites 1 and 4 are in state -1, and sites 2, 3, and 5 are in state $+1$ (we let the rightmost bit of the binary representation denote the first of n states, and the leftmost bit denote the last of n states). For any configuration σ and integer $j = 1, \ldots, n$, let σ^j denote the configuration that is identical to σ, but with the jth spin flipped; for example, if $\sigma = 10110$ and $j = 2$, then $\sigma^j = 10100$.

We define dynamics on these configurations as follows. Consider a temporal sequence of configurations $\sigma_0, \sigma_1, \ldots, \sigma_T$ representing the trajectory of the evolution of some initial configuration σ_0. At each of the T evolution steps, we only allow for one of two possible transitions: either the configuration remains unchanged, or exactly one spin site flips; we call the configurations that differ at exactly one spin site from σ the configurations that are *reachable* from σ.

Let $v(\sigma, j, t)$ denote the probability that at time step t, the jth spin of the configuration σ flips; we can thus consider $v_t = v(\cdot, \cdot, t)$ as a matrix of shape $(2^n, n)$ that controls the transition probabilities between all possible configurations at each time step t; we hence

call v_t the *control at time t*. Note that the entries of v_t must be nonnegative, and the row sums must be ≤ 1, since each row encodes the probabilities of a particular configuration transitioning to each of its reachable configurations. Finally, let $\rho(\sigma, t)$ denote the probability of observing a configuration σ at time step t; then $\rho_t = \rho(\cdot, t)$ is a probability vector of size 2^n over Ω. Note that a transition v_t induces an evolution of a distribution ρ_t to another distribution ρ_{t+1}.

We can write down this evolution explicitly. Note that if the configuration σ is at time $t+1$, either (a) the configuration σ was observed at time t and the configuration remained unchanged, or (b) the configuration σ^j was observed at time t for some $j = 1 \ldots, n$ and the jth bit was flipped in the transition. We can write this down as:

$$\rho(\sigma, t+1) = \rho(\sigma, t) \left(1 - \sum_{j=1}^{n} v(\sigma, j, t)\right) + \sum_{j=1}^{n} \rho(\sigma^j, t) v(\sigma^j, j, t). \tag{3.1.1}$$

In statistical physics contexts this is sometimes called the master equation in the Glauber model [42, 43]; we will henceforth refer to it as the *evolution equation* of our model. Our evolution differs from the standard Glauber dynamics in that we have few constraints on the transition probabilities v. Typically, the spins are arranged in some graph, and $v(\sigma, j, t)$ must be a function of the neighbors of spin site j of a particular form. Our dynamics are thus a considerable relaxation of the classical Glauber dynamics.

Oftentimes, (3.1.1) is written with the notation

$$\frac{\partial}{\partial t}\rho(\sigma, t) = -\sum_{j=1}^{n} \rho(\sigma, t) v(\sigma, j, t) + \sum_{j=1}^{n} \rho(\sigma^j, t) v(\sigma^j, j, t). \tag{3.1.2}$$

From (3.1.1) and (3.1.2), we have

$$\rho(\sigma, t) + \frac{\partial}{\partial t}\rho(\sigma, t) = \rho(\sigma, t+1),$$

justifying the derivative notation in (3.1.2).

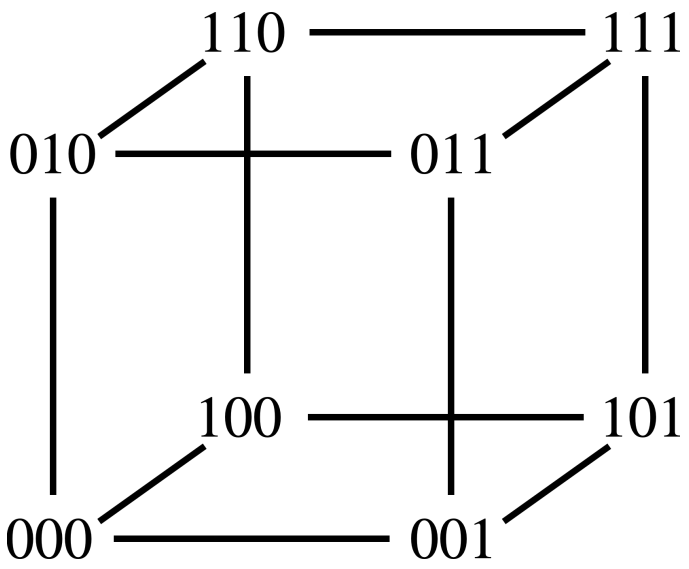

Figure 3.1: The Hamming cube representation for the configurations of a spin system with $n = 3$ spins.

The binary representation of the configurations and our transition rules give rise to another natural representation of Ω as the hypercube graph Q_n. We may associate each element $\sigma \in \Omega$ with a vertex of the hypercube graph, where edges are drawn between configurations whose binary representations have a Hamming distance of exactly 1 (sometimes this is referred to as the *Hamming cube*, see Figure 3.1); note that these edges are exactly those which connect configurations that are reachable from each other. We can then interpret ρ_t as a distribution of mass on the vertices of Q_n.

We wish to consider the following problem: given initial and final probability distributions $\mu = \rho_0$ and $\nu = \rho_T$ on Ω, we wish to find a sequence of controls $v_0, \ldots, v_{T-1}$ that transitions μ to ν. Furthermore, we will later introduce a cost function on the sequence of distributions ρ_t and controls v_t that we wish to minimize. This is therefore a discrete time optimal control problem, with state vectors given by the distributions ρ_t and controls given by v_t.

3.1.2 The Evolution Equation and Connection to Optimal Transport

Before further discussion of our spin system model, we first introduce the Benamou-Brenier formulation of optimal transport, with which we will later show a connection to the dynamics

of our model.

The classical Monge-Kantorovich problem in optimal transport is formulated as follows. Let X and Y be complete separable metric spaces, and $c : X \times Y \to [0, +\infty]$ be a continuous cost function. Let μ and ν be probability measures on X and Y respectively. Let $\Pi(\mu, \nu)$ denote the set of *transport plans*, i.e., the collection of measures on $X \times Y$ with X marginals μ and Y marginals ν. The Monge-Kantorovich problem is to find the *optimal transport cost* between μ and ν, given by

$$\inf \left\{ \int_{X \times Y} c \, d\gamma : \gamma \in \Pi(\mu, \nu) \right\}. \tag{3.1.3}$$

The classical interpretation of this is that we wish to move a distribution of mass μ to another distribution ν, and for measurable sets $A \in X$ and $B \in Y$, the measure $\gamma(A \times B)$ represents the amount of mass being moved from A to B. The integral in (3.1.3) is then the cost of moving the mass from X to Y.

When $X = Y$ and $c(x, y) = \|x - y\|^p$ for some $p \in [1, \infty)$, the quantity (3.1.3) is referred to as the *Wasserstein metric* and denoted $W_p(\mu, \nu)$. When we furthermore require $X = Y = \mathbb{R}^d$, we have the following fluid mechanics reinterpretation known as the *Benamou-Brenier* formulation (first formulated for W_2 in [44], then generalized to W_p in [45, 46]):

$$\begin{aligned} W_p^p(\mu, \nu) = \min_v \quad & \int_0^1 \int_{\mathbb{R}^d} \|v(x,t)\|^p \rho(x,t) \, dx \, dt \\ \text{such that} \quad & \frac{\partial}{\partial t} \rho(x,t) = -\nabla \cdot (\rho v) \\ & \rho(x,0) = \mu(x), \quad \rho(x,1) = \nu(x). \end{aligned} \tag{3.1.4}$$

Here we are considering μ and ν as probability density functions rather than measures. The function $v : \mathbb{R}^d \times [0,1] \to \mathbb{R}$ denotes some time-dependent velocity field, and $\rho(\cdot, t)$ denotes a probability density for each t. The interpretation here is that $\mu = \rho(\cdot, 0)$ and $\nu = \rho(\cdot, 1)$ are some initial and final distributions of mass, and the field v pushes μ towards ν, with intermediate states given by $\rho(\cdot, t)$. The first constraint in (3.1.4) is known as the

continuity equation; by the divergence theorem, this enforces conservation of mass of ρ, so that $\int \rho(x,t)\,dx = 1$ for all $t \in [0,1]$.

Observe that the continuity equation in (3.1.4) states that the time derivative of probability density is equal to some net flux term. Observe also that our evolution equation in the form (3.1.2) makes an analogous statement, that the time derivative of the probability density is equal to net inflow minus net outflow. Therefore we see that the dynamics of our model can be viewed as a discrete analogue of the dynamics underlying the Benamou-Brenier formulation of optimal transport.

Earlier we stated that we would assign a cost to a sequence of distributions ρ_t and controls v_t. To continue the analogy with the Benamou-Brenier formula, the straightforward choice that most closely resembles the cost in (3.1.4) is

$$\sum_{t=0}^{T-1} \sum_{\sigma \in \Omega} \sum_{j=1}^{n} v(\sigma, j, t)^p \rho(\sigma, t). \tag{3.1.5}$$

3.1.3 Problem Formulation: Optimal Transport on Spin Systems

We are now ready to give a precise statement of our discrete optimal transport problem on spin systems.

Fix an integer number of time steps $T \geq 1$ and a number of spins $n \in \mathbb{N}$. Fix $p \in [1, \infty)$. Let μ, ν be prescribed initial and final probability distributions on the configuration space $\Omega_n = \{\pm 1\}^n$. We wish to find a sequence of controls $v_0, \ldots, v_{T-1}$ that solves the following minimization problem.

$$
\begin{aligned}
\min_{v} \quad & \sum_{t}\sum_{\sigma}\sum_{j} v(\sigma,j,t)^p \rho(\sigma,t) \\
\text{such that} \quad & \rho(\sigma,t+1) = \rho(\sigma,t)\left(1 - \sum_{j=1}^{n} \rho(\sigma,t)v(\sigma,j,t)\right) + \sum_{j=1}^{n} \rho(\sigma^j,t)v(\sigma^j,j,t) \\
& \rho_0 = \mu, \;\; \rho_T = \nu \\
& v(\sigma,j,t) \geq 0, \;\; \sum_{j} v(\sigma,j,t) \leq 1 \quad \text{for all} \quad \sigma,t
\end{aligned}
\tag{$\star$}
$$

We will refer to the first constraint as the evolution equation. The second constraint enforces initial and final conditions. The third constraint enforces that each v_t is a valid control.

3.2 A Fast Solver for Optimal Transport on Spin Systems in the case $p = 1$

In this section, we will reformulate the problem ($\star$) for $p = 1$ as a linear program. Though the dimension of this linear program is exponential in the number of spin sites n, we will describe some implementation details that make solving the problem computationally feasible for problems up to size $n \approx 20$.

3.2.1 A Linear Programming Reformulation

Henceforth, we will only consider the case where $p = 1$ in ($\star$); the corresponding metric is often called the *earth mover's distance* in classical optimal transport settings. In this section, we reformulate the problem as a network flow problem on the hypercube graph Q_n.

Given controls $v_0, \ldots, v_{T-1}$ and distributions $\rho_0, \ldots, \rho_{T-1}$, we define the following new quantity:

$$
f(\sigma,j,t) = v(\sigma,j,t)\rho(\sigma,t). \tag{3.2.1}
$$

We will refer to this quantity as the *flow*, since $f(\sigma,j,t)$ denotes the net flow of probability

mass from the configuration σ to the configuration σ^j at time step t. From (3.1.1), we can see that the flow at time step t transitions ρ_t to ρ_{t+1} in the following way:

$$\rho(\sigma, t+1) = \rho(\sigma, t) + \sum_j f(\sigma^j, j, t) - \sum_j f(\sigma, j, t). \tag{3.2.2}$$

We can interpret our optimal transport problem as finding a sequence of flows on the hypercube graph Q_n to map one probability distribution on the vertices to another in a way that minimizes the net probability mass that is moved across all time steps. Since the nodes of Q_n correspond to configurations $\sigma \in \Omega_n$, we will interchangably refer to configurations as nodes. We may interpret the quantities $\sum_j f(\sigma^j, j, t)$ and $\sum_j f(\sigma, j, t)$ in (3.2.2) as the net inflow and outflow, respectively, into/out of the node σ.

We now wish to rewrite $(\star)$ for $p = 1$ as a linear program in the variable f. For clarity of exposition, we first consider the case where $T = 1$, so that there is only one flow step $f(\sigma, j, 0)$ that transitions μ to ν. Then, dropping the time step argument t, the minimization problem $(\star)$ becomes

$$\begin{aligned} \min_v \quad & \sum_\sigma \sum_j f(\sigma, j) \\ \text{such that} \quad & \nu(\sigma) - \mu(\sigma) = \sum_j f(\sigma^j, j) - \sum_j f(\sigma, j) \\ & f(\sigma, j) \geq 0, \ \sum_j f(\sigma, j) \leq \mu(\sigma). \end{aligned} \tag{3.2.3}$$

The first constraint term in (3.2.3) enforces that the net change in probability at each spin site must equal the inflow minus the outflow, while the second term enforces that flows must be nonnegative, and no node can donate more mass than it already has. This is a linear

program in f, which we can rewrite in the following standard form:

$$\begin{aligned} \min_{\overline{f}} \quad & \mathbf{1}^T \overline{f} \\ \text{such that} \quad & \mathrm{A}_{\mathrm{ub}}\, \overline{f} \leq \mu \\ & \mathrm{A}_{\mathrm{eq}}\, \overline{f} = \nu - \mu \\ & \overline{f} \geq 0. \end{aligned} \tag{3.2.4}$$

(Here ub and eq are short for *upper bound* and *equality* constraints.) We use $\overline{f}$ to denote the $2^n n$-sized vector that results from the row-major (lexicographically-ordered) flattening of the matrix representation of $f(\cdot, \cdot)$, and $\mathbf{1}$ to denote the all-ones vector of the same size. The matrices A_{ub} and A_{eq} are chosen so that

$$[\mathrm{A}_{\mathrm{ub}}\, \overline{f}](\sigma) = \sum_j f(\sigma, j) \tag{3.2.5}$$

$$[\mathrm{A}_{\mathrm{eq}}\, \overline{f}](\sigma) = \sum_j f(\sigma^j, j) - \sum_j f(\sigma, j). \tag{3.2.6}$$

These are $(2^n, 2^n n)$-shaped block matrices; in the case $n = 2$, they are given by

$$\mathrm{A}_{\mathrm{ub}} = \left[\begin{array}{cc|cc|cc|cc} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right] \tag{3.2.7}$$

$$\mathrm{A}_{\mathrm{eq}} = \left[\begin{array}{cc|cc|cc|cc} -1 & -1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & -1 & -1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & -1 & -1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 & -1 \end{array}\right] \tag{3.2.8}$$

In general, these matrices will consist of 2^n blocks arranged in a row, each with shape $(2^n, n)$.

The pattern for the structure of A_{ub} is apparent. For A_{eq}, using zero-indexing, the σth block has negative terms in the σth row and positive terms in the (σ^j, j) slots for all $j = 1, \ldots, n$.

Now we wish to derive a standard formulation of the linear program $(\star)$ for general $T \geq 1$ analogous to (3.2.4). Since the flow $f(\cdot, \cdot, t)$ at time step t transitions the distribution ρ_t to ρ_{t+1}, we can unroll this recursion and write $\rho(\sigma, t)$ in terms of the flow f and initial distribution $\rho(\sigma, 0)$:

$$\begin{aligned}
\rho(\sigma, 0) &= \rho(\sigma, 0) \\
\rho(\sigma, 1) &= \rho(\sigma, 0) + \sum_j f(\sigma^j, j, 0) - \sum_j f(\sigma, j, 0) \\
\rho(\sigma, 2) &= \rho(\sigma, 1) + \sum_j f(\sigma^j, j, 1) - \sum_j f(\sigma, j, 1) \\
&= \rho(\sigma, 0) + \left[\sum_j f(\sigma^j, j, 0) - \sum_j f(\sigma, j, 0)\right] \\
&\quad + \left[\sum_j f(\sigma^j, j, 1) - \sum_j f(\sigma, j, 1)\right] \\
&\vdots \\
\rho(\sigma, T) &= \rho(\sigma, 0) + \sum_{t=1}^{T-1} \left(\sum_j f(\sigma^j, j, t) - \sum_j f(\sigma, j, t)\right) \\
\nu(\sigma) &= \mu(\sigma) + \sum_{t=0}^{T-1} \mathrm{A}_{\mathrm{eq}}\, \overline{f}_t.
\end{aligned}$$

(In the last line we are replacing ρ_0 and ρ_T with the prescribed initial and final distributions μ and ν.) We can thus write the initial and final condition constraints in $(\star)$ in terms of the flow as

$$\tilde{\mathrm{A}}_{\mathrm{eq}}\, \overline{f} = \nu - \mu. \tag{3.2.9}$$

Here $\tilde{\mathrm{A}}_{\mathrm{eq}}$ is a block matrix of shape $(2^n, T2^n n)$ with block structure consisting of T copies

of the matrix $\mathrm{A_{eq}}$ in a row:

$$\tilde{\mathrm{A}}_{\mathrm{eq}} = \begin{bmatrix} \mathrm{A_{eq}} & \mathrm{A_{eq}} & \cdots & \mathrm{A_{eq}} \end{bmatrix}. \tag{3.2.10}$$

The vector $\overline{f}$ is size $T2^n n$ and is the concatenation of $\overline{f}_0, \overline{f}_1, \ldots, \overline{f}_{T-1}$:

$$\overline{f} = \left[\overline{f}_0, \overline{f}_1, \ldots, \overline{f}_{T-1}\right] \tag{3.2.11}$$

We now must rewrite the constraint $\sum_j f(\sigma, j, t) \leq \rho(\sigma, t)$ for all t in terms of the flow and initial and final distributions. From (3.2.5), (3.2.6), and our previous unrolled recursion, we can write this as

$$\begin{aligned} \mathrm{A_{ub}}\,\overline{f}_0 &\leq \rho_0 \\ \mathrm{A_{ub}}\,\overline{f}_1 - \mathrm{A_{eq}}\,\overline{f}_0 &\leq \rho_0 \\ \mathrm{A_{ub}}\,\overline{f}_2 - \mathrm{A_{eq}}(\overline{f}_0 + \overline{f}_1) &\leq \rho_0 \\ &\vdots \\ \mathrm{A_{ub}}\,\overline{f}_{T-1} - \mathrm{A_{eq}}(\overline{f}_0 + \cdots + \overline{f}_{T-2}) &\leq \rho_0. \end{aligned}$$

We can rewrite the above system of inequalities as

$$\tilde{\mathrm{A}}_{\mathrm{ub}}\,\overline{f} = \mu_T. \tag{3.2.12}$$

Here $\tilde{\mathrm{A}}_{\mathrm{ub}}$ is a block lower triangular matrix of shape $(T2^n, T2^n n)$, where the block structure is shape (T, T) and consists of $\mathrm{A_{ub}}$ on the main diagonal and $-\mathrm{A_{eq}}$ on the strict lower

triangular portion, shown here for $T = 4$:

$$\tilde{\mathrm{A}}_{\mathrm{ub}} = \begin{bmatrix} \mathrm{A}_{\mathrm{ub}} & 0 & 0 & 0 \\ -\mathrm{A}_{\mathrm{eq}} & \mathrm{A}_{\mathrm{ub}} & 0 & 0 \\ -\mathrm{A}_{\mathrm{eq}} & -\mathrm{A}_{\mathrm{eq}} & \mathrm{A}_{\mathrm{ub}} & 0 \\ -\mathrm{A}_{\mathrm{eq}} & -\mathrm{A}_{\mathrm{eq}} & -\mathrm{A}_{\mathrm{eq}} & \mathrm{A}_{\mathrm{ub}} \end{bmatrix}. \tag{3.2.13}$$

We use μ_T to denote the vector of size $T2^n$ consisting of the T-times concatenation of μ with itself:

$$\mu_T = [\mu, \mu, \ldots, \mu]. \tag{3.2.14}$$

Combining (3.2.9) and (3.2.12), we can now rewrite ($\star$) as the following linear program in f in standard form:

$$\begin{aligned} \min_{\overline{f}} \quad & \mathbf{1}^T \overline{f} \\ \text{such that} \quad & \tilde{\mathrm{A}}_{\mathrm{ub}}\, \overline{f} \leq \mu_T \\ & \tilde{\mathrm{A}}_{\mathrm{eq}}\, \overline{f} = \nu - \mu \\ & \overline{f} \geq 0. \end{aligned} \tag{$\dagger$}$$

Although this is an exponentially large linear program ($\overline{f}$ is dimension $T2^n n$, $\tilde{\mathrm{A}}_{\mathrm{ub}}$ is dimension $(T2^n, T2^n n)$ and $\tilde{\mathrm{A}}_{\mathrm{eq}}$ is dimension $(2^n, Tn2^n)$, we will later show that we are nonetheless able to write an implementation that solves the linear program for moderately large n and T (up to $n = 23$, $T = 6$); this is because the constraint matrices are sparse and highly structured, with the only nonzero entries being ± 1. In addition, the dyadic structure of A_{eq} and A_{ub} allows us to express operations with these matrices using bitwise operations, which significantly speeds up computation.

3.2.2 Solving the Linear Program

As one of the most well-studied and classical problems in applied mathematics, there are a variety of ways of solving linear programs [47–49]. The two most classical methods are the simplex method [48] and the interior point method [50]. For both methods, the computational bottleneck is in a matrix factorization [51]; since the matrices in our linear programming are exponentially large, these methods are sure to lead to memory issues in computing and storing matrix factors. For large linear program first-order methods are often more suitable since the bottleneck is a matrix-vector multiplication, which is amenable for GPU acceleration. Indeed, one of the most successful modern linear program solvers is PDLP [52] developed by Google Research; an internally distributed implementation is able to solve linear programs with up to 92B nonzero constraint parameters [53]. The method is based on engineering improvements to the primal-dual hybrid gradient method, otherwise known as the Chambolle-Pock algorithm, first described in [54] and popularized by [55]. The Chambolle-Pock algorithm was initially motivated for convex optimization problems for image denoising problems. We will solve our linear program (†) using the standard Chambolle-Pock algorithm, noting that potential improvements using the techniques from [52] are possible.

We first briefly review the Chambolle-Pock algorithm. Given two finite dimensional vector spaces X and Y and a linear operator $K : X \to Y$, the Chambolle-Pock algorithm solves the saddle point problem of the form

$$\min_{x \in X} \max_{y \in Y} \langle Kx, y \rangle + G(x) - F^*(y). \tag{3.2.15}$$

The functions $G : X \to [0, +\infty]$ and $F^* : Y \to [0, +\infty]$ are proper convex lower semicontinuous (l.s.c.) functions, and F^* is the convex conjugate of a convex l.s.c. function F. This is

a primal-dual formulation of the primal problem

$$\min_x F(Kx) + G(x) \tag{3.2.16}$$

or the dual problem

$$\max_y -(G^*(-K^*y) + F^*(y)). \tag{3.2.17}$$

For F^* and G, we define the *proximal operators*

$$\begin{aligned} \text{prox}_{\sigma F^*}(\tilde{y}) &:= \text{argmin}_{y' \in Y} \left\{ \frac{\|y' - \tilde{y}\|^2}{2\sigma} + F^*(y') \right\} \\ \text{prox}_{\tau G}(\tilde{x}) &:= \text{argmin}_{x' \in X} \left\{ \frac{\|x' - \tilde{x}\|^2}{2\tau} + G(x') \right\}. \end{aligned} \tag{3.2.18}$$

The Chambolle-Pock algorithm solves (3.2.15) via the iteration in Algorithm 5.

Algorithm 5: The general form of the Chambolle-Pock algorithm

```
Initialize stepsizes τ, σ > 0, relaxation parameter θ ∈ [0, 1], primal and dual iterates
  (x^0, y^0) ∈ X × Y, set x̄^0 ← x^0, iteration counter k ← 0
while not converged do
    y^{k+1} ← prox_{σF*}(y^k + σK x̄^k)
    x^{k+1} ← prox_{τG}(x^k − τK* y^{k+1})
    x̄^{k+1} ← x^{k+1} + θ(x^{k+1} − x^k)
    k ← k + 1
end
```

To apply the Chambolle-Pock algorithm to our problem, we must first write (†) in the

form (3.2.16). This can be done with the following choices of K, F, and G:

$$
\begin{aligned}
K &:= \begin{bmatrix} \tilde{\mathrm{A}}_{\mathrm{ub}} & 0 \\ 0 & \tilde{\mathrm{A}}_{\mathrm{eq}} \end{bmatrix} \\
F(y,z) &:= F_1(y) + F_2(z) := \begin{cases} 0 & y \le \mu_T \\ +\infty & \text{else} \end{cases} + \begin{cases} 0 & z = \nu - \mu \\ +\infty & \text{else} \end{cases} \\
G(x) &:= \mathbf{1}^T x + \begin{cases} 0 & x \ge 0 \\ +\infty & \text{else}\,. \end{cases}
\end{aligned}
\tag{3.2.19}
$$

Here x corresponds to our primal variable $\overline{f}$. We have decoupled the dual variable into y and z, corresponding to the inequality and equality constraints respectively. The proximal operators of F_1^*, F_2^*, and G can be computed in closed form, and hence Algorithm 5 takes the form of Algorithm 6 when applied to (†). Notice that the performance bottleneck is in the matrix-vector products with $\tilde{\mathrm{A}}_{\mathrm{ub}}$, $\tilde{\mathrm{A}}_{\mathrm{eq}}$, $\tilde{\mathrm{A}}_{\mathrm{ub}}{}^T$, and $\tilde{\mathrm{A}}_{\mathrm{eq}}{}^T$. However, since these matrices are extremely sparse with only ± 1 nonzero entries, we are able to perform these operations very cheaply without ever explicitly forming the large matrices $\tilde{\mathrm{A}}_{\mathrm{ub}}$ and $\tilde{\mathrm{A}}_{\mathrm{eq}}$ and storing them in memory. The block structure of the matrices also is amenable to GPU acceleration. Furthermore, the action of the matrices on vectors involve accessing entries of the vectors whose indices are powers of two, allowing for speedup via bitwise operations. We have implemented fast matrix multiplications with $\tilde{\mathrm{A}}_{\mathrm{ub}}$, $\tilde{\mathrm{A}}_{\mathrm{eq}}$, $\tilde{\mathrm{A}}_{\mathrm{ub}}{}^T$, and $\tilde{\mathrm{A}}_{\mathrm{eq}}{}^T$ in JAX [35]; we compare the performance of our fast matrix multiplications with standard JAX/numpy matrix multiplication (i.e., `A@x`) in Figure 3.2.

In [55], the authors prove convergence with rate $O(1/N)$ of Algorithm 5 for $\theta = 1$ and $\tau\sigma\|K\|^2$, where here $\|\cdot\|$ denotes the operator norm. Computing the operator norm of our K in (3.2.19) is intractable for large n and T, but the block structure of K allows us to

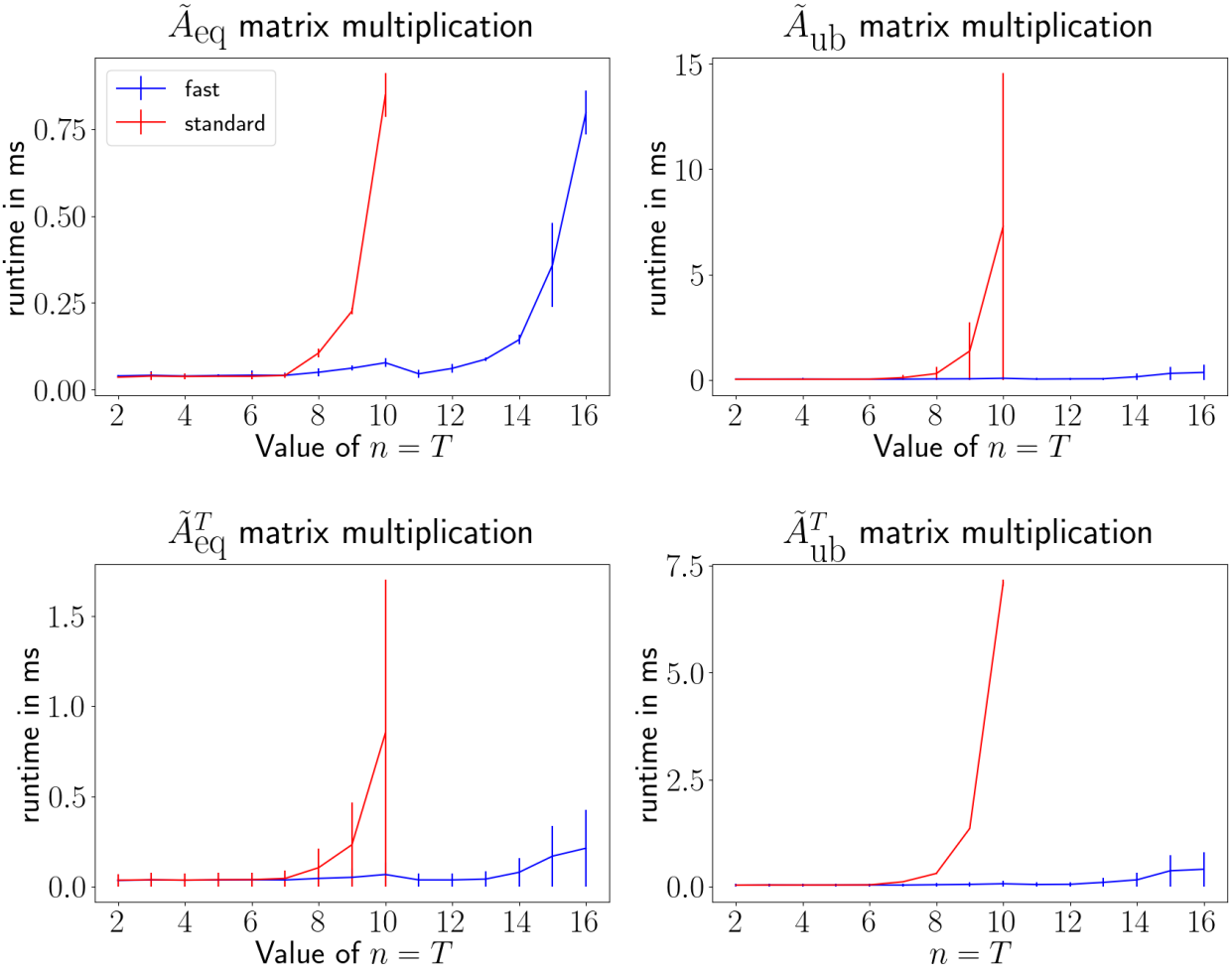

Figure 3.2: Comparing standard matrix multiplication (red) with our fast matrix-free methods (blue) of computing matrix-vector products with $\tilde{\text{A}}_{\text{ub}}$, $\tilde{\text{A}}_{\text{eq}}$, $\tilde{\text{A}}_{\text{ub}}{}^T$, and $\tilde{\text{A}}_{\text{eq}}{}^T$. We report the runtime (in milliseconds) of the standard and fast matrix multiplication for $n = T = 2, 3, \ldots, 16$; for fairness, both methods are implemented in JAX and just-in-time (JIT) compiled. The standard method is only able to run up to $n = T = 10$ due to memory limitations. Experiments were run on a single NVIDIA A40 GPU with 48GB of memory. Reported numbers are averages out of 100 runs; in each run, the matrices are multiplied by a different vector.

Algorithm 6: The Chambolle-Pock algorithm for solving (†: `SpinCP`)

Input: Initial and final prescribed distributions on Ω_n μ and ν; number of time steps T; number of spins n

Output: Estimated T-step flow f that solves (†).

Initialize step sizes $\tau, \sigma > 0$, relaxation parameter $\theta \in [0, 1]$, primal iterate $x^0 \in \mathbb{R}^{T2^n n}$, inequality constraint dual iterate $y^0 \in \mathbb{R}^{T2^n}$, equality constraint dual iterate $z^0 \in \mathbb{R}^{2^n}$, set $\overline{x}^0 \leftarrow x^0$, iteration counter $k \leftarrow 0$

while *not converged* **do**

  Dual updates:

    $y^{k+1} \leftarrow \max(0, y^k + \sigma(\tilde{\mathrm{A}}_{\mathrm{ub}}\, x^k - \mu_T))$

    $z^{k+1} \leftarrow z^k + \sigma(\tilde{\mathrm{A}}_{\mathrm{eq}}\, \overline{x}^k - (\nu - \mu))$

  Primal update:

    $x^{k+1} \leftarrow \max(0, x^k - \tau(\tilde{\mathrm{A}}_{\mathrm{ub}}{}^T\, y^{k+1} + \tilde{\mathrm{A}}_{\mathrm{eq}}{}^T\, z^{k+1} + \mathbf{1}))$

  Primal extrapolation:

    $\overline{x}^{k+1} = x^{k+1} + \theta(x^k - x^{k-1})$

  $k \leftarrow k + 1$

end

$\overline{f} \leftarrow x^k$

$f \leftarrow$ unflattening of $\overline{f}$

compute the following closed-form estimate:

$$\|K\| \leq \max\left\{2\sqrt{nT}, \sqrt{n} + \sqrt{n}\csc\left(\frac{\pi}{4T-2}\right)\right\}. \tag{3.2.20}$$

The authors of [55] also introduce an acceleration with dynamic stepsizes τ_n, θ_n and dynamic relaxation parameter θ_n that guarantees $O(1/N^2)$ convergence if either G or F^* is uniformly convex. For G uniformly convex with constant $\gamma > 0$, the acceleration is achieved by the dynamic stepsizes and relaxation parameter updates given by

$$\begin{aligned}
\theta_{k+1} &\leftarrow \frac{1}{\sqrt{1 + 2\gamma\tau_k}} \\
\tau_{k+1} &\leftarrow \tau_n\theta_n \\
\sigma_{k+1} &\leftarrow \sigma_n/\theta_n.
\end{aligned}$$

Although neither G nor F^* in (3.2.19) are uniformly convex, through experimentation we

have observed that in practice, choosing some $\gamma > 0$ and applying the acceleration nonetheless gives us faster convergence rates. After acceleration, our algorithm for solving (†) is given by Algorithm 7.

Algorithm 7: The Accelerated Chambolle-Pock algorithm for solving (†): `AccelSpinCP`

Input: Initial and final prescribed distributions on Ω_n μ and ν; number of time steps T; number of spins n

Output: Estimated T-step flow f that solves (†).

Initialize step sizes $\tau_0, \sigma_0 > 0$, relaxation parameter $\theta_0 \in [0, 1]$, primal iterate $x^0 \in \mathbb{R}^{T2^n n}$, inequality constraint dual iterate $y^0 \in \mathbb{R}^{T2^n}$, equality constraint dual iterate $z^0 \in \mathbb{R}^{2^n}$, uniform convexity constant $\gamma > 0$; set $\overline{x}^0 \leftarrow x^0$, iteration counter $k \leftarrow 0$

while *not converged* **do**

  Dual updates:

    $y^{k+1} \leftarrow \max(0, y^k + \sigma_k(\tilde{\mathrm{A}}_{\mathrm{ub}}\, x^k - \mu_T))$

    $z^{k+1} \leftarrow z^k + \sigma_k(\tilde{\mathrm{A}}_{\mathrm{eq}}\, \overline{x}^k - (\nu - \mu))$

  Primal update:

    $x^{k+1} \leftarrow \max(0, x^k - \tau_k(\tilde{\mathrm{A}}_{\mathrm{ub}}{}^T\, y^{k+1} + \tilde{\mathrm{A}}_{\mathrm{eq}}{}^T\, z^{k+1} + \mathbf{1})$

  Stepsize and extrapolation parameter update:

    $\theta_{k+1} \leftarrow \frac{1}{\sqrt{1+2\gamma\tau_k}}$

    $\tau_{k+1} \leftarrow \tau_n \theta_n$

    $\sigma_{k+1} \leftarrow \sigma_n / \theta_n$

  Primal Extrapolation:

    $\overline{x}^{k+1} = x^{k+1} + \theta_k(x^k - x^{k-1})$

  $k \leftarrow k + 1$

end

$\overline{f} \leftarrow x^k$

$f \leftarrow$ unflattening of $\overline{f}$

In practice, we compute the estimate k for $\|K\|$ given by (3.2.20), then set $\tau_0 = \sigma_0 = 1/k$. We set $\theta_0 = 1$ and choose $\gamma = 100$. We initialize x^0, y^0, and z^0 to be all zeros. Our convergence criterion is in the form of a fixed tolerance for the primal-dual gap, or a maximum number of iterations, whichever comes first.

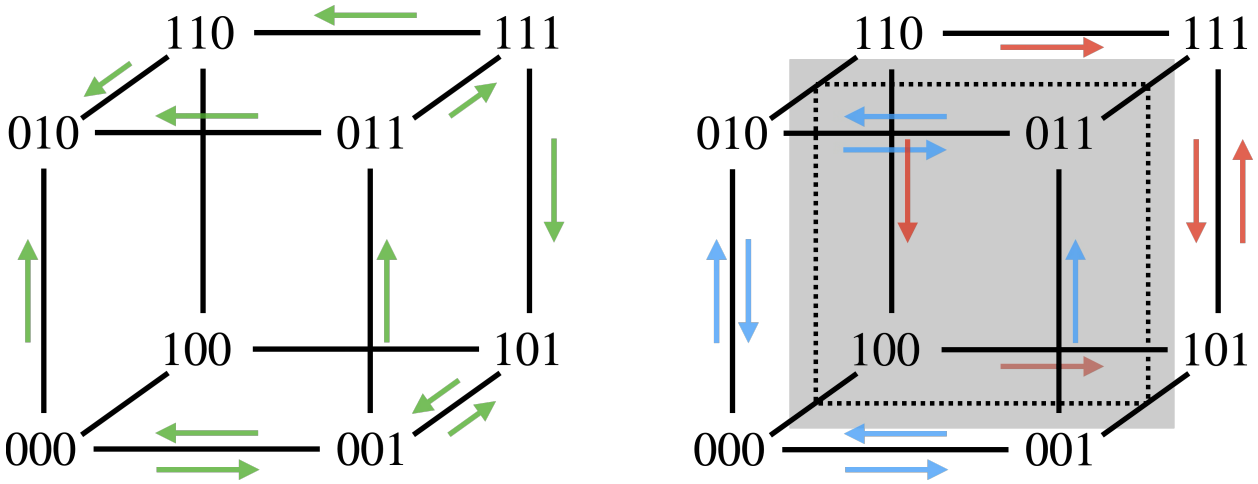

Figure 3.3: On the left, we depict a flow that is allowed to transport mass from any vertex of Q_3 to any other vertex. On the right, we partition Q_3 into two copies of Q_2 and only allow for the flow of mass to occur within each half; this gives rise to two smaller instances of our optimal transport problem that can be solved in parallel.

3.2.3 A Mean-Field Approximation

We now present a mean-field approach to solving (†) that, in practice, is able to find near-optimal solutions in a parallelizable way. The idea of the approach is to consider dyadic partitions of the hypercube Q_n and restrict the solution space to flows that do not cross between partitions; this gives rise to subproblems that can be solved in parallel.

To illustrate the idea, consider the graph flow interpretation of (†) on the hypercube Q_n for $n = 3$. Consider the partitioning of the cube into two halves along the axis corresponding to the most significant bit, as illustrated in Figure 3.3. If we could restrict our search for an optimal flow to those that do not cross the partition, then we can reduce our problem to two subproblems that are half the size of the original that can be solved in parallel. We call such a partition a *dyadic partition of depth 1*. We may further parallelize the problem by increasing the depth $d < n$ of the dyadic partition to divide the original problem into 2^d parallel subproblems.

However, note that for the subproblems to be feasible, we need the total mass within each dyadic partition to match the total mass within the dyadic partitions of the target distribution ν. Therefore, we need to first perform a rebalancing step that redistributes

mass around Q_n such that the mass within the dyadic partitions are equal (see Figure 3.4). To do this, we need to first solve the following linear program:

$$\begin{aligned} \min_{\overline{f}_r} \quad & \mathbf{1}^T \overline{f}_r \\ \text{such that} \quad & \tilde{\mathrm{A}}_{\mathrm{ub}}\, \overline{f}_r \leq \mu \\ & D_d\, \tilde{\mathrm{A}}_{\mathrm{eq}}\, \overline{f}_r = D_d(\nu - \mu) \\ & \overline{f}_r \geq 0. \end{aligned} \tag{3.2.21}$$

(The subscript r in $\overline{f}_r$ stands for *rebalancing.*) Here, D_d (short for "dyadic sum with depth d") is the $(2^d, 2^n)$-shaped matrix that computes the total mass within each of the 2^d dyadic partitions of a probability vector in $\mathbb{R}^{2^d}$; for example, in the case $n = 3$,

$$\begin{aligned} D_1(\mu_0, \mu_1, \mu_2, \mu_3, \mu_4, \mu_5, \mu_6, \mu_7) &= (\mu_0 + \mu_1 + \mu_2 + \mu_3, \mu_4 + \mu_5 + \mu_6 + \mu_7) \\ D_2(\mu_0, \mu_1, \mu_2, \mu_3, \mu_4, \mu_5, \mu_6, \mu_7) &= (\mu_0 + \mu_1, \mu_2 + \mu_3, \mu_4 + \mu_5, \mu_6 + \mu_7). \end{aligned}$$

For a feasible point $\overline{f}_r$ of (3.2.21), the resulting distribution $\mu' = \mu + \mathrm{A}_{\mathrm{eq}}\, \overline{f}_r$ will have the same mass in each dyadic partition as the target distribution ν.

In practice, for large n, we find that we are always able to transition μ to a rebalanced μ' with a single step flow ($T = 1$). Furthermore in practice, the amount of mass that must be transferred to transition μ to μ' is much smaller than the amount of mass that must be transferred to transition μ' to ν. Therefore, to speed up convergence in the rebalancing step, we set the primal objective in (3.2.21) to be identically zero and instead solve the rebalancing

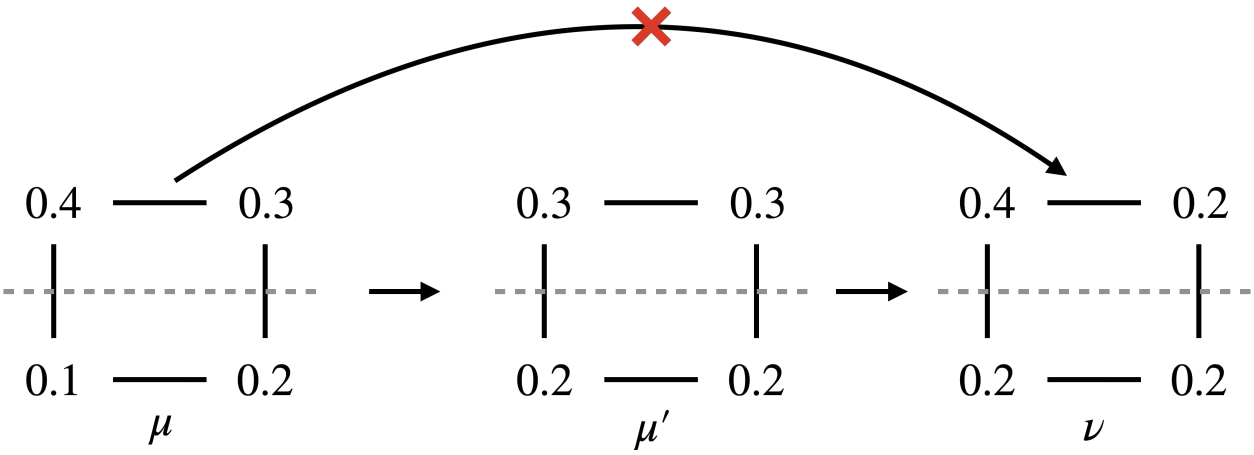

Figure 3.4: On the left, an initial distribution μ on Q_2, and on the right, a target distribution ν. If we partition Q_2 via the dotted line, it is impossible to find a dyadically partitioned flow that transitions μ to ν; the total masses on each half of μ are 0.3 and 0.7, while the total masses on each half of ν are 0.4 and 0.6. Thus we must first perform a rebalancing step that transitions μ to some distribution μ' such that the total mass within each dyadic partition is equal to the total mass within the dyadic partitions of the target distribution.

linear program for feasability only, rather than feasability and optimality.

$$\begin{aligned} \min_{\overline{f}_r} \quad & 0 \\ \text{such that} \quad & \tilde{\mathrm{A}}_{\mathrm{ub}}\,\overline{f}_r \leq \mu \\ & D_d\,\tilde{\mathrm{A}}_{\mathrm{eq}}\,\overline{f}_r = D_d(\nu - \mu) \\ & \overline{f}_r \geq 0. \end{aligned} \tag{3.2.22}$$

This linear program can also be solved using the Chambolle-Pock algorithm, and the matrix D_d once again is amenable to matrix-free matrix multiplication. Once we find a feasible rebalancing flow f_r and corresponding rebalanced distribution μ', we apply Algorithm 7 to each dyadic partition of μ' to transition to the dyadic partitions of ν; we can then merge the optimal flow f with f_r to form a flow transitioning μ to ν. This mean-field approach is fully detailed in Algorithm 8; note that this approach does not exactly solve the original problem stated in (†), since it outputs a $1 + T$-step flow rather than a T-step flow. This discrepancy can be amended by finding a $(T-1)$-step flow after the rebalancing step, but we

keep the number of steps at T for the subproblems in order to fairly compare computational performance later. Note critically that the iterations of the for-loop in Algorithm 8 can be solved in parallel.

Algorithm 8: The Mean-Field approach for solving (†): `MeanFieldSpinCP`

Input: Initial and final prescribed distributions on Ω_n μ and ν; number of time steps T; number of spins n; dyadic partitioning depth $d \geq 1$

Output: Estimated $1+T$-step flow f that solves (†).

Solve the rebalancing linear program (3.2.22) via Chambolle-Pock to get a feasible one-step rebalancing flow f_r and a rebalanced distribution μ'.

Split μ' and ν into dyadic partitions $[\mu'^0, \mu'^1, \ldots, \mu'^{2^d-1}]$, $[\nu^0, \nu^1, \ldots, \nu^{2^d-1}]$

for *partitions* $i = 0, \ldots, 2^d - 1$ **do**

  Solve $f^i = \texttt{AccelSpinCP}((\mu'^i, \nu^i, T, n-d))$.

end

Merge $f^0, \ldots, f^{2^d-1}$ to a single T step flow f' transitioning μ' to ν.

Merge f_r and f' to a single $1+T$ flow transitioning μ to ν.

3.3 Numerical Results

Here we present some numerical results on the performance of Algorithms 7 (`AccelSpinCP`) and 8 (`MeanFieldSpinCP`) for solving (†) with $T = 6$ time steps and $n = 23$ spins. In these dimensions, the primal variable f has $1,157,627,904$ entries and a memory profile of 4.63 GB; this is the largest problem instance we are able to solve on a single NVIDIA A40 GPU (48 GB of memory) with our implementation. Our implementation is written in Python with all numerical computing done in JAX. We generate initial and final distributions μ and ν in $\mathbb{R}^{2^n}$ from the flat Dirichlet distribution (i.e., the uniform distribution on the probability simplex [56]) and apply `AccelSpinCP` to find an optimal T-step transition from μ to ν; we set our tolerance to 1e$-$6, the maximum number of Chambolle-Pock iterations to be $500,000$, and $\gamma = 100$. We also run `MeanFieldSpinCP` on the same μ and ν with dyadic partitions of depth $d = 2$, 4, and 6. We will refer to the experiment with `MeanFieldSpinCP` as the $d = 0$ case.

In Figure 3.5, we plot the trajectories of the evolution $\mu = \rho_0, \rho_1, \ldots, \rho_T = \nu$ for the optimal flow computed by the four experiments above; we see that in each case, we are able to successfully compute a flow that transitions μ to ν.

In Table 3.1, we compare the runtimes (in seconds) and attained minima for our experiments; we may consider the minima attained for $d = 0$ as the ground truth. Observe that our mean field approaches are able to run orders of magnitude more quickly with only a small tradeoff in optimality.

Table 3.1: Comparing the runtime/accuracy tradeoff between various depths of dyadic splitting for solving (†) for $T = 6$ time steps and $n = 23$ spins. The vanilla approach without dyadic splitting takes 40.5 hours to run and reaches the maximum number of iterations. The most parallelized version that splits the problem into 2^6 parallel subproblems attains a minimum within 2.5% of the vanilla approach with a 595× speedup in runtime (with a total runtime of approximately 4 minutes).

Dyadic depth d	Attained minimum $\mathbf{1}^T\overline{f}$	Rebalancing step runtime (s)	Avg. dyadic subproblem runtime (s)
0	0.50127	–	145969 ± 0
2	0.50326	19	30628 ± 1757
4	0.50655	20	2981 ± 1418
6	0.51336	20	245 ± 112

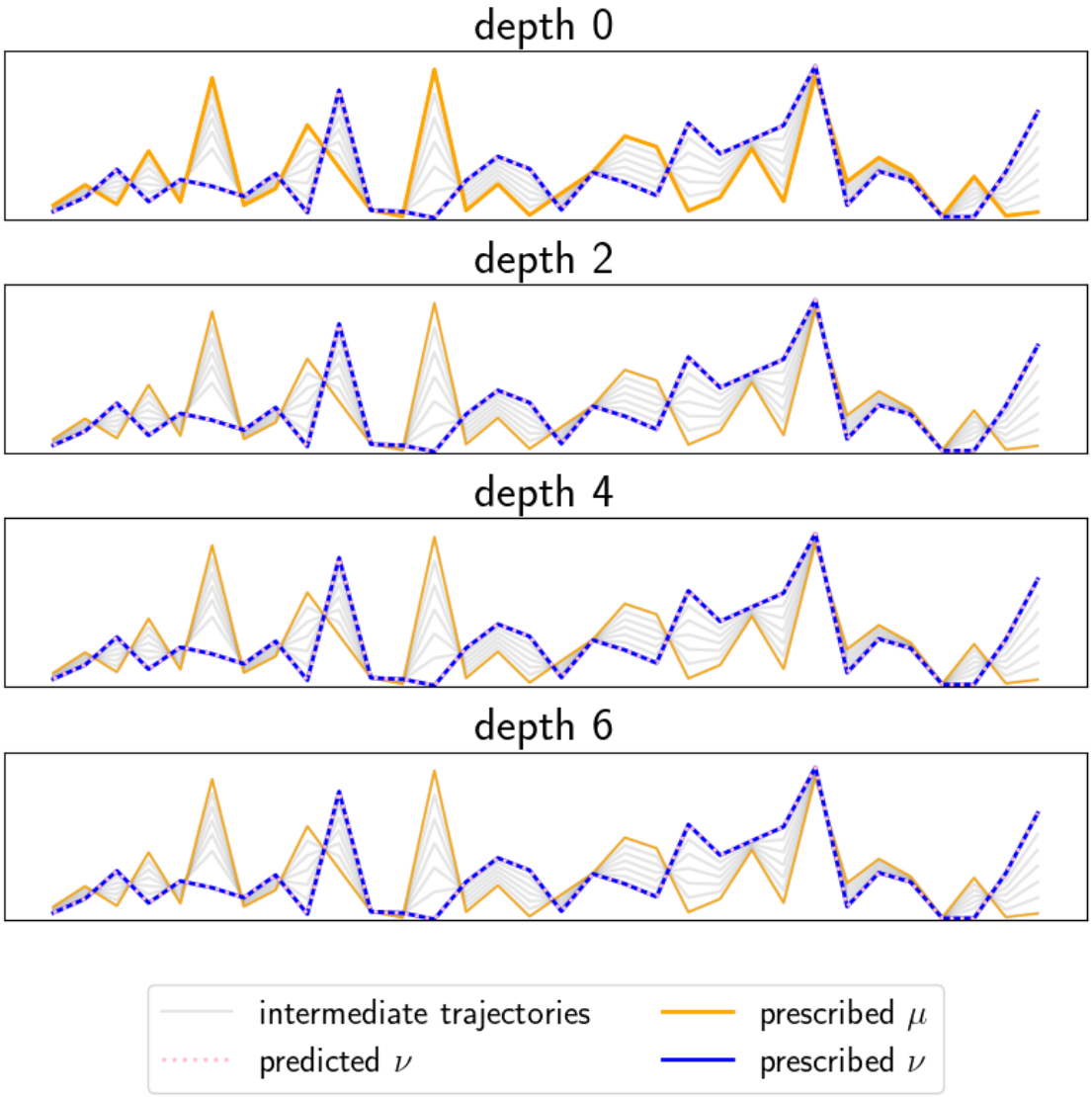

Figure 3.5: Computed optimal spin probability trajectories for $d = 0, 2, 4, 6$, $T = 6$, $n = 23$. The actual probability vectors are of size 2^{23}; we randomly sample 2^5 indices from the trajectories to plot. We see that in each case, our minimized flow evenly interpolates between the prescribed initial and final states.

3.4 Discussion

In this chapter we have drawn a connection between a relaxation of the Glauber dynamics on spin systems (one of the most foundational models in statistical physics) and optimal transport (one of the most foundational problems in applied mathematics). In particular, we view our relaxed Glauber dynamics as a discrete analogue of the Benamou-Brenier formulation of optimal transport. As with many discrete optimization problems, we have set up a linear program that is exponentially large. Nonetheless, we are able to write a Python implementation that solves the linear program for primal variables with up to 1.1 billion entries; the key ingredient that allows us to do this is the Chambolle-Pock algorithm applied to solving linear programs, combined with fast, matrix-free matrix-vector multiplications. We furthermore introduce a mean-field approximation to the problem that allows us to get approximately optimal solutions much more quickly.

The work in this chapter opens up many more avenues for further exploration. Some potential follow-up questions include, in order of roughly increasing scope:

1. Can we find an efficient method to solve ($\star$) for $p \geq 1$? What if we replace the cost function with some general convex function of v and ρ? What about a nonconvex function?

2. Can we adapt our technique from our relaxed Glauber model to more physically standard Glauber dynamics?

3. Can we find an approximate solver that removes the curse of dimensionality from the exponential size of the problem? For example, [57] uses marginal and cluster moment relaxation techniques to solve another exponentially large problem (solving the Fokker-Planck equation). Such techniques applied in our context would allow us to significantly scale our ability to solve larger problems.

4. What does optimal transport look like in a more general framework of the evolution

of a discrete system?

To paraphrase [58], the Ising model is to statistical mechanics as the fruit fly is to genetics; techniques initially developed for the Ising model have been adapted to study a wide variety of phenomena, both in physics and beyond. We hope that our initial foray into optimal transport in spin systems can be the seed for further insights into optimal transport for a variety of other discrete dynamical systems.

CHAPTER 4

HYPERDIFFUSIONFIELDS (HYDIF): DIFFUSION-GUIDED HYPERNETWORKS FOR LEARNING IMPLICIT MOLECULAR NEURAL FIELDS

In this chapter, we introduce HyperDiffusionFields (HyDiF), a framework that models 3D molecular conformers as continuous fields rather than discrete atomic coordinates or graphs. At the core of our approach is the Molecular Directional Field (MDF), a vector field that maps any point in space to the direction of the nearest atom of a particular type. We represent MDFs using per-molecule neural implicit fields, which we call Molecular Neural Fields (MNFs). To enable learning across molecules and facilitate generalization, we adopt an approach where a shared hypernetwork, conditioned on a molecule, generates the weights of the given molecule's MNF. To endow the model with generative capabilities, we train the hypernetwork as a denoising diffusion model, enabling sampling in the function space of MNFs. Our design naturally extends to a masked diffusion mechanism to support structure-conditioned generation tasks, such as molecular inpainting, by selectively noising regions of the field. Beyond generation, the localized and continuous nature of MDFs enables spatially fine-grained feature extraction for molecular property prediction, something not easily achievable with graph or point cloud based methods. Furthermore, we demonstrate that our approach scales to larger biomolecules, illustrating a promising direction for field-based molecular modeling.

This chapter is adapted from joint work with Sudarshan Babu and Aly Azeem Khan.

4.1 Introduction

Modeling complex, structured 3D data is a fundamental challenge in machine learning, with critical applications ranging from drug discovery and materials science to robotics and computer graphics [59–65]. Traditional approaches typically rely on discrete representations such

as point clouds, meshes, or voxel grids. However, each of these comes with limitations in resolution, expressiveness, or computational efficiency [66]. Recently, implicit neural representations (INRs), which model continuous fields over 3D space using coordinate-conditioned neural networks, have shown remarkable success in capturing high-fidelity geometry and generalizing across spatial scales [67, 68]. By operating directly in continuous space, INRs are uniquely suited for capturing fine-grained local geometric relationships crucial for modeling 3D objects [69, 70].

In this work, we focus on the domain of 3D modeling of small molecules, a setting where precise geometric understanding is essential for downstream tasks of biological interest such as binding affinity prediction, docking, and structure-based drug design [71–73]. The majority of existing approaches represent molecules as discrete objects, typically as graphs with nodes denoting atoms and edges denoting bonds, or as 3D point clouds representing spatial atomic positions. These representations treat molecular geometry as a sparse set of inputs, limiting the ability of machine learning models to capture continuous, fine-grained spatial relationships. This discrete treatment hinders the ability to learn localized features and gradients that are critical for modeling complex molecular interactions.

We introduce a paradigm shift in 3D molecular modeling by working directly in the continuous field space, rather than over discrete coordinates, molecular graphs, or latents. Instead of generating atomic positions, our method models the entire molecular structure as a spatial field. Specifically, we introduce the Molecular Direction Field (MDF), a representation that maps any point in 3D space to a vector pointing toward the nearest atom of a particular type. This field-based approach encodes molecular geometry as a smooth, dense function, offering several key advantages. First, it provides high-resolution structural information that can be queried at arbitrary spatial locations. Second, the continuous nature of the field enables implicit modeling of local geometry and spatial relationships. Third, the inherent redundancy in this representation, where nearby spatial points encode correlated directional signals, confers robustness to perturbations and offers a richer inductive bias than

sparse, discrete coordinate sets.

To model the MDF, we propose **Hy**per**Di**ffusion**F**ields (HyDiF), a generative framework for learning continuous molecular representations. We model each MDF with a Molecular Neural Field (MNF), a coordinate-conditioned neural implicit that models the MDF. Since training a separate MNF for each molecule prevents learning shared structure across molecules, we adopt an approach similar to HyperFields [74], where a shared hypernetwork generates the parameters of per-molecule MNFs, enabling cross-molecular generalization. This adoption is further motivated by recent findings that hypernetworks often exhibit improved generalization in out-of-distribution (OOD) settings [74, 75]. To enable generative capabilities, we train the hypernetwork as a denoising diffusion model [76] over the space of MDFs, conditioning it to produce molecule-specific MNFs. This allows explicit generation in the continuous function space of molecular fields, rather than discrete coordinate or graph representations or continuous latent representations.

Building on the model's ability to capture fine-grained local geometry through the MDF, we extend our framework to support molecular inpainting, a critical task for *in silico* drug discovery [77–80]. This leverages the locality of our representation to enable structure-conditioned generation, a more practical and relevant capability than unconditional generation. The ability of HyDiF to selectively regenerate parts of a molecule makes it well-suited for applications that require incorporating known structural constraints while completing or optimizing the rest.

In addition to generation, we utilize HyDiF to extract representations of molecules for downstream property prediction tasks. Generative models inherently learn rich, data-driven representations and recent work has demonstrated that such models excel at representation learning [81–85], enabling a unified framework that supports both generation and representation. Unlike traditional molecular encoders that produce a single global representation, our model naturally produces spatially localized features.

Finally, we demonstrate the scalability of our approach by applying HyDiF to proteins,

which were previously intractable with existing methods. This extension underscores the flexibility of our framework in handling molecules of varying size and complexity. Overall, our work highlights a promising new direction in molecular modeling, one that shifts from discrete representations toward continuous field-based models that unify representation learning and generation within a single framework.

4.2 Background and Related Work

In this section, we position our work relative to molecular representations, molecular generative and inpainting models, implicit neural representations, and hypernetworks.

4.2.1 Molecular Representations

A central problem in computational chemistry is that of how to represent molecules in a computer-parsable way, with a spectrum of different representations with tradeoffs between simplicity and physical fidelity/resolution. See [86] for a comprehensive and practical guide; here we offer a very brief overview of the most common molecular representations.

At the more simple end of the spectrum are string-based representations which map each valid molecule to an ASCII string. The most classical such method is SMILES (Simplified Molecular Input Line Entry System, [87–89]), first described in the 1980s and still the de-facto standard string-based molecular representation for computational chemistry purposes. Despite widespread use, SMILES strings do not fully capture the stereochemistry of a molecule and are not unique (one molecular species may have multiple valid SMILES representations), although there are are canonical algorithms for computing standardized SMILES strings. Another popular method is InChI (International Chemical Identifier, [90]), developed by IUPAC in the 2000s, which captures more stereochemistry information and is unique (each molecule has exactly one InChI representation), although at the expense of human-readability (SMILES strings are reasonably human-readable for smaller molecules).

In the age of generative artificial intelligence, the SELFIES (SELF-referencIng Embedded Strings, [91, 92]) method was developed in 2020, primarily motivated by the task of generating novel materials and molecules; the primary innovation of SELFIES is that every single combination of characters in the SELFIES alphabet corresponds to a valid molecule, allowing generative models to work directly on strings without having to worry about validity or parsability. String-based representations have the benefit of being cheap to generate and store, but their inherently linear format makes them an unnatural way to represent molecules and ill-suited for capturing rich geometric or topological information.

Graphical representations are perhaps the most broadly familiar representation, where a molecule is represented as a graph with vertices denoting atoms and edges denoting bonds. The vertices are tagged with information such as atom type, formal charge, implicit hydrogens, etc., while bonds are tagged with bond order information. Graph representations are inherently more natural and interpretable and are suitable for analysis via graph neural networks. However, the graph representation breaks down for molecules that are not accurately described by valence bond theory (i.e., the classical model of chemical bonds as either single, double, or triple bonds), as well as organometallic compounds containing ionic or metal-metal bonds. A far larger deficiency is that molecules are inherently three-dimensional structures, while graphs can only capture topological information. The three-dimensional structure of small molecules is especially critical for studying interactions with large biomolecules, which is essential for medicinal chemistry. As an extreme example illustrating the deficiency of graph representations, consider the case of molecular knots, i.e., molecules with a nontrivial knot structure. The first synthetic molecular knot was a trefoil knot synthesized in 1989 [93], and there is interest in such molecules for potential biological or materials science applications [94]. A graph representation has no way of capturing the unique structure of these molecules.

At a higher level of fidelity and complexity is the three-dimensional point cloud representation of a molecular *conformer*, i.e., a specific arrangement of a molecule in three-

dimensional space. A molecule is represented as an array of atomic positions in $\mathbb{R}^3$, tagged with atomic properties such as atom type and formal charge. While physically faithful to the three-dimensional nature of molecule, obtaining ground truth conformers is far more expensive than computing string or graph representations. X-ray crystallography and NMR spectroscopy experiments must be performed in the laboratory to obtain approximate structures, which then must be optimized with quantum mechanics calculations [95, 96]. A variety of classical and quantum mechanical methods exist to compute 3D conformers from physical first principles, most of which are based on energy minimization with the Merck molecular force field [97] or density functional theory methods [98–102]; these methods have varying degrees of physical fidelity and computational cost. One of the most commonly used computational methods is ETKDG (Experimental-Torsion basic Knowledge Distance Geometry, [103]) which has been implemented in the open-source chemoinformatics package RDKit [104]. Conformer generation with ETKDG has a relatively low computational cost compared to other physics-based methods, but the algorithm performs a sparse search over the space of all molecular conformers and can often miss many low-energy conformations or fail to find conformers for large, flexible molecules with many rotatable bonds. The current gold standard dataset of ground truth 3D conformer data for small molecules is GEOM (Geometric Ensemble of Molecules, [105]), a collection of 37 million high accuracy computed conformers for over 450,000 different small molecule species. Recently there has been considerable effort in ML-based methods for generating molecular conformers [106–114]; most of these methods, including ours, are trained on the GEOM dataset.

4.2.2 Models for 3D Molecular Generation

Existing methods for molecular generation employ variational autoencoders [115–117], normalizing flows [118–120], or generative adversarial networks [121–123] over strings or molecular graphs. These models often lack fine-grained control over 3D geometry, a critical limitation for tasks like conformer generation or structure-based drug design [106–108, 124,

125].

Denoising diffusion models have recently advanced the state-of-the-art in 3D molecular generation, with methods such as EDM [126], GeoLDM [127], and MiDi [128], applying noise to conformers and denoising in coordinate or latent spaces. These models often rely on equivariant architectures, but they still operate on sparse atomic representations. VoxMol [129] instead uses voxelized occupancy grids, providing a denser spatial representation, though at the cost of resolution and memory efficiency.

FuncMol [130] models 3D molecules using neural occupancy fields and performs diffusion in a learned latent space, capturing global molecular geometry while operating on compressed representations. This approach requires a two-stage training process, first learning the field representation, then training a separate diffusion model in latent space. Another related model MCF (Molecular Conformer Fields, [131]) approaches conformer generation by learning a distribution over functions that map molecular graph elements to 3D coordinates using diffusion.

In contrast to these approaches, HyDiF performs diffusion directly in the function space of Molecular Directional Fields (MDFs). Unlike latent-based methods, our framework is trained end-to-end and supports localized generation via masked denoising. While MCF models a mapping from molecular graphs to conformers and is limited to graph-to-structure generation, HyDiF models a function from 3D space to directional vectors, enabling both generation from noise and spatially localized structure-conditioned editing.

4.2.3 Molecular Inpainting and Scaffold-Constrained Generation

Molecular inpainting is the task of masking a portion of a molecule and filling it in with a generative model. Several existing methods such as ScaffoldGVAE [132], Sc2Mol [133], and MoLeR [80] operate on SMILES or molecular graphs, generating molecules by conditioning on a core scaffold and elaborating it via learned motifs or side-chain transformations. Others such as DiffSBDD [124] perform inpainting directly on 3D molecular coordinates in the

context of protein-ligand binding.

In contrast, HyDiF performs localized editing directly in continuous 3D space via masked denoising on Molecular Directional Fields. Unlike prior work, our edits are not constrained to predefined scaffolds or external protein contexts; instead, we enable user-specified, spatially targeted edits by masking arbitrary regions in the molecular field, offering a flexible, geometry-aware approach to molecular structure manipulation.

4.2.4 Molecular Foundational Models

Molecular foundational models train with a self-supervised task on a large corpus of molecules (on the order of millions) in order to learn rich vector representations of the data for downstream finetuning tasks. ChemBERTa-2 [134] is a foundational model trained on 77 million unique SMILES from PubChem using two self-supervised tasks. The first is a masked-language modeling (MLM) task, where a subset of characters in a SMILES string are masked and the model is tasked with predicting the missing characters. The second is a multitask regression (MTR) self-supervised task, where 200 molecular properties that can be easily calculated from SMILES strings alone are simultaneously regressed on. Uni-Mol [135] is another model, trained on 209 million ETKDG-generated conformers of 19 million molecular species, with the self-supervised task being the recovery of corrupted 3D atomic positions.

ChemBERTa-2 generates a global representation of a molecule (i.e., a molecule is mapped to a single vector), making it impossible to extract localized information from a molecule. Uni-Mol is able to generate both per-atom and global representations (the global representation being the average of the per-atom representations). In contrast, the continuous dense representation offered by our MDF parameterization of molecules allows for fine-grained representations of molecules at arbitrary locations, as we will elucidate in §4.4.3.

4.2.5 Implicit Neural Representations, Hypernetworks and Hyperfields

Implicit neural representations (INRs) have gained popularity in recent years as a way of modeling discrete data with a continuous function, parameterized by a multilayer perceptron (MLP). As an illustrative example, consider an $n \times n$-pixel RGB image; a traditional representation is simply the $n \times n \times 3$ array of per-pixel RGB values. An INR instead models the image as a function $f_\theta : [0, 1]^2 \to [0, 1]^3$, mapping each continuous point in the image to an RGB value. INRs have had considerable success in computer vision, both for 2D images and 3D volumes, especially for modeling local, high-frequency features [69, 136–138].

Traditionally in computer vision applications, INRs are per-prompt representations, i.e., an INR must be separately trained for each image or 3D scene. However, we wish to construct a model that is able to synthesize information *across* a wide variety of molecular conformers. A natural solution is to use *hypernetworks*, which are neural networks that themselves generate the weights for another neural network [139, 140]. Hypernetworks have been successfully applied in computer vision to successfully amortize the process of learning a new INR for each scene, synthesize information across scenes, and perform well in out-of-distribution generation tasks [74, 75, 141, 142].

HyDiF represents the first work to apply these insights from computer vision to molecules and train an end-to-end 3D molecular diffusion model *directly on MDF representations of molecules*. This enables parameter sharing across molecules, supports generation and inference over unseen structures, and allows us to train on large libraries of molecule-specific neural implicits in a unified framework.

4.2.6 Generative Models as Feature Extractors

Recent work in computer vision has shown that diffusion models can serve as effective feature extractors, with intermediate activations capturing rich semantic information necessary for generation [82, 143, 144]. Motivated by this, we explore the use of intermediate representations from HyDiF for downstream molecular tasks. This dual generation and representation

role makes HyDiF the first model in molecular conformer modeling to unify generation and feature extraction within a single framework.

4.3 Method

In this section, we describe the elements of our HyDiF framework. Each molecule is represented as a dense field that maps every point in 3D space to a vector pointing toward the nearest atom, yielding a locally structured and geometrically aware representation. Our model learns to generate such fields by reversing a diffusion process using a transformer-based hypernetwork that predicts the parameters of a neural field conditioned on observations of noisy fields. This formulation supports both unconditional generation and molecular inpainting, and enables local feature extraction from molecular structure. HyDiF brings together diffusion models for flexible generative modeling, hypernetworks for task-specific adaptivity, and implicit neural representations for capturing fine-grained 3D molecular structure. We provide a broad overview in this section, with more technical details left to §B.1.

4.3.1 Molecular Direction Fields

We represent 3D molecular conformers using *Molecular Direction Fields*, or MDFs, vector fields over continuous space that capture local geometric structure. At every point in space, the field points toward the nearest atom in the molecule, offering a dense representation of molecular geometry.

To build intuition for MDFs, consider constructing a direction field for a toy molecule consisting of atoms of a single type, e.g. carbon. Let $\mathcal{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$ denote the set of atomic coordinates. We define the vector field $\mathbf{F} : \mathbb{R}^3 \to \mathbb{R}^3$ where for any query point $\mathbf{q} \in \mathbb{R}^3$, the output $\mathbf{F}(\mathbf{q})$ is the vector pointing from $\mathbf{q}$ to the nearest atom in $\mathcal{A}$:

$$\mathbf{F}(\mathbf{q}) = \mathbf{a}_{\text{nearest}} - \mathbf{q}, \text{ where } \mathbf{a}_{\text{nearest}} = \text{argmin}_{\mathbf{a}_i \in \mathcal{A}} \|\mathbf{q} - \mathbf{a}_i\|_2. \tag{4.3.1}$$

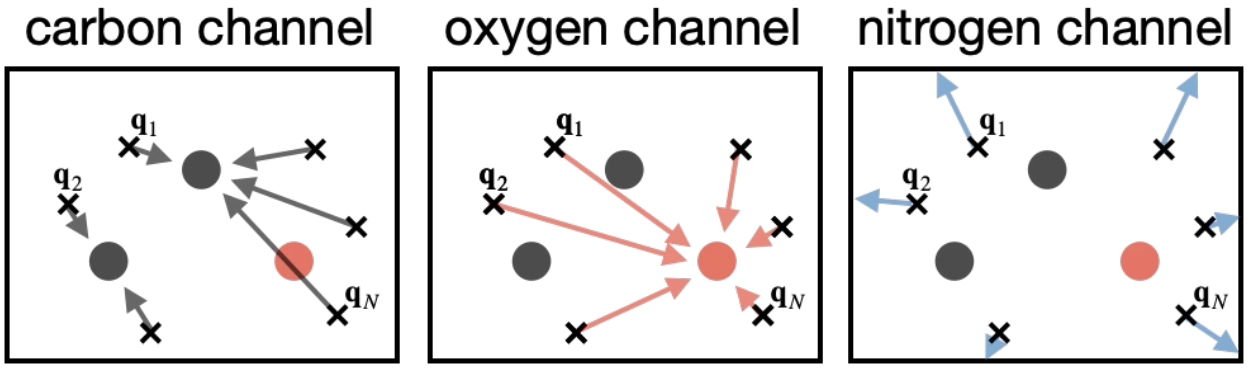

Figure 4.1: Channels $\mathbf{F}^{(k)}$ of an MDF for a molecule with two carbons (black), one oxygen (red), and no nitrogens (blue). Each query point is mapped to the nearest atom of the corresponding type. Since nitrogen is not present in the molecule, the nitrogen field points radially outwards.

This field assigns a direction even in regions of space far from any atom. By evaluating $\mathbf{F}$ over a dense set of points $Q = \{\mathbf{q}_j\}_{j=1}^{N}$, we obtain a high-resolution encoding of a molecule's geometry.

To extend to molecules with multiple atom types, we construct a separate direction field $\mathbf{F}^{(k)}$ for each atom type k over the subset $\mathcal{A}_k \subset \mathcal{A}$ which contains only atoms of type k. The resulting representation is a multi-channel vector field $\mathbf{F} : \mathbb{R}^3 \to \mathbb{R}^{K\times 3}$, where K is the number of distinct atom types in the dataset. Each channel $\mathbf{F}^{(k)}(\mathbf{q})$ is the vector from $\mathbf{q}$ to the nearest atom of type k.

Each molecule in our dataset is thus represented by a K-channel direction field, one channel for each possible atom type. If a particular type k is not present in a given molecule, we populate that channel with outward-pointing vectors that radiate from the molecule's center of mass toward a bounding sphere. This ensures that the representation remains dense and consistent across all molecules. This construction is illustrated in Figure 4.1.

4.3.2 Molecular Neural Fields

We approximate Molecular Direction Fields using *Molecular Neural Fields*, or MNFs coordinate-based neural networks trained to approximate the ground truth MDF of a molecule. Each

MNF is a function $\mathbf{F}_\theta : \mathbb{R}^3 \to \mathbb{R}^{K\times 3}$ that maps a query point to a multi-channel output, where the kth channel predicts the direction to the nearest atom of type k. The HyDiF architecture is trained to match $\mathbf{F}_\theta$ to $\mathbf{F}$ at sampled points and is parameterized as an MLP with sinusoidal activations known as a SIREN [67], which is well-suited for modeling high-frequency geometric signals.

4.3.3 Forward Diffusion Process

To model a distribution over MDFs, we adopt the denoising diffusion probabilistic model framework (DDPM, [76]), applied to vector fields. The forward diffusion process defines a Markov chain that gradually transforms a clean field into Gaussian noise over T discrete steps.

Given a ground truth direction field $\mathbf{F}$ evaluated at a set of spatial query points $Q = \{\mathbf{q}_j\}_{j=1}^N$, we construct the noised field at time step t as:

$$\mathbf{F}_t(Q) = \sqrt{\bar{\alpha}_t} \cdot \mathbf{F}(Q) + \sqrt{1-\bar{\alpha}_t} \cdot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, I), \tag{4.3.2}$$

where $\bar{\alpha}_t = \prod_{i=1}^t (1-\beta_i)$, and $\{\beta_i\}_{i=1}^T$ is a fixed noise schedule. We adopt the cosine schedule proposed in [145], which improves stability over the usual linear schedule in high-resolution spatial domains. Equation (4.3.2) corrupts each vector $\mathbf{F}_t^{(k)}(\mathbf{q})$ with additive Gaussian noise scaled according to the diffusion schedule (see Figure 4.2).

At time step $t = 0$, the noised field is exactly the ground truth clean field, while at the final step $t = T$, the noised field $\mathbf{F}_T(Q)$ becomes indistinguishable from isotropic Gaussian noise. This progression forms a trajectory through field space, allowing the model to learn to denoise under varying signal-to-noise ratios. The noise is applied independently to each vector component at each spatial location and for each channel in the field.

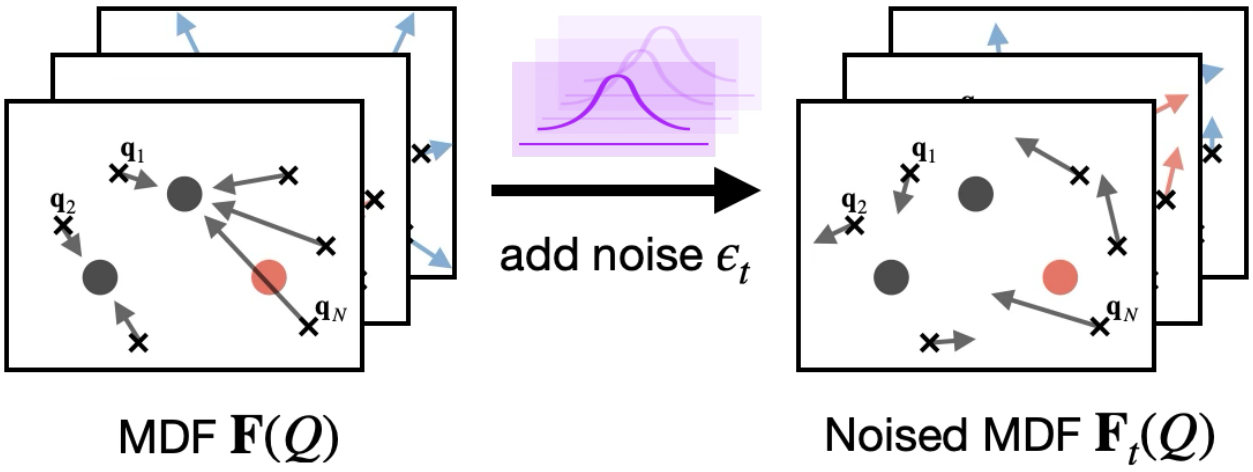

Figure 4.2: The forward noising process for an MDF, where i.i.d. Gaussian noise is added to each component of each vector in each channel at each query point. Here, $\boldsymbol{\epsilon}_t$ is shorthand for the scheduled noise $\sqrt{1-\bar{\alpha}_t} \cdot \boldsymbol{\epsilon}$.

4.3.4 Architecture

The core idea of the HyDiF architecture is to use a hypernetwork to generate the parameters of a MNF that denoises a noised molecular direction field (see Figure 4.3). The hypernetwork, denoted H_ϕ, takes three inputs: the noised direction field at time step t, denoted $\mathbf{F}_t(Q)$; the corresponding 3D query points $Q = \{\mathbf{q}_j\}_{j=1}^N$; and the diffusion time step t itself. The output is a set of parameters θ, which define a coordinate-based neural network $\mathbf{F}_\theta$ that serves as the denoised MNF. Formally, we have

$$\theta = H_\phi(\mathbf{F}_t(Q), Q, t). \tag{4.3.3}$$

Our training objective is to learn the hypernetwork weights ϕ that give the optimal θ so that the resulting MNF matches the ground truth field as close as possible, i.e. $\mathbf{F}_\theta \approx \mathbf{F}$.

4.3.5 Reverse Process and Training

Once the parameters θ are generated by the hypernetwork, we use the resulting MNF $\mathbf{F}_\theta$ to reverse the forward noising process. Specifically, we evaluate $\mathbf{F}_\theta$ at the same query points

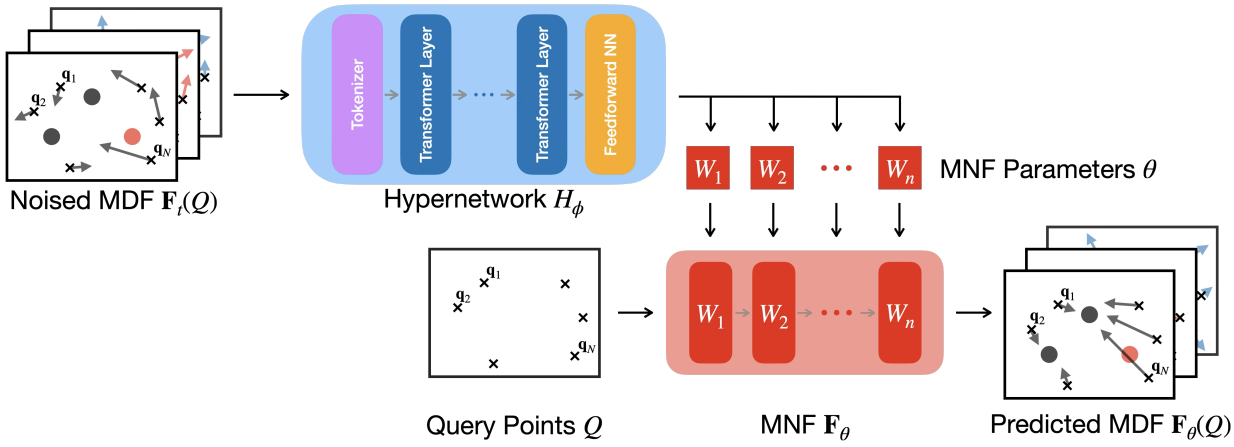

Figure 4.3: Overview of the HyDiF architecture. A hypernetwork H_ϕ takes a noised direction field $\mathbf{F}_t(Q)$, query points Q, and noising time step t, and produces the parameters θ of an MNF $\mathbf{F}_\theta$. The neural field is evaluated at the same query points to generate a denoised field, which is compared against the ground truth to compute the training loss. This enables HyDiF to model molecules in a spatially localized manner.

used in the forward process to predict the clean direction field. In theory, these predictions are then compared to the ground-truth direction vectors to minimize the following training objective:

$$\mathcal{L}_{\text{vec}}(\phi) = \frac{1}{|Q|} \sum_{\mathbf{q} \in Q} \sum_{k=1}^{K} \left\| \mathbf{F}_\theta^{(k)}(\mathbf{q}) - \mathbf{F}^{(k)}(\mathbf{q}) \right\|_1 . \tag{4.3.4}$$

In practice, we find this vector-based objective to be unstable for training purposes. Molecular direction fields contain high-frequency discontinuities at the boundaries between the Voronoi cells defined by the atomic positions. These discontinuities make the direction field difficult to fit with compact MLPs, especially when training with noisy inputs.

To address this, we instead train the MNF to predict the scalar *distance field*, defined as the Euclidean distance from a point $\mathbf{q}$ to the nearest atom of each type, i.e.,

$$f^{(k)}(\mathbf{q}) = \|\mathbf{a}^{(k)}_{\text{nearest}} - \mathbf{q}\|, \text{ where } \mathbf{a}^{(k)}_{\text{nearest}} = \text{argmin}_{\mathbf{a} \in \mathcal{A}_k} \|\mathbf{q} - \mathbf{a}\|_2. \tag{4.3.5}$$

This scalar function is smoother and easier for neural networks to approximate. We correspondingly define the MNF as a function $f_\theta : \mathbb{R}^3 \to \mathbb{R}^K$, where each output channel predicts

the distance to the nearest atom of type k. The associated training objective is then

$$\mathcal{L}_{\text{dist}}(\phi) = \frac{1}{|Q|} \sum_{\mathbf{q} \in Q} \sum_{k=1}^{K} \left| f_\theta^{(k)}(\mathbf{q}) - f^{(k)}(\mathbf{q}) \right|. \tag{4.3.6}$$

Through experimentation, we observe that the best performance is achieved when the hypernetwork is *conditioned on* a noised direction field, but trained to *output* a distance field. This hybrid setup leverages the high-frequency information present in the direction field for conditioning, while taking advantage of the smoother, lower-frequency distance field as the target for regression. We believe this setup helps the transformer process geometric structure more effectively. We provide ablation studies supporting this choice in §B.2.2.

One challenge with this formulation is that it introduces a mismatch between the model's input and output, which poses a challenge during generation (see §4.3.6): the input to the hypernetwork is a vector direction field, but the output is a scalar distance field. To resolve this, note that the direction field can be recovered from the distance field as follows:

$$\nabla_{\mathbf{q}} \frac{1}{2} \left(f^{(k)}(\mathbf{q}) \right)^2 = \nabla_{\mathbf{q}} \frac{1}{2} \left\| \mathbf{a}_{\text{nearest}}^{(k)} - \mathbf{q} \right\|^2 = \mathbf{a}_{\text{nearest}}^{(k)} - \mathbf{q} = \mathbf{F}^{(k)}(\mathbf{q}) \tag{4.3.7}$$

Here, $\mathbf{a}_{\text{nearest}}^{(k)}$ denotes the nearest atom of type k to point $\mathbf{q}$. This allows us to compute $\mathbf{F}^{(k)}(\mathbf{q})$ from the predicted scalar field using automatic differentiation. During generation, we apply this transformation at each time step to produce the direction field needed for the next step of the reverse diffusion.

We summarize the training loop in Algorithm 9.

4.3.6 Unconditional Generation

Algorithm 10 illustrates how we generate novel molecules after training HyDiF via Algorithm 9. We start by sampling Gaussian noise and taking that to be a molecule noised at time step $t = 1000$. We then ask the hypernetwork to denoise it to $t = 0$, then noise the denoised

Algorithm 9: Training Loop for HyDiF

Input: Query points Q, ground truth atomic coordinates $\mathcal{A}$

$\mathbf{F}(Q) \leftarrow$ ground truth MDF for $\mathcal{A}$;
for epoch $= 1, 2, \ldots$ **do**
  Sample a maximum noising level:
  if epoch ≤ 100 **then**
    $T \leftarrow 10$
  end
  $T \leftarrow \max(\text{epoch} - 100 + 10, 1000)$
  Randomly uniformly sample noise level t from $\{0, 1, \ldots, T\}$.
  Set noising coefficient α_t.
  Sample Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$ and noise the ground truth MDF: $\mathbf{F}_t(Q) \leftarrow \sqrt{\overline{\alpha}_t} \cdot \mathbf{F}(Q) + \sqrt{1 - \overline{\alpha}_t} \cdot \boldsymbol{\epsilon}$.
  Pass noised MDF into H_ϕ to get predicted MNF parameters: $\theta \leftarrow H_\phi(\mathbf{F}_t(Q), Q, t)$.
  Compute loss against ground truth distance field:
  $$\mathcal{L}_{\text{dist}} \leftarrow \frac{1}{|Q|} \sum_{\mathbf{q} \in Q} \sum_{k=1}^{K} \left| f_\theta^{(k)}(\mathbf{q}) - f^{(k)}(\mathbf{q}) \right|.$$
  Update hypernetwork parameters ϕ.
end

prediction by noise level $t = 999$ and repeat. Since the outputs of the hypernetwork are the weights to an implicit *distance field*, we must first compute the gradients with respect to the query points to convert it back to a *direction field* that we can pass back into the hypernetwork.

Algorithm 10: Generation Procedure for HyDiF

Initialize noise level $t \leftarrow 1000$, random query points Q.
Set noisy field to be Gaussian noise: $F_t(Q) \leftarrow \boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$.
for $t = 1000, 999, \ldots, 1$ **do**
  Push noisy field through the hypernetwork to get predicted MNF parameters: $\theta \leftarrow H_\phi(F_t(Q), Q, t)$.
  Compute predicted cleaned direction field by taking gradient of distance field w.r.t. query points: $\mathbf{F}_\theta(Q) \leftarrow \nabla_Q f_\theta(Q)$.
  Re-noise predicted direction field: $\mathbf{F}_{t-1}(Q) \leftarrow \sqrt{\overline{\alpha}_{t-1}} \cdot \mathbf{F}_\theta(Q) + \sqrt{1 - \overline{\alpha}_{t-1}} \cdot \boldsymbol{\epsilon}$.
end

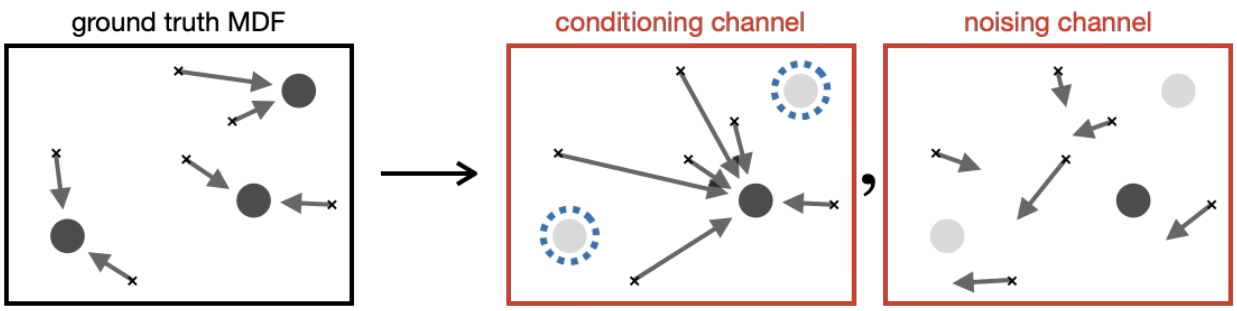

Figure 4.4: We extend HyDiF to support conditional generation by expanding the hypernetwork input to two channels. We randomly select some atoms from a molecule to noise and delete them before recomputing an MDF; this forms our *conditioning channel*, which conditions the hypernetwork on a partial molecule. We feed this into the hypernetwork, along with the *noising channel* computed by noising the ground truth MDF.

4.3.7 Masked Diffusion Procedure for Molecular Inpainting

The fact that HyDiF performs diffusion directly on geometric space allows us to train for a molecular inpainting task, in which a portion of a molecule is masked and the network is asked to complete the molecule. This allows us to selectively modify portions of existing molecules, a crucial step in drug discovery commonly referred to as *lead optimization* [146–149].

The training and generation pipelines for partial noising and inpainting are identical to that in §4.3.5 and §4.3.6, except in addition to a noised MDF, we also include a *conditioning MDF* as an input to the hypernetwork. The conditioning MDF is computed by randomly deleting a subset of atoms from a conformer and recomputing the MDF for the partial molecule. This allows the network to learn to denoise a fully noised MDF *conditioned on* a partial molecule being held constant (see Figure 4.4).

4.3.8 Feature Extraction with HyDiF

We illustrate how we leverage internal representations of HyDiF as feature representations of molecules for downstream molecular property prediction in Figure 4.5. Our method is inspired by recent work showing that diffusion models can be used for feature extraction

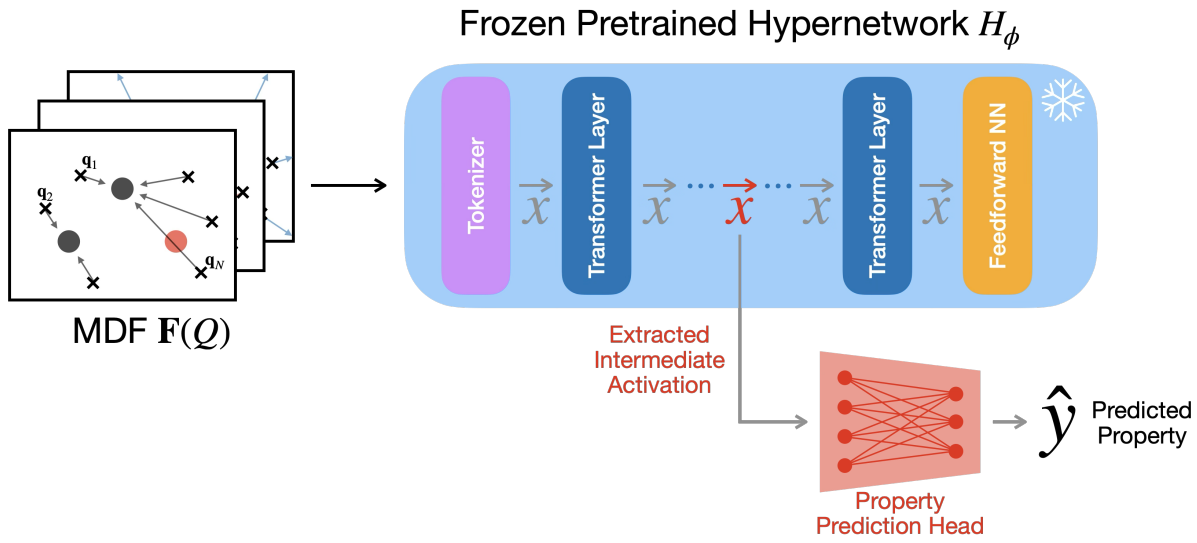

Figure 4.5: Overview of how we use HyDiF as a foundational model for property prediction. In practice, we always extract the central activation within the stack of transformer layers, e.g., given 26 transformer blocks, we extract the 14th intermediate activation.

in other domains [143, 144]. We pass an unnoised MDF for the molecule of interest into a frozen pretrained hypernetwork and extract per-query point intermediate activations, which we average over the query points. We then pass this activation into a property prediction head, which for us is a two layer MLP with tanh activations.

4.3.9 From MDFs to Molecules

While the output of HyDiF is an MDF for a conformer, at inference time it is more desirable to have a human-parsable representation of a molecule. Our method of parsing MDFs into molecules involves two steps: (1) recovering atomic point clouds from MNFs and (2) recovering molecular bond information from atomic point clouds. Observe that the zeros of a ground truth MDF are exactly the locations of atomic coordinates. Therefore to recover atomic coordinates from MNFs, we find minima of the generated distance field and use these minima as atomic coordinates, from which we can construct an atomic point cloud. To recover molecular bond information from atomic point clouds, we use a combination of the standard chemoinformatics toolkits RDKit [104] and OpenBabel [150] bond determination

algorithms to construct RDKit-parsable molecules.

4.4 Experiments and Results

Here we discuss our experiments demonstrating the efficacy of HyDiF in four domains: (a) unconditional generation of conformers, (b) localized partial generation of conformers, (c) extracting features from conformers for downstream property prediction tasks, and (d) scaling to biomolecules.

4.4.1 Pretraining HyDiF and Unconditional Generation

We pretrain HyDiF on the gold standard GEOM (Geometric Ensemble of Molecules, [105]) dataset, a collection of high quality conformers computed *in silico* using semi-empirical tight-binding density functional theory. We train on the same GEOM-drugs dataset and splits as [130], discarding the small number of molecules that contain elements other than C, H, O, N, F, S, Cl, and Br. The training set contains 1.1M conformers for 242K unique species. In Figure 4.6, we exhibit unconditionally generated samples from HyDiF after pretraining, highlighting the ability of HyDiF as a generative model for 3D conformers.

4.4.2 Molecular Inpainting

We train HyDiF on our inpainting task similarly to the pretraining method in §4.3.5 but with the masked diffusion procedure in §4.3.7. In Figure 4.7, we exhibit a conformer from the validation set with three different maskings and the corresponding inpainted molecules. In each case, HyDiF is able to successfully perform local edits on a molecule while maintaining chemical validity. This capability is primarily because of the flexibility of our model in performing diffusion directly on the space of molecules; in contrast, models that perform diffusion in latent space such as FuncMol and GeoLDM do not have this capability.

To quantitatively assess performance, we recover molecular graphs from inpainted MDFs

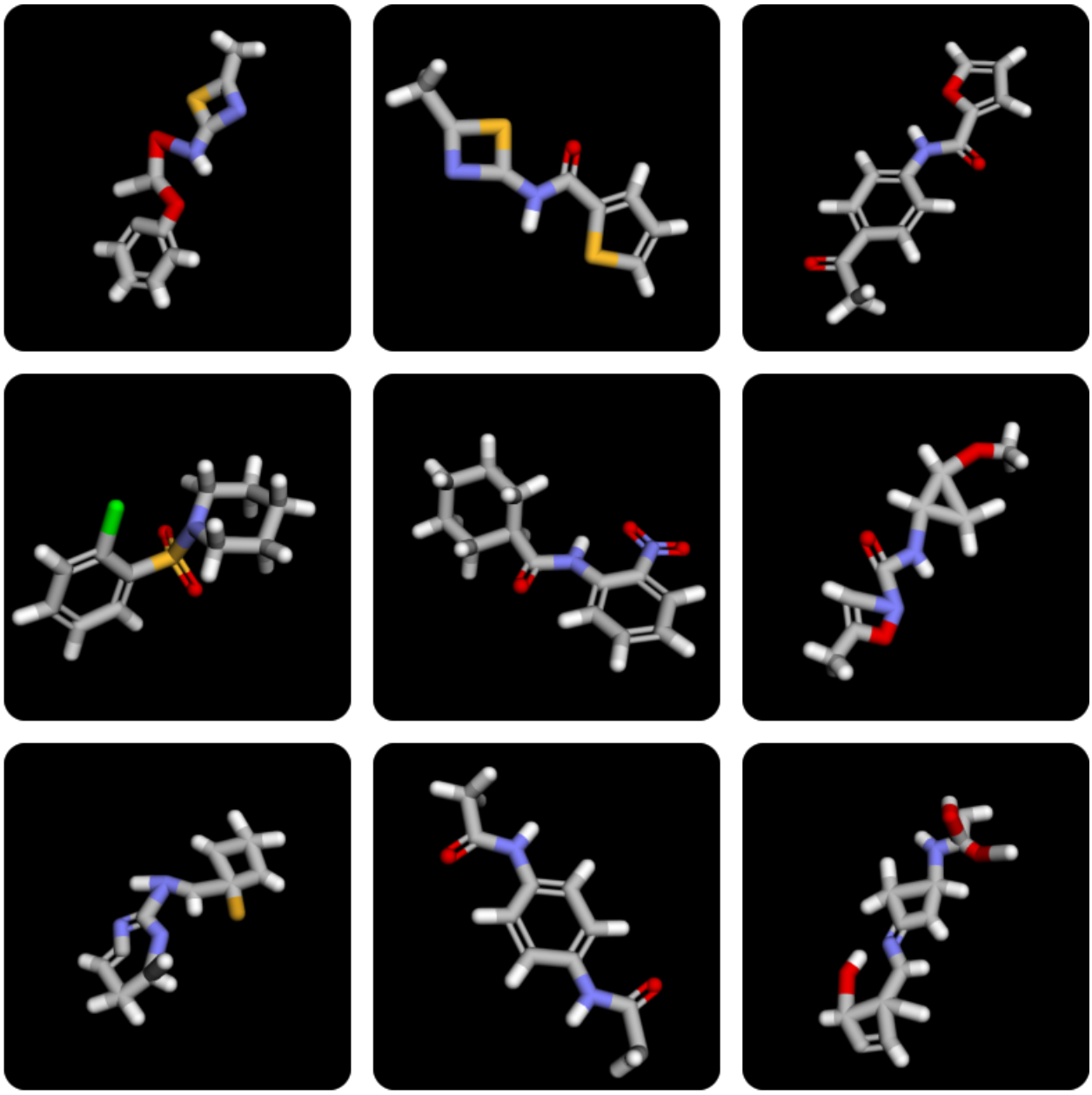

Figure 4.6: Nine curated samples from unconditional generation with HyDiF.

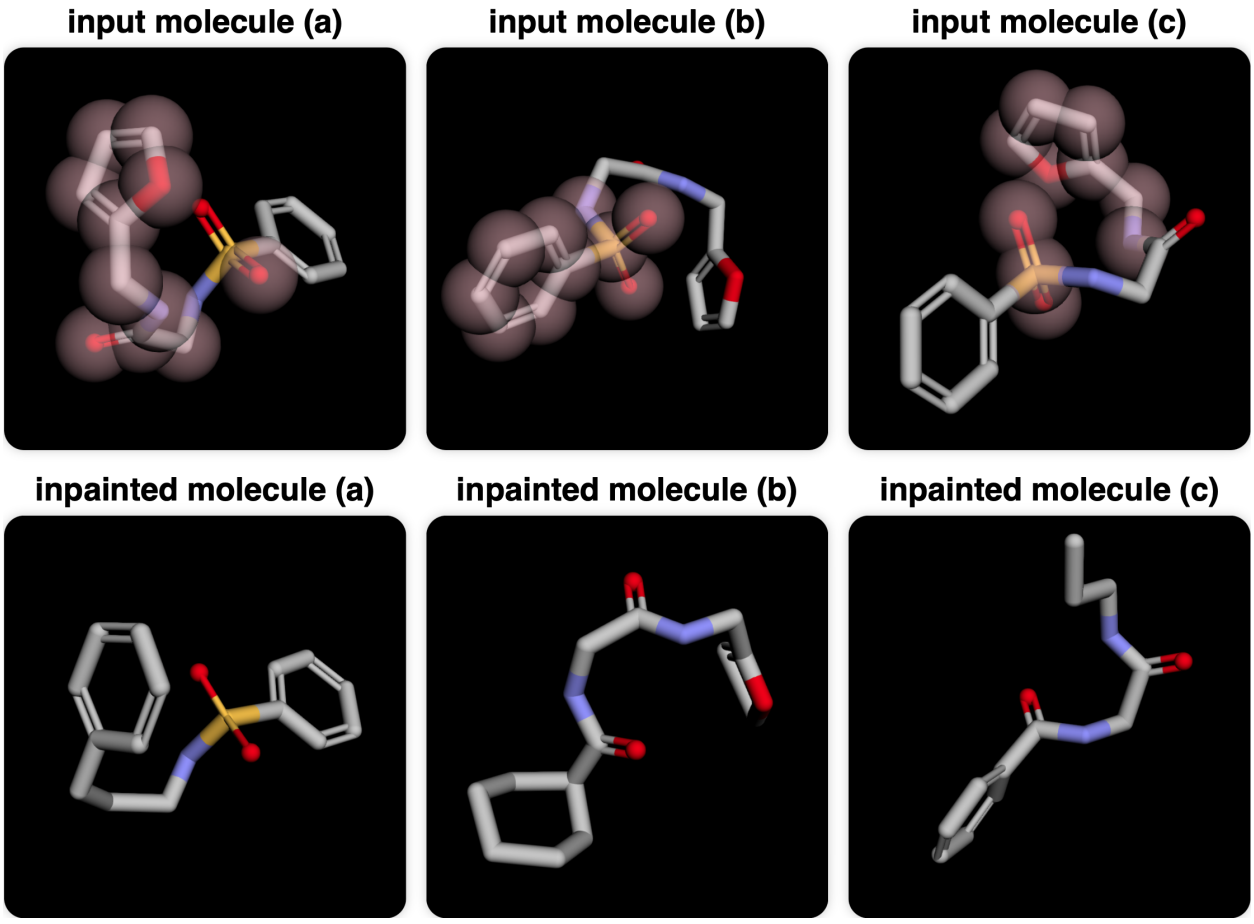

Figure 4.7: A molecule from the GEOM validation set with three different maskings (top row) and their corresponding inpaintings (bottom row). The spheres denote atoms in the molecule that are selected to be masked out. The above examples demonstrate the ability of HyDiF to inpaint effectively. Observe in inpainting (a) that the five-member ring with an oxygen has been replaced with a benzene ring. In inpainting (b), the benzene ring has been replaced with a hexane, and sulfur has been replaced with a carbon, and an oxygen has been deleted. In inpainting (c), an entire ring has been deleted.

Table 4.1: Molecular inpainting metrics on GEOM (higher is better). Metrics for EDM and HyDiF are computed over 100 generated conformers. HyDiF's high resolution molecular representation are able to inpaint more chemically reasonable molecules than EDM's discrete representation.

	Atom Stable %	Mol. Stable %	Validity %
data	100.0%	100.0%	96.9%
EDM	93.1%	13%	85%
HyDiF	**96.3**%	**47**%	**94**%

using the reconstruction procedure described in Section 4.3.9. Following prior work, we report three standard evaluation metrics: *atom stability* (fraction of atoms with correct valency), *molecule stability* (fraction of molecules where all atoms have correct valency), and *validity* (fraction of molecules that pass RDKit's sanitization check). Results for HyDiF and EDM, which we use as a baseline, are presented in Table 4.1. We evaluate EDM on the inpainting task by passing partially masked molecules and computing the metrics on the inpainted generated conformers. GeoLDM and FuncMol are not included in the conditional generation comparison, as it is not straightforward to adapt latent diffusion models for spatially grounded conditional generation.

We also include in Table 4.1 the metrics on the ground truth training data, computed over a random subset of 10K conformers; these represent an approximate upper bound.

4.4.3 Property Prediction

Following some of the ideas of [143, 144], we use a pretrained HyperDiffusionFields as a foundational model by feeding an unnoised MDF computed from a conformer and query points Q as input to a frozen pretrained hypernetwork, extracting a tensor of intermediate activations between self-attention blocks, and using that as input to a property prediction head for downstream tasks. For our experiments, we uniformly randomly sample a dense set of query points Q around the volume of the molecule and obtain a *global* feature by aggregating features over query points. However, observe that since the MDF and corresponding

intermediate activations are a function of Q, we could obtain *local* features by spatially localizing Q within a molecule. For instance, if we were interested in extracting a feature for a particular functional group of a molecule, we could concentrate the query points Q around the functional group. Other foundational models such as Uni-Mol and ChemBERT-a2 are unable to extract features in such a flexible manner.

We use a two-layer MLP with tanh activations as our property prediction head. As baselines, we compare the performance of our features generated from the hypernetwork to features from the MLM version of ChemBERTa-2. We also extract intermediate features from the diffusion models EDM and GeoLDM to compare against other diffusion models as feature extractors. In practice, we always extract features after the most central layer in each diffusion model, including our own. We also run an experiment where we concatenate our features with ChemBERTa-2 MLM features.

For downstream tasks, we select a subset of the tasks from the standard MoleculeNet benchmark [151], outlined in Table 4.2. Molecules in MoleculeNet datasets are specified as SMILES rather than 3D conformers, so in order to run EDM, GeoLDM, and HyDiF on MoleculeNet data, we must first generate conformers from SMILES strings using ETKDG.

Table 4.2: Overview of the subset of MoleculeNet tasks that we perform property prediction on.

Task Name	**Task Type**	**Train Size**	**Description**
BBBP	classification	1,631	blood-brain barrier penetration
BACE	classification	1,210	human β-secretase 1 binding
Lipo	regression	3,360	lipophilicity
ESOL	regression	902	water solubility
FreeSolv	regression	513	hydration free energy
QM7	regression	5,464	atomization energy

In Table 4.3, we show the results of our downstream property prediction experiments. Observe that for each task, the features taken from intermediate HyDiF are competitive with those from the foundational model ChemBERTa-2, as well as the features taken from the intermediate activations of the diffusion models EDM and GeoLDM. This is remarkable

due to the fact that ChemBERTa-2 is trained on 77 million unique species, approximately *300 times more species* than the data that HyDiF is trained on, and yet we are still able to achieve competitive performance, and state-of-the-art performance in five out of six tasks when we concatenate our features with those of ChemBERTa-2.

This highlights the ability of HyDiF to learn rich representations, even with a relative paucity of data. We attribute this performance to a fundamental difference in how our model represents molecules: rather than producing a single global embedding directly, HyDiF aggregates localized features learned from spatially dense query points. This allows the model to capture fine-grained geometric information that is otherwise lost in global or sequence-based representations.

Table 4.3: Performance comparison on classification (AUROC) and regression (RMSE) MoleculeNet tasks.

	classification (AUROC ↑)		**regression (RMSE ↓)**			
	BBBP	BACE	Lipo	ESOL	FreeSolv	QM7
ChemBERTa-2	0.70	0.79	0.81	0.98	2.12	143.9
EDM	0.67	0.78	0.94	1.03	2.25	**112.3**
GeoLDM	0.68	0.79	0.97	1.26	2.60	115.7
HyDiF	0.71	0.79	0.83	0.90	1.87	116.7
HyDiF+ChemBERTa-2	**0.73**	**0.80**	**0.78**	**0.89**	**1.75**	114.6

4.4.4 Scaling to Larger Biomolecules

HyDiF scales to large biomolecules in a way that current baseline architectures cannot. Most molecular generative models, such as EDM, GeoLDM, and VoxMol, are limited to small molecules with fewer than 100 heavy atoms. This is primarily due to architectural bottlenecks; graph neural network-based models like EDM and GeoLDM construct fully connected molecular graphs, resulting in $\mathcal{O}(n^2)$ pairwise interactions for n atoms, which becomes computationally expensive as molecular size grows. VoxMol represents molecules using dense 3D voxel grids, which are memory-intensive and scale cubically with the grid size. As shown in [130], VoxMol fails to scale to the CREMP dataset [152], which includes

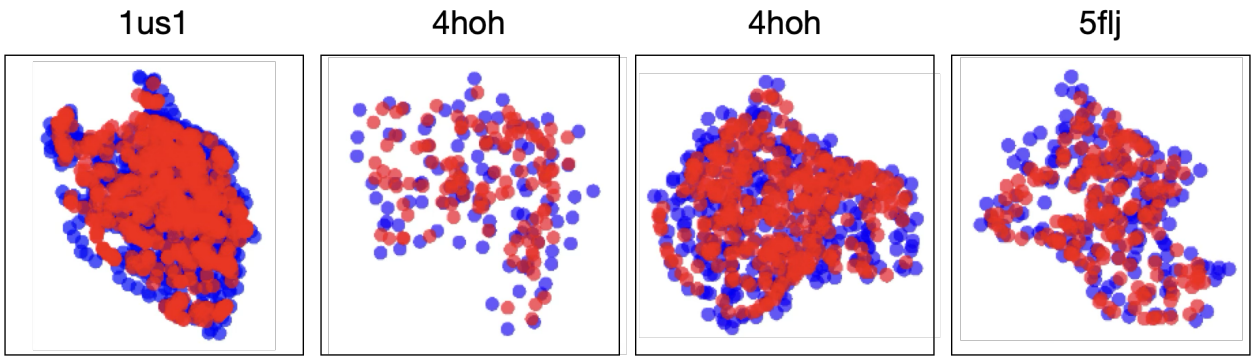

Figure 4.8: Four uncurated comparisons of ground truth alpha carbon coordinates of proteins from PLINDER (in blue) compared with coordinates reconstructed from denoising a noised MDF (in red). We see that in all four cases, we are able to reconstruct the overall shape of the protein. Each protein contains between 100 and 700 alpha carbons. This illustrates the scalability of HyDiF to large biomolecules. Subplot captions are Protein Data Bank identifiers.

macrocyclic peptides with an average of 74 heavy atoms.

Proteins contain hundreds to thousands of atoms. To evaluate the scalability of HyDiF, we train our diffusion pipeline (see §4.3.5) on alpha carbon traces from the PLINDER dataset [153], a collection of 3D protein structures. Figure 4.8 shows reconstructed proteins from the training set, generated by noising and denoising their molecular direction fields. In each case, we are able to recover the overall structure of the protein, a first step in demonstrating that HyDiF can learn the geometry large-scale macromolecules.

The key insight enabling scalability is that the computational cost of HyDiF scales quadratically with the number of query points, rather than the number of atoms in the molecule. This decouples computational complexity from molecular size: we train on large proteins by sampling a sparse set of query points, as the model sees each protein multiple times over training.

4.5 Discussion

This work introduces HyDiF, a generative framework that models 3D molecular structure via direct diffusion on continuous vector fields and neural function spaces. We first intro-

duce the Molecular Distance Field, a novel, dense representation of 3D molecular conformers, suitable for learning via implicit neural representations. We then introduce a hypernetwork architecture that is able to synthesize knowledge over a large library of conformers to generate per-conformer implicit representations which we call Molecular Neural Fields. We train this architecture as a diffusion model and demonstrate efficacy in both unconditional and structure-conditioned generation of conformers. We furthermore demonstrate that our diffusion model learns sufficiently rich internal representations of molecules to be used as features for downstream property prediction tasks, and our representations are competitive with both foundational and generative models on benchmark tasks. Moreover, the coordinate- and grid-free nature of our representation allows HyDiF to scale to larger biomolecules. We believe the contributions of HyDiF represent a promising direction for future work in geometric deep learning, one that bridges the gap between high-fidelity representation and flexible, spatially aware generation.

We believe the HyDiF paradigm opens up many new exciting avenues in the field of molecular modeling. For instance, we are interested in how we may adapt HyDiF for not just structure-conditioned generation, but property-conditioned generation, where we generate molecules conditioned on some desired chemical property, typically of pharmacological interest (e.g., absorption, solubility, etc.). This work can also be extended to protein-conditioned generation, where we seek to generate small molecule drugs that bind to a target protein, or even conditional generation of proteins themselves. Such advances would have the potential to significantly accelerate the process of drug discovery and understanding of structural biology.

CHAPTER 5

CONCLUSION AND OUTLOOK

We have considered in this work three extraordinarily different problems in applied mathematics, both in domain and in approach. In Chapter 2, inspired by theoretical challenges in cryo-EM, we study the recovery of one-dimensional manifolds from extremely noisy data. We prove a rigorous statement about the difficulty of this problem, paired with an algorithm that is able to solve the problem with the asymptotically minimal number of samples. Despite the difficulty of the problem, we are miraculously able to perform the recovery with nothing more than access to the first three moments of the ground truth curve; the difficulty is entirely shifted towards having enough samples to estimate these moments accurately. Highlighted in this work are the algebraic and analytic techniques used to study the moments. Our study of the noisy curve recovery problem is only a start to studying high-noise manifold learning problems; there is much work to be done to generalizing our results to arbitrary one-dimensional manifolds, or even manifolds of arbitrary dimension. There are also many exciting avenues in application of our recovery algorithm in scientific domains; our technique may prove fruitful in the analysis of real world high dimensional high-noise temporal data, including the actual cryo-EM problem.

In Chapter 3, we considered a different challenge, where it was the *dimension* of the data that was large, rather than the *quantity*. We developed and implemented a numerical method that was able to solve a linear program with over one billion variables. This linear program arose from an optimal transport problem on spin states inspired by the Glauber dynamics for the Ising model, which we reformulated to be amenable to a parallelized solution optimized for GPU acceleration. The core of our success lies in the implementation of extremely fast matrix-vector multiplies that exploit the sparse structure of the linear program. This work exhibits the performance gains in numerical optimization that are made possible by taking a deep dive into matrix arithmetic. This represents the beginning of many potential avenues of exploration in discrete optimal transport, particularly in more physically grounded problems.

Furthermore, despite being able to solve large problem instances, our method has not yet overcome the exponential size of the problem. Perhaps there are moment-based techniques that are able to drastically compress the size of the spin configuration space from being exponential in n to being polynomial or even linear in n. Overcoming this barrier would allow for the analysis of problems of significantly larger size.

Finally in Chapter 4, we designed a novel representation of conformers and a hybrid generative and foundational model for small molecules, achieving state-of-the-art results in molecular inpainting and property prediction. Our key contributions are in this novel representation, as well as applying hypernetwork methods to the molecular domain. This work represents the unreasonable effectiveness of neural networks to fit intractable, complex probability distributions over real data. Our new paradigm for molecular modeling represents the beginning of many potential approaches to grand challenges in structural biology, including modeling ligand-protein or protein-protein interactions, understanding the relationship between drug structure and biological function, and performing more and more of the drug discovery pipeline *in silico*. More broadly, we hope our approach elucidates the potential of dense, field-based representations in other scientific domains where data are typically represented by discrete objects.

Each of these three works began with a daunting task to be performed on a huge quantity of data. It was only after careful study of the underlying structure of the data and application of known priors that the intractable was made tractable, optimized for solutions via modern computing architectures. In dealing with data at this scale, such underlying structure is often one's only hope for learning anything from it.

That we have a seemingly unlimited amount of data in today's world is a wonder. That we are able to tame it with computations that ultimately reduce to arithmetic on floating point numbers is a miracle.

APPENDIX A

SUPPLEMENTARY MATERIAL TO CHAPTER 2

A.1 Cloud Tracing Algorithm for the Low-Noise Regime

In this section, we very briefly outline our algorithm for tracing through low-noise point clouds. We do not perform any large-scale experiments or detailed analysis of the method.

For simplicity, we consider point clouds coming from a PWL curve with an *a priori* known constant segment lengths L. First consider the issue of resolving an "elbow" with three points, $c_0, c_1, c_2 \in \mathbb{R}^d$, where c_1 is known. Since noise is low, we can compute the point cloud local to this elbow by considering all points in the cloud within distance L of c_1. The two-dimensional subspace formed by the three points can be estimated from the second empirical moment, onto which we can project the local point cloud. The direction (which we can identify with an angle in the two-dimensional subspace) of each vector pointing from c_1 to a projected point from the local cloud can be computed. The histogram of angle counts should be bimodal, with the peaks θ_0 and θ_2 occurring at the directions in which the projections of c_0 and c_2 lie. Since the segment lengths are known, we can then estimate the projections of c_0 and c_2 and deproject back to $\mathbb{R}^d$.

Given a full curve and an estimate of c_0, we can first compute the point cloud local to c_0, that is, all points that are within distance L away from c_0. We can apply a similar technique as above on this local cloud and c_0 to get an estimate for c_1, then iteratively apply the elbow resolution technique to the elbow centered at c_i and the point cloud local to c_i to estimate c_{i+1}. To account for the error that might propagate at each step, we can average the two curves obtained by applying this method with c_0 and c_M as the starting points. Figure 2.2 shows the outcome of this method on curves in $\mathbb{R}^2$ applied to only the initial point c_0.

The reason why this method fails for the high noise case is that the local point clouds are no longer meaningful. When noise is low, the points within distance L of c_i have a high likelihood of coming from the elbow given by c_{i-1}, c_i, and c_{i+1}, and so the geometric

structure of the curve around c_i is preserved by the local cloud. When noise is sufficiently high, the points within distance L of c_i are likely to have come from elsewhere in the curve. Without performing further detailed analysis, note that since this algorithm only requires accurate estimation of the second moment, the number of points needed for this method to work is order $O(\sigma^4)$, rather than the requirement of $O(\sigma^6)$ to accurately estimate the third moment for our high noise recovery algorithm.

A.2 Proof of Proposition 2.3.3

Let m_1 and m_2 be the first two moments of C^*, let $p^* \in \mathbb{R}^M$ be the proportional segment lengths of C^*, and let $Z^* > 0$ be the total length of C^*. Then C^*, p^*, Z^* is one particular solution to the following system of polynomial equations in $C = [c_0, \ldots, c_M]$, p, Z that we can write down using the relaxed moment expressions in (2.2.11):

$$\begin{aligned} \frac{1}{2} C^T A_s p &= m_1 \\ C^T \overline{A}(p) C &= m_2 \\ Z^2 p_i^2 &= \|c_i - c_{i-1}\|^2, \qquad i = 1, \ldots, M \\ \sum_{i=1}^{M} p_i &= 1. \end{aligned} \tag{A.2.1}$$

The first line in (A.2.1) is matching the relaxed first moment from C and p to the prescribed first moment m_1, and similarly for the second line and the second moment m_2. The third line "unrelaxes" the relaxed moments by requiring that the p_i are equal to the proportional segment lengths, and the fourth line ensures that Z is equal to the length of the full curve.

The first line of (A.2.1) consists of M equations, the second line consists of $M(M+1)/2$ equations (m_2 is a symmetric matrix, so we can eliminate all equations coming from the lower triangular part of m_2), the third line consists of M equations, and the last line is a single equation, so we have a total of $M(M+1)/2 + 2M + 1$ equations. To count the number

of variables, we have $M(M+1)$ variables from C, M variables from p, and one variable from Z for a total of $(M+1)^2$ variables. We therefore have more variables than equations, and hence this is an underdetermined system of polynomial equations.

If we subtract the right hand side from both sides of each line of (A.2.1), we can consider the left hand side of the entire system as a function $f : \mathbb{R}^{(M+1)^2} \to \mathbb{R}^{M(M+1)/2+2M+1}$ of (C, p, Z) (where scalars, vectors, and matrices are flattened and concatenated appropriately), and we can think of the solution set to (A.2.1) as the preimage of 0 under f.

Let $Df \in \mathbb{R}^{(M(M+1)/2+2M+1)\times(M+1)^2}$ be the Jacobian of f in the variables C, p, and Z; this is a short and wide matrix. In the next proposition, we claim that for almost all (C, p, Z) satisfying (A.2.1), the matrix $Df(C, p, Z)$ is full rank.

Proposition A.2.2. *Let (C, p, Z) be a solution to (A.2.1) and suppose the Jacobian $Df(C, p, Z)$ is not full rank. Then the all ones vector $\mathbf{1} \in \mathbb{R}^{M+1}$ must be in the image of C, where here we are thinking of C as a matrix.*

We omit the proof, which consists of routine calculation of the total derivative of f. Since C^* is a tall matrix with shape $(M+1, M)$, its image must be rank deficient, and therefore $\mathbf{1}$ is not in the image of C^* for almost all C^*. Therefore, for almost all C^*, $Df(C^*, p^*, Z^*)$ is full rank. Since rank is lower semicontinuous, for such a C^* there exists a neighborhood U of (C^*, p^*, Z^*) where Df is full rank and hence surjective at all (C, p, Z) in U. Let $f|_U : U \to \mathbb{R}^{(M+1)^2}$ be the restriction of f to U. Then 0 is a regular value of $f|_U$, i.e., a point such that $Df|_U$ is surjective at every point in $f|_U^{-1}(0)$. Note that we know that $f|_U^{-1}(0)$ is nonempty, since it contains at least the point (C^*, p^*, Z^*). Then by the regular level set theorem from differential topology (also called the preimage theorem, see Theorem 9.9 in [154]), $f^{-1}|_U(0)$ is a submanifold of U with dimension $(M+1)^2 - [M(M+1)/2+2M+1] = M(M-1)/2 > 0$. In other words, (C^*, p^*, Z^*) is not an isolated point in the real algebraic variety of (A.2.1). Therefore, for sufficiently small $\epsilon > 0$, we can find some $(\Gamma, \pi, \zeta) \in U$ such that $\|C^* - \Gamma\|_F = \epsilon$ and (Γ, π, ζ) also solves (A.2.1), and hence Γ has the same first and second moments as C^*.

A.3 Proof of Proposition 2.3.5

Our proof is modeled on the proof of Theorem A.1 in [22]. The idea of the proof is to bound the χ^2 divergence with a Taylor series of the differences of moments. We first need to prove a few technical lemmas and introduce some notation. Let $\phi_\sigma(x)$ denote the density of a Gaussian centered at the origin with covariance $\sigma^2 I_d$:

$$\phi_\sigma(x) = \frac{1}{\sqrt{(2\pi\sigma^2)^d}} \exp\left(-\frac{\|x\|^2}{2\sigma^2}\right). \tag{A.3.1}$$

Let $p_{C,\sigma}(x)$ denote the density of the noisy curve $P_{C,\sigma}$. Note that we have the expression

$$p_{C,\sigma}(x) = \frac{1}{Z}\int_0^1 \phi_\sigma(x - C(t))\, dt = \frac{1}{Z}\int_0^1 \frac{1}{\sqrt{(2\pi\sigma^2)^d}} \exp\left(-\frac{\|x - C(t)\|^2}{2\sigma^2}\right) dt. \tag{A.3.2}$$

Here Z is a normalizing constant dependent on C; for the rest of this section, we will assume $Z = 1$ for notational simplicity. We will also be using the facts that C is mean zero and $\|C(t)\| < 1$ throughout.

Lemma A.3.3. *We have the lower bound*

$$p_{C,\sigma}(x) \geq \phi_\sigma(x) \exp\left(-\frac{1}{2\sigma^2}\right). \tag{A.3.4}$$

Proof. Using the fact that the exponential function is convex, we compute

$$\begin{aligned}
p_{C,\sigma}(x) &= \int_0^1 \frac{1}{\sqrt{(2\pi\sigma^2)^d}} \exp\left(-\frac{\|x - C(t)\|^2}{2\sigma^2}\right) dt \\
&= \int_0^1 \frac{1}{\sqrt{(2\pi\sigma^2)^d}} \exp\left(-\frac{\|x\|^2}{2\sigma^2}\right) \exp\left(\frac{\langle C(t), x\rangle}{\sigma^2}\right) \exp\left(-\frac{\|C(t)\|^2}{2\sigma^2}\right) dt \\
&\geq \phi_\sigma(x) \exp\left(-\frac{1}{2\sigma^2}\right) \int_0^1 \exp\left(\frac{\langle C(t), x\rangle}{\sigma^2}\right) dt \\
&\geq \phi_\sigma(x) \exp\left(-\frac{1}{2\sigma^2}\right) \exp\left(\int_0^1 \frac{\langle C(t), x\rangle}{\sigma^2}\right) dt
\end{aligned}$$

$$= \phi_\sigma(x) \exp\left(-\frac{1}{2\sigma^2}\right)$$

The penultimate line is by Jensen's inequality, and the last line is by the fact that C is mean zero. ■

Lemma A.3.5. *Let C and Γ be any two curves. Then*

$$\int_{\mathbb{R}^d} \frac{p_{C,\sigma}(x) p_{\Gamma,\sigma}(x)}{\phi_\sigma(x)}\, dx = \int_{[0,1]^2} \exp\left(\frac{\langle C(t), \Gamma(s)\rangle}{\sigma^2}\right)\, dtds. \tag{A.3.6}$$

Proof. Note that we have

$$p_{C,\sigma}(x) = \phi_\sigma(x) \int_0^1 \exp\left(\frac{\langle C(t), x\rangle}{\sigma^2}\right) \exp\left(-\frac{\|C(t)\|^2}{2\sigma^2}\right)\, dt,$$

and similarly for $p_{\Gamma,\sigma}$. Then

$$\begin{aligned}
&\int_{\mathbb{R}^d} \frac{p_{C,\sigma}(x) p_{\Gamma,\sigma}(x)}{\phi_\sigma(x)}\, dx \\
&\quad = \int_{\mathbb{R}^d} \phi_\sigma(x) \left[\int_0^1 \exp\left(\frac{\langle C(t), x\rangle}{\sigma^2}\right) \exp\left(-\frac{\|C(t)\|^2}{2\sigma^2}\right)\, dt\right] \\
&\qquad\qquad \left[\int_0^1 \exp\left(\frac{\langle \Gamma(s), x\rangle}{\sigma^2}\right) \exp\left(-\frac{\|\Gamma(s)\|^2}{2\sigma^2}\right)\, ds\right] dx \\
&\quad = \int_{\mathbb{R}^d} \frac{1}{\sqrt{(2\pi\sigma^2)^d}} \exp\left(-\frac{\langle x, x\rangle}{2\sigma^2}\right) \left[\int_{[0,1]^2} \exp\left(\frac{2\langle C(t) + \Gamma(s), x\rangle}{2\sigma^2}\right)\right. \\
&\qquad\qquad \left. \exp\left(-\frac{\|C(t)\|^2 + \|\Gamma(s)\|^2}{2\sigma^2}\right)\, dtds\right] dx \\
&\quad = \int_{[0,1]^2} \left[\int_{\mathbb{R}^d} \frac{1}{\sqrt{(2\pi\sigma^2)^d}}\right. \\
&\qquad\qquad \exp\left(-\frac{\langle x, x\rangle - 2\langle C(t) + \Gamma(s), x\rangle + \langle C(t) + \Gamma(s), C(t) + \Gamma(s)\rangle}{2\sigma^2}\right) \\
&\qquad\qquad \left. \exp\left(\frac{\langle C(t) + \Gamma(s), C(t) + \Gamma(s)\rangle}{2\sigma^2}\right) \exp\left(-\frac{\|C(t)\|^2 + \|\Gamma(s)\|^2}{2\sigma^2}\right)\, dx\right] dtds \\
&\quad = \int_{[0,1]^2} \underbrace{\int_{\mathbb{R}^d} \frac{1}{\sqrt{(2\pi\sigma^2)^d}} \exp\left(-\frac{\|x - (C(t) + \Gamma(s))\|^2}{2\sigma^2}\right)\, dx}_{=1} \\
&\qquad\qquad \exp\left(\frac{\|C(t)\|^2 + \|\Gamma(s)\|^2 + 2\langle C(t), \Gamma(s)\rangle - \|C(t)\|^2 - \|\Gamma(s)\|^2}{2\sigma^2}\right)\, dtds \\
&\quad = \int_{[0,1]^2} \exp\left(\frac{\langle C(t), \Gamma(s)\rangle}{\sigma^2}\right)\, dtds.
\end{aligned}$$

In line 4, we are using Fubini's theorem and completing the square. ■

Lemma A.3.7. *Let $C(t)$, $\Gamma(t)$ be parametric curves such that $\|C(t) - \Gamma(t)\| \leq 1/3$ and $\|C(t)\|, \|\Gamma(t)\| \leq 1$ for all $t \in [0,1]$. With $\rho(C,\Gamma)$ as defined in (2.3.2), we have the upper bound*

$$\|m_k(C) - m_k(\Gamma)\|_F^2 \leq 12 \cdot 2^k \rho(C,\Gamma). \tag{A.3.8}$$

Proof. Using Jensen's inequality, we compute

$$\begin{aligned}
\|m_k(C) - m_k(\Gamma)\|_F^2 &= \left\| \int_0^1 C(t)^{\otimes k} - \Gamma(t)^{\otimes k}\, dt \right\|_F^2 \\
&\leq \int_0^1 \left\| C(t)^{\otimes k} - \Gamma(t)^{\otimes k} \right\|_F^2 dt \\
&\leq 12 \cdot 2^k \int_0^1 \|C(t) - \Gamma(t)\|^2 \, dt && \text{(Lemma A.5.1)} \\
&= 12 \cdot 2^k \rho(C,\Gamma).
\end{aligned}$$

■

We are now ready to prove Proposition 2.3.5.

Proof (of Proposition 2.3.5). Writing out the definition of the χ^2 divergence, we have

$$\begin{aligned}
&\chi^2(P_{C,\sigma} \| P_{\Gamma,\sigma}) \\
&\quad = \int_{\mathbb{R}^d} \frac{(p_{C,\sigma}(x) - p_{\Gamma,\sigma}(x))^2}{p_{C,\sigma}(x)}\, dx \\
&\quad \leq \underbrace{\exp\left(\frac{1}{2\sigma^2}\right)}_{\leq e^{1/2}} \int_{\mathbb{R}^d} \frac{(p_{C,\sigma}(x) - p_{\Gamma,\sigma}(x))^2}{\phi_\sigma(x)}\, dx && \text{(Lem. A.3.3)} \\
&\quad \lesssim \int_{[0,1]^2} \exp\left(\frac{\langle C(t), C(s)\rangle}{\sigma^2}\right) - 2\exp\left(\frac{\langle C(t), \Gamma(s)\rangle}{\sigma^2}\right) + \exp\left(\frac{\langle \Gamma(t), \Gamma(s)\rangle}{\sigma^2}\right) dtds && \text{(Lem. A.3.5)} \\
&\quad \leq \int_{[0,1]^2} \sum_{k\geq 0} \frac{1}{\sigma^{2k} k!} \left(\langle C(t), C(s)\rangle^k - 2\langle C(t), \Gamma(s)\rangle^k + \langle \Gamma(t), \Gamma(s)\rangle^k \right) dtds && \text{(Taylor exp.)} \\
&\quad = \sum_{k\geq 0} \frac{1}{\sigma^{2k} k!} \int_{[0,1]^2} \Big[\left\langle C(t)^{\otimes k}, C(s)^{\otimes k} \right\rangle && \text{(Fubini)}
\end{aligned}$$

$$
\begin{aligned}
&\quad - 2\left\langle C(t)^{\otimes k}, \Gamma(s)^{\otimes k}\right\rangle + \left\langle \Gamma(t)^{\otimes k}, \Gamma(s)^{\otimes k}\right\rangle \Bigg] dtds \\
&= \sum_{k\geq 0} \frac{1}{\sigma^{2k} k!} \Bigg[\langle m_k(C), m_k(C)\rangle - 2\langle m_k(C), m_k(\Gamma)\rangle + \langle m_k(\Gamma), m_k(\Gamma)\rangle \Bigg] \\
&= \sum_{k\geq 0} \frac{1}{\sigma^{2k} k!} \|m_k(C) - m_k(\Gamma)\|_F^2 \\
&\leq \sum_{k\geq \ell} \frac{12\cdot 2^k}{\sigma^{2k} k!} \rho(C,\Gamma) \qquad \text{(Lem. A.3.7)} \\
&= \mathcal{K}_\ell \sigma^{-2\ell} \rho(C,\Gamma).
\end{aligned}
$$

In line 4, $\lesssim$ denotes inequality up to a universal constant (here, $e^{1/2}$). In line 5, we are using the fact that $\langle x, y\rangle^k = \left\langle x^{\otimes k}, y^{\otimes k}\right\rangle$, where $\langle \cdot, \cdot \rangle$ denotes the Frobenius inner product for tensor arguments. In the penultimate line we are using the fact that the moments match up to order $\ell - 1$. ■

A.4 Vectorized Notation for Moments

Here we elaborate on the notation in Definition 2.2.9. We use A^s, A^l, and A^r to denote the following three $(M+1, M)$-shaped matrix constants (here $M = 3$ for illustrative purposes):

$$
A^s = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \quad A^l = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad A^r = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.4.1}
$$

(s stands for sum, l for left, and r for right, since the rows of $(A^s)^T C$ are $c_{i-1} + c_i$, the rows of $(A^l)^T C$ are c_{i-1}, and the rows of $(A^r)^T C$ are c_i). The $(M+1, M+1)$-shaped matrix $\overline{A}(p)$ is given by

$$
\overline{A}(p) := \frac{1}{6}\left(A^s \operatorname{diag}(p)(A^s)^T + A^l \operatorname{diag}(p)(A^l)^T + A^r \operatorname{diag}(p)(A^r)^T \right), \tag{A.4.2}
$$

We use $\alpha(p)$ to denote the symmetric $(M+1, M+1, M+1)$-shaped tensor whose mno entry is given by

$$[\alpha(p)]_{mno} = \sum_{i=1}^{M} p_i \left(\frac{1}{12} A^s_{mi} A^s_{ni} A^s_{oi} + \frac{1}{6} A^l_{mi} A^l_{ni} A^l_{oi} + \frac{1}{6} A^r_{mi} A^r_{ni} A^r_{oi} \right). \tag{A.4.3}$$

A.5 Miscellaneous Technical Lemmas

In this section, we collect miscellaneous technical lemmas and information-theoretic facts.

Lemma A.5.1. *Let $x, y \in \mathbb{R}^d$, $\|x\|, \|y\| \leq 1$. Suppose also that we have $\|x - y\| \leq 1/3$. Then for any $k \geq 1$, we have the bound*

$$\|x^{\otimes k} - y^{\otimes k}\|_F^2 \leq 12 \cdot 2^k \|x - y\|^2. \tag{A.5.2}$$

The proof is adapted from the proof of Lemma B.12 in [155].

Proof. Let $\gamma := \langle x, y - x \rangle$. Note $|\gamma| \leq \|x\|\|x - y\| \leq \|x - y\|$. Observe that we can write

$$\begin{aligned} \langle x, y \rangle &= \|x\|^2 + \gamma \\ \|y\|^2 &= \|x\|^2 + 2\gamma + \|x - y\|^2. \end{aligned}$$

By the binomial theorem,

$$\begin{aligned} &\|x^{\otimes k} - y^{\otimes k}\|_F^2 \\ &\quad = \|x^{\otimes k}\|^2 - 2\left\langle x^{\otimes k}, y^{\otimes k} \right\rangle + \|y^{\otimes k}\|_F^2 \\ &\quad = \|x\|^{2k} - 2\langle x, y \rangle^k + (\|x\|^2 + 2\gamma + \|x - y\|^2)^k \\ &\quad = \|x\|^{2k} - 2\left(\|x\|^2 + \gamma\right)^k + \left(\|x\|^2 + 2\gamma + \|x - y\|^2\right)^k \\ &\quad = \|x\|^{2k} - 2\sum_{j=0}^{k} \binom{k}{j} \|x\|^{2(k-j)} \gamma^j + \sum_{j=0}^{m} \binom{k}{j} \|x\|^{2(k-j)} (2\gamma + \|x - y\|^2)^j \end{aligned}$$

$$
\begin{aligned}
&= \|x\|^{2k} - 2\left(\|x\|^{2k} + k\|x\|^{2k-2}\gamma + \sum_{j=2}^{k}\binom{k}{j}\|x\|^{2(k-j)}\gamma^j\right) \\
&\qquad + \left(\|x\|^{2k} + k\|x\|^{2k-2}(2\gamma + \|x-y\|^2)\right. \\
&\qquad\quad \left. + \sum_{j=2}^{k}\binom{k}{j}\|x\|^{2(k-j)}(2\gamma + \|x-y\|^2)^j\right) \\
&= -2\gamma^2\sum_{j=2}^{k}\binom{k}{j}\underbrace{\|x\|^{2(k-j)}\gamma^{j-2}}_{\geq -1} + k\underbrace{\|x\|^{2k-2}}_{\leq 1}\|x-y\|^2 \\
&\qquad + (2\gamma + \|x-y\|^2)^2\sum_{j=2}^{k}\underbrace{\|x\|^{2(k-j)}(2\gamma + \|x-y\|^2)^{j-k}}_{\leq 1} \\
&\leq 2\gamma^2\cdot 2^k + k\|x-y\|^2 + \left(4\gamma^2 + 4\gamma\|x-y\|^2 + \|x-y\|^4\right)2^k \\
&\leq (11\cdot 2^k + k)\|x-y\|^2 \\
&\leq 12\cdot 2^k\|x-y\|^2.
\end{aligned}
$$

In the third line from the bottom we are using the fact that the sum of the binomial coefficients is equal to 2^m. ■

Definitions A.5.3. Let p and q be the densities of two measures P and Q that are absolutely continuous with respect to the Lebesgue measure. The χ^2 divergence, KL divergence, and TV distance are defined as

$$
\begin{aligned}
\chi^2(P\|Q) &= \int \frac{(p(x)-q(x))^2}{q(x)}\,dx \\
\mathrm{KL}(P\|Q) &= \int p(x)\log\left(\frac{p(x)}{q(x)}\right)\,dx \\
\mathrm{TV}(P,Q) &= \sup_A |P(A) - Q(A)| = \sup\left|\int_A (p(x)-q(x))\,dx\right|.
\end{aligned}
$$

In the definition of the TV norm, the supremum is over all measurable sets.

Lemma A.5.4. *We have the bound*

$$\mathrm{KL}(P\|Q) \leq \chi^2(P\|Q). \tag{A.5.5}$$

Proof. This is an immediate consequence of the fact that $\log(x) \leq x - 1$. ■

Lemma A.5.6 (Pinsker's Inequality, Lemma 2.5 in [156])**.** *For any two measures P and Q,*

$$\mathrm{TV}(P, Q) \leq \sqrt{\mathrm{KL}(P\|Q)/2}. \tag{A.5.7}$$

Lemma A.5.8. *Let P and Q be random variables and $P^{\otimes N}$ and $Q^{\otimes N}$ denote the distribution of N independent draws from P and Q. Then $\mathrm{KL}(P^{\otimes N}\|Q^{\otimes N}) = N\,\mathrm{KL}(P\|Q)$.*

Proof. For notational clarity, we only prove the case where $N = 2$. Then

$$\begin{aligned}
&\mathrm{KL}(P^{\otimes 2}\|Q^{\otimes 2}) \\
&\quad= \int_Y \int_X p(x)p(y) \log\left(\frac{p(x)p(y)}{q(x)q(y)}\right) dxdy \\
&\quad= \int_Y p(y) \int_X p(x) \left[\log\left(\frac{p(x)}{q(x)}\right) + \log\left(\frac{p(y)}{q(y)}\right)\right] dxdy \\
&\quad= \int_Y p(y) \int_X p(x) \log\left(\frac{p(x)}{q(x)}\right) dxdy + \int_Y p(y) \int_X p(x) \log\left(\frac{p(y)}{q(y)}\right) dxdy \\
&\quad= \int_X p(x) \log\left(\frac{p(x)}{q(x)}\right) dx + \int_Y p(y) \log\left(\frac{p(y)}{q(y)}\right) dy \\
&\quad= 2\,\mathrm{KL}(P\|Q).
\end{aligned}$$

■

Lemma A.5.9. *Assume that $c_0, \ldots, c_M \in \mathbb{R}^d$ are in generic position so that any collection*

of M of the c_i's is linearly independent. Let F^m be the matrix given by

$$F^m = \begin{cases} \frac{1}{4}p_1(c_0+c_1)^{\otimes 2} + \frac{1}{2}p_1 {c_0}^{\otimes 2} & \text{if } m = 0 \\ \frac{1}{4}p_m(c_{m-1}+c_m)^{\otimes 2} + \frac{1}{4}p_{m+1}(c_m+c_{m+1})^{\otimes 2} & \\ \quad + \frac{1}{2}p_m {c_m}^{\otimes 2} + \frac{1}{2}p_{m+1}{c_m}^{\otimes 2} & \text{if } 1 \le m \le M-1 \\ \frac{1}{4}p_M(c_{M-1}+c_M)^{\otimes 2} + \frac{1}{2}p_M {c_M}^{\otimes 2} & \text{if } m = M. \end{cases}$$

Then $\{F^m\}_{m=0}^M$ is a linearly independent set.

Proof. We want to show that if $T := \sum_{m=0}^M k_m F^m = 0$, then $k_m = 0$ for all $m = 0, \ldots, M$. We can reindex the sum to write

$$T = \sum_{i=1}^{M} \frac{k_{i-1}+k_i}{4} p_i (c_{i-1}+c_i)^{\otimes 2} + \frac{k_{i-1}}{2} p_i c_{i-1}^{\otimes 2} + \frac{k_i}{2} p_i c_i^{\otimes 2} \tag{A.5.10}$$

For $j = 2, \ldots, M-1$, let x_j be a vector in $\mathbb{R}^d$ such that $x_j \perp c_i$ for all $i \neq j-1, j$. Let Tx_j denote the tensor contraction along the last index of T. Then Tx_j kills a large number of terms and we have

$$\begin{aligned} Tx_j = &\frac{k_{j-2}+k_{j-1}}{4} p_{j-1}\langle c_{j-1}, x_j\rangle (c_{j-2}+c_{j-1}) \\ &+ \frac{k_{j-1}+k_j}{4} p_j \langle c_{j-1}+c_j, x_j\rangle (c_{j-1}+c_j) \\ &+ \frac{k_j+k_{j+1}}{4} p_{j+1}\langle c_j, x_j\rangle (c_j+c_{j+1}) \\ &+ \frac{k_{j-1}}{2} p_j \langle c_{j-1}, x\rangle c_{j-1} + \frac{k_j}{2} p_{j+1}\langle c_j, x\rangle c_j \\ &+ \frac{k_{j-1}}{2} p_{j-1}\langle c_{j-1}, x\rangle c_{j-1} + \frac{k_j}{2} p_j \langle c_j, x\rangle c_j \end{aligned}$$

We can rewrite this as $Tx_j = \alpha_{j-2}c_{j-2} + \alpha_{j-1}c_{j-1} + \alpha_j c_j + \alpha_{j+1}c_{j+1}$, where

$$\begin{aligned} \alpha_{j-2} &= \frac{k_{j-2}+k_{j-1}}{4} p_{j-1}\langle c_{j-1}, x_j\rangle \\ \alpha_{j-1} &= \frac{k_{j-2}+k_{j-1}}{4} p_{j-1}\langle c_{j-1}, x_j\rangle + \frac{k_{j-1}+k_j}{4} p_j \langle c_{j-1}+c_j, x_j\rangle \end{aligned}$$

$$
\begin{aligned}
&\quad + \frac{k_{j-1}}{2} p_j \langle c_{j-1}, x \rangle + \frac{k_{j-1}}{2} p_{j-1} \langle c_{j-1}, x \rangle \\
\alpha_j &= \frac{k_{j-1} + k_j}{4} p_j \langle c_{j-1} + c_j, x_j \rangle + \frac{k_j + k_{j+1}}{4} p_{j+1} \langle c_j, x_j \rangle \\
&\quad + \frac{k_j}{2} p_{j+1} \langle c_j, x \rangle + \frac{k_j}{2} p_j \langle c_j, x \rangle \\
\alpha_{j+1} &= \frac{k_j + k_{j+1}}{4} p_{j+1} \langle c_j, x_j \rangle
\end{aligned}
$$

By our assumption of genericness, $c_{j-2}, \ldots, c_{j+1}$ are linearly independent. Note also that x_j is not in the span of c_{j-2} and c_{j+1} by our definition of x_j. Therefore, if the linear combination of the four terms above is to be zero, that means we must have $\alpha_{j-2}, \alpha_{j+1} = 0$. Our genericness assumption also implies that $\langle c_{j-1}, x_j \rangle$ and $\langle c_j, x_j \rangle$ are nonzero. Note also that all p_i are nonzero (positive, in fact). Therefore, if $\alpha_{j-2}, \alpha_{j+1} = 0$, then it must follow that $k_{j-2} + k_{j-1} = 0$ and $k_j + k_{j+1} = 0$. Repeat this for $j = 2, \ldots, M-1$. Then if $\sum_{m=0}^{M} k_m F^m = 0$, then (A.5.10) becomes

$$
\sum_{i=1}^{M} \left(\frac{k_{i-1}}{2} p_i c_{i-1}^{\otimes 2} + \frac{k_i}{2} p_i c_i^{\otimes 2} \right) = 0.
$$

We've thus reduced our problem to whether $c_0^{\otimes 2}, \ldots, c_M^{\otimes 2}$ is a linearly independent set.

Consider the sum

$$
\sum_{i=0}^{M} \gamma_i c_i^{\otimes 2} \tag{A.5.11}
$$

where γ_i are scalars. We can rewrite this as

$$
\gamma_0 c_0^{\otimes 2} + \sum_{i=1}^{M} \gamma_i c_i^{\otimes 2}. \tag{A.5.12}
$$

Suppose for the sake of contradiction that (A.5.12) is equal to zero and not all γ_i are equal to zero; assume without loss of generality that γ_0 is nonzero. Since $c_1, \ldots, c_M$ are linearly independent, the summation is rank r, where r is the number of nonzero coefficients $\gamma_1, \ldots, \gamma_M$, so (A.5.12) is the sum of a rank 1 and rank r matrix. If $r > 1$, then this sum cannot possibly

be zero. If $r = 1$, suppose without loss of generality that γ_1 is the only nonzero coefficient among $\gamma_1, \ldots, \gamma_M$, so that $\gamma_0 c_0^{\otimes 2} + \gamma_1 c_1^{\otimes 2} = 0$. Then c_0 must be parallel to c_1, but this is not possible by our assumption that c_i are generic. ■

APPENDIX B

SUPPLEMENTARY MATERIAL TO CHAPTER 4

B.1 Miscellaneous Implementation Details

B.1.1 Hypernetwork Implementation Details

As described in Equation 4.3.3 in Section 4.3.4, the hypernetwork is a function H_ϕ that maps a noised MDF $\mathbf{F}_t(Q)$, a set of query points Q, and a diffusion time step t to the parameters θ of an MNF $\mathbf{F}_\theta$. These inputs are tokenized as follows. To encode the input MDF $\mathbf{F}_t(Q)$, we begin by drawing a bounding box around the molecule and partitioning the enclosed volume into a uniform $c \times c \times c$ 3D grid. A fixed number of query points are uniformly sampled within each cell. For each query point $\mathbf{q}_j$, we first apply a learned positional encoding to its 3D coordinate, then concatenate the resulting embedding with the corresponding noised field vector $\mathbf{F}_t(\mathbf{q}_j)$. These encoded query points form the input tokens to the transformer-based hypernetwork. The diffusion time step t is embedded using sinusoidal Fourier features, as in [68], and broadcast across all tokens. The transformer processes the sequence using self-attention layers to capture spatial dependencies across grid cells, followed by MLPs. The final token representations are passed through linear projections to produce the complete set of MLP weights θ for the downstream field $\mathbf{F}_\theta$.

B.1.2 Training Curriculum

We train with a curriculum on the noise level. For the first 100 epochs of training, the maximum noise level t that we add to a sample is 10 (out of a total of $T = 1000$ noise levels). For every subsequent epoch, the maximum possible noise level is increased by 1, until $T = 1000$ is reached; we find that this accelerates training in practice; we show ablation studies in §B.2.1.

B.1.3 Hyperparameters

We use the LAMB optimizer [157] with a learning rate of 1×10^{-2}, a batch size of 5500, and dropout set to 0.1. Both the noise schedule and the learning rate follow a cosine decay. The hypernetwork consists of 26 transformer layers and contains approximately 52 million parameters. All diffusion models are trained on NVIDIA H100 GPUs for 3500 epochs.

B.2 Ablation Studies

We conduct two ablation experiments to better understand key design choices in HyDiF: (1) the impact of using a noise-level curriculum during training, and (2) the effect of the input field type provided to the hypernetwork.

B.2.1 Effect of Noise Scale Curriculum

In Figure B.1, we compare training with and without a noise-level curriculum. In the curriculum setting, the maximum diffusion noise level is gradually increased over the course of training, starting from low noise levels and eventually reaching pure noise. This strategy leads to faster convergence and improved training stability compared to the baseline with a fixed maximum noise level throughout. Notably, we observe a rise in training loss around epoch 800 in the curriculum setting, which coincides with the introduction of highly noised inputs that are intrinsically more difficult to denoise.

B.2.2 Effect of Input Field Type

In Figure B.2, we assess the impact of the type of input field (either a distance field or a direction field) provided to the hypernetwork. We find that using a direction field as input leads to lower training loss and more effective learning. We attribute this improvement to the fact that direction fields contain higher-frequency signals compared to distance fields, providing a more expressive representation of local molecular structure.

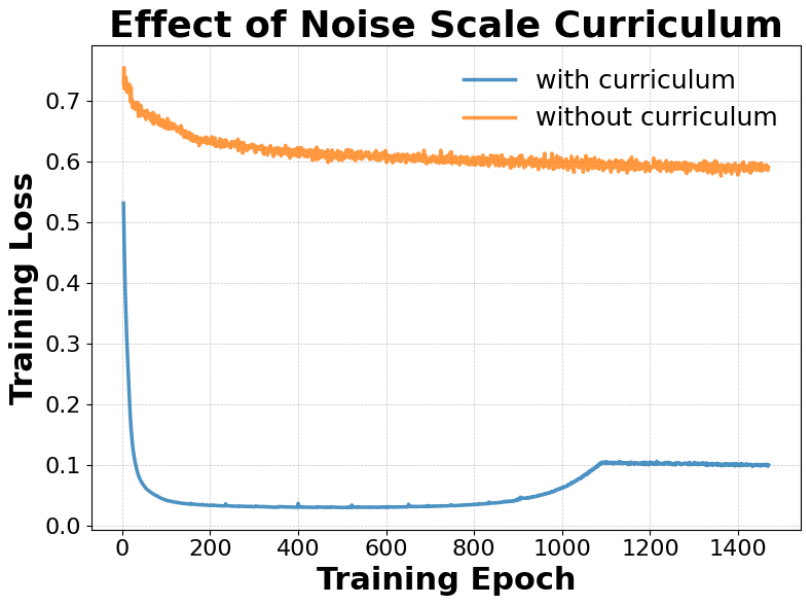

Figure B.1: Ablation study illustrating the benefit of using a training curriculum (blue) on the diffusion noise scale versus no curriculum (orange). The curriculum improves convergence and early performance by exposing the model to gradually harder denoising tasks. The rise in loss near epoch 800 reflects the model being trained on highly noised inputs.

In both experiments, we report ℓ_1 loss of the field on the subset of query points that are *very close* to the ground truth atomic coordinates—defined as the top 2% of query points with the smallest distance to any atom. These points represent the highest-frequency components of the MDF, and accurately modeling them is critical for capturing precise atomic structure.

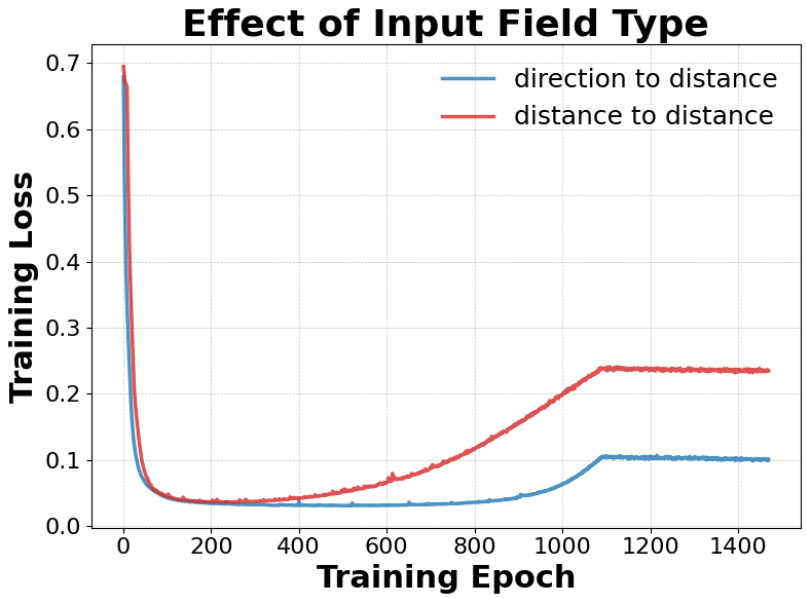

Figure B.2: Ablation study comparing the effect of input field type. Using a *direction* field (blue) as input leads to improved training over a *distance* field (red), likely due to the richer high-frequency information present in direction fields.

BIBLIOGRAPHY

[1] David A. King. "The Encyclopaedia of Islam, Second Edition". In: vol. 5. Leiden: Brill, 2006. Chap. Kibla: Astronomical Aspects. DOI: 10.1163/1573-3912_islam_COM_0513.

[2] J. L. Russell. "Kepler's Laws of Planetary Motion: 1609-1666". In: *The British Journal for the History of Science* 2.1 (1964), pp. 1–24. ISSN: 00070874, 1474001X.

[3] Nicolas Bacaër. *A Short History of Mathematical Population Dynamics*. Springer London, 2011. ISBN: 9780857291158. DOI: 10.1007/978-0-85729-115-8.

[4] J. Dubochet and A.W. McDowall. "Vitrification of pure water for electron microscopy". In: *Journal of microscopy* 124.3 (1981), pp. 3–4.

[5] Steven J Ludtke et al. "Seeing GroEL at 6 Å Resolution by Single Particle Electron Cryomicroscopy". In: *Structure* 12.7 (2004), pp. 1129–1136. ISSN: 0969-2126. DOI: 10.1016/j.str.2004.05.006.

[6] C. Chaudhry et al. *GroEL: 1ss8*. Mar. 2004. DOI: 10.2210/pdb1ss8/pdb.

[7] Michelle W. Y. Southey and Michael Brunavs. "Introduction to small molecule drug discovery and preclinical development". In: *Frontiers in Drug Discovery* Volume 3 - 2023 (2023). ISSN: 2674-0338. DOI: 10.3389/fddsv.2023.1314077.

[8] Regine S. Bohacek, Colin McMartin, and Wayne C. Guida. "The art and practice of structure-based drug design: A molecular modeling perspective". In: *Medicinal Research Reviews* 16.1 (1996), pp. 3–50. DOI: 10.1002/(SICI)1098-1128(199601)16:1<3::AID-MED1>3.0.CO;2-6.

[9] Jacqueline Kuan et al. "Keeping pace with the explosive growth of chemical libraries with structure-based virtual screening". In: *WIREs Computational Molecular Science* 13.6 (2023), e1678. DOI: 10.1002/wcms.1678.

[10] K.-P. Schröder and Robert Connon Smith. "Distant future of the Sun and Earth revisited". In: *Monthly Notices of the Royal Astronomical Society* 386.1 (Mar. 2008), pp. 155–163. ISSN: 0035-8711. DOI: 10.1111/j.1365-2966.2008.13022.x.

[11] Phillip Lo and Yuehaw Khoo. *Method of Moments for Estimation of Noisy Curves*. 2024. arXiv: 2410.23220 [math.ST]. URL: https://arxiv.org/abs/2410.23220.

[12] Marina Meilă and Hanyu Zhang. "Manifold Learning: What, How, and Why". In: *Annual Review of Statistics and Its Application* 11 (Apr. 2024), pp. 393–417. DOI: 10.1146/annurev-statistics-040522-115238.

[13] Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. "Testing the Manifold Hypothesis". In: *Journal of the American Mathematical Society* 29.4 (Feb. 2016), pp. 983–1049. DOI: 10.1090/jams/852.

[14] Kitty Mohammed and Hariharan Narayanan. *Manifold Learning Using Kernel Density Estimation and Local Principal Components Analysis*. 2017. arXiv: 1709.03615 [math.ST]. URL: https://arxiv.org/abs/1709.03615.

[15] Zhigang Yao and Yuqing Xia. *Manifold Fitting under Unbounded Noise*. 2024. arXiv: 1909.10228 [stat.ML]. URL: https://arxiv.org/abs/1909.10228.

[16] Charles Fefferman et al. "Fitting a Putative Manifold to Noisy Data". In: *Proceedings of the 31st Conference On Learning Theory*. Vol. 75. Proceedings of Machine Learning Research. PMLR, July 2018, pp. 688–720.

[17] Hau-Tieng Wu. *Design a Metric Robust to Complicated High Dimensional Noise for Efficient Manifold Denoising*. 2024. arXiv: 2401.03921 [stat.ML]. URL: https://arxiv.org/abs/2401.03921.

[18] Dayang Wang et al. *Manifoldron: Direct Space Partition via Manifold Discovery*. 2022. arXiv: 2201.05279 [cs.LG]. URL: https://arxiv.org/abs/2201.05279.

[19] Boris Landa and Xiuyuan Cheng. "Robust Inference of Manifold Density and Geometry by Doubly Stochastic Scaling". In: *SIAM Journal on Mathematics of Data Science* 5.3 (July 2023), pp. 589–614. DOI: 10.1137/22M1516968.

[20] Charles Fefferman et al. *Fitting a manifold to data in the presence of large noise*. 2023. arXiv: 2312.10598 [math.ST]. URL: https://arxiv.org/abs/2312.10598.

[21] Amit Singer and Fred J. Sigworth. "Computational methods for Single-Particle Electron Cryomicroscopy". In: *Annual Review of Biomedical Data Science* 3.1 (July 2020), pp. 163–190. DOI: 10.1146/annurev-biodatasci-021020-093826.

[22] Amelia Perry et al. "The Sample Complexity of Multireference Alignment". In: *SIAM Journal on Mathematics of Data Science* 1.3 (2019), pp. 497–517. DOI: 10.1137/18M1214317.

[23] Amit Moscovich et al. "Cryo-EM Reconstruction of Continuous Heterogeneity by Laplacian Spectral Volumes". In: *Inverse Problems* 36.2 (Jan. 2020), p. 024003. DOI: 10.1088/1361-6420/ab4f55.

[24] Roy R. Lederman, Joakim Andén, and Amit Singer. "Hyper-molecules: on the Representation and Recovery of Dynamical Structures for Applications in Flexible Macromolecules in Cryo-EM". In: *Inverse Problems* 36.4 (Mar. 2020), p. 044005. DOI: 10.1088/1361-6420/ab5ede.

[25] Alan E. Gelfand. "Seriation Methods for Archaeological Materials". In: *American Antiquity* 36.3 (July 1971), pp. 263–274. DOI: 10.2307/277714.

[26] Jonathan E. Atkins, Erik G. Boman, and Bruce Hendrickson. "A Spectral Algorithm for Seriation and the Consecutive Ones Problem". In: *SIAM Journal on Computing* 28.1 (1998), pp. 297–310. DOI: 10.1137/S0097539795285771.

[27] Fajwel Fogel, Alexandre d'Aspremont, and Milan Vojnovic. "SerialRank: Spectral Ranking Using Seriation". In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc., 2014.

[28] Philippe Rigollet and Jonathan Weed. "Uncoupled Isotonic Regression via Minimum Wasserstein deconvolution". In: *Information and Inference: A Journal of the IMA* 8.4 (Apr. 2019), pp. 691–717. DOI: 10.1093/imaiai/iaz006.

[29] Yuehaw Khoo et al. *Temporal label recovery from noisy dynamical data*. 2024. arXiv: 2406.13635 [stat.ME]. URL: https://arxiv.org/abs/2406.13635.

[30] João M. Pereira, Joe Kileel, and Tamara G. Kolda. *Tensor Moments of Gaussian Mixture Models: Theory and Applications*. 2022. arXiv: 2202.06930 [stat.ML]. URL: https://arxiv.org/abs/2202.06930.

[31] Steven J. Gortler, Alexander D. Healy, and Dylan P. Thurston. *Characterizing Generic Global Rigidity*. 2010. arXiv: 0710.0926 [math.MG]. URL: %7Bhttps://arxiv.org/abs/0710.0926%7D.

[32] Boris Mityagin. *The Zero Set of a Real Analytic Function*. 2015. arXiv: 1512.07276 [math.CA]. URL: https://arxiv.org/abs/1512.07276.

[33] Björn Holmquist. "Moments and Cumulants of the Multivariate Normal Distribution". In: *Stochastic Analysis and Applications* 6.3 (Jan. 1988), pp. 273–278. DOI: 10.1080/07362998808809148.

[34] Animashree Anandkumar et al. "Tensor decompositions for learning latent variable models". In: *J. Mach. Learn. Res.* 15.1 (Jan. 2014), pp. 2773–2832. ISSN: 1532-4435.

[35] James Bradbury et al. *JAX: Composable Transformations of Python + NumPy programs*. Version 0.3.13. 2018.

[36] Mathieu Blondel et al. *Efficient and Modular Implicit Differentiation*. 2022. arXiv: 2105.15183 [cs.LG]. URL: https://arxiv.org/abs/2105.15183.

[37] Cedric Villani. *Optimal Transport*. Grundlehren der mathematischen Wissenschaften. Berlin, Germany: Springer, Dec. 2009.

[38] Filippo Santambrogio. *Optimal Transport for Applied Mathematicians: Calculus of Variations, PDEs, and Modeling*. Springer International Publishing, 2015. ISBN: 9783319208282. DOI: 10.1007/978-3-319-20828-2.

[39] Justin Solomon. *Optimal Transport on Discrete Domains*. 2018. arXiv: 1801.07745 [math.OC]. URL: https://arxiv.org/abs/1801.07745.

[40] Jozef Strecka and Michal Jascur. *A brief account of the Ising and Ising-like models: Mean-field, effective-field and exact results*. 2015. arXiv: 1511.03031 [cond-mat.stat-mech]. URL: https://arxiv.org/abs/1511.03031.

[41] Roy J. Glauber. "Time-Dependent Statistics of the Ising Model". In: *Journal of Mathematical Physics* 4.2 (Feb. 1963), pp. 294–307. ISSN: 0022-2488, 1089-7658. DOI: 10.1063/1.1703954.

[42] Mário J. De Oliveira. "Linear Glauber model". In: *Physical Review E* 67.6 (June 2003), p. 066101. ISSN: 1063-651X, 1095-3787. DOI: 10.1103/PhysRevE.67.066101.

[43] Shaon Sahoo and Soumya Kanti Ganguly. "Optimal Linear Glauber Model". In: *Journal of Statistical Physics* 159.2 (Apr. 2015), pp. 336–357. ISSN: 0022-4715, 1572-9613. DOI: 10.1007/s10955-015-1188-y.

[44] Jean-David Benamou and Yann Breiner. "A Computational Fluid Mechanics Solution to the Monge-Kantorovich Mass Transfer Problem". In: *Numerische Mathematik* 84 (Jan. 2000), pp. 375–393. DOI: 10.1007/s002110050002.

[45] Benedetto Piccoli and Francesco Rossi. "On Properties of the Generalized Wasserstein Distance". In: *Archive for Rational Mechanics and Analysis* 222 (July 2016), pp. 1339–1364. DOI: 10.1007/s00205-016-1026-7.

[46] Lorenzo Brasco. "A Survey on Dynamical Transport Distances". In: *Journal of Mathematical Sciences* 181 (Mar. 2012), pp. 755–781. DOI: 10.1007/s10958-012-0713-7.

[47] L. V. Kantorovich. "Mathematical Methods of Organizing and Planning Production". In: *Management Science* 6.4 (July 1960), pp. 366–422. ISSN: 1526-5501. DOI: 10.1287/mnsc.6.4.366.

[48] George B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1991.

[49] Robert J. Vanderbei. *Linear Programming: Foundations and Extensions*. Springer International Publishing, 2020. ISBN: 9783030394158. DOI: 10.1007/978-3-030-39415-8.

[50] Renato D. C. Monteiro and Ilan Adler. "Interior Path Following Primal-dual algorithms. Part I: Linear Programming". In: *Mathematical Programming* 44.1-3 (May 1989), pp. 27–41. ISSN: 0025-5610, 1436-4646. DOI: 10.1007/BF01587075.

[51] Haihao Lu. *First-Order Methods for Linear Programming*. 2024. arXiv: 2403.14535 [math.OC]. URL: https://arxiv.org/abs/2403.14535.

[52] David Applegate et al. "Practical Large-Scale Linear Programming using Primal-Dual Hybrid Gradient". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 20243–20257.

[53] Vahab Mirrokni. *Google Research, 2022 & Beyond: Algorithmic Advances*. https://ai.googleblog.com/2023/02/google-research-2022-beyond-algorithmic.html. 2023-02-10.

[54] Mingqiang Zhu and Tony Chan. "An Efficient Primal-dual Hybrid Gradient Algorithm for Total Variation Image Restoration". In: *UCLA CAM Reports* 34 (2008), pp. 8–34.

[55] Antonin Chambolle and Thomas Pock. "A First-order Primal-dual Algorithm for Convex Problems with Applications to Imaging". In: *Journal of Mathematical Imaging and Vision* 40 (2011), pp. 120–145.

[56] Samuel Kotz, N. Balakrishnan, and Norman L. Johnson. *Continuous Multivariate Distributions: Models and Applications*. Wiley, Apr. 2000. ISBN: 9780471722069. DOI: 10.1002/0471722065.

[57] Yian Chen, Yuehaw Khoo, and Lek-Heng Lim. *Convex Relaxation for Fokker-Planck*. 2023. arXiv: 2306.03292 [math.NA]. URL: https://arxiv.org/abs/2306.03292.

[58] Kurt Binder and Erik Luijten. "Monte Carlo Tests of Renormalization-group Predictions for Critical Phenomena in Ising Models". In: *Physics Reports* 344.4 (2001), pp. 179–253. ISSN: 0370-1573. DOI: 10.1016/S0370-1573(00)00127-7.

[59] Michael M. Bronstein et al. "Geometric Deep Learning: Going Beyond Euclidean Data". In: *IEEE Signal Processing Magazine* 34.4 (2017), pp. 18–42.

[60] R. Qi Charles et al. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 77–85. DOI: 10.1109/CVPR.2017.16.

[61] Yin Zhou and Oncel Tuzel. "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4490–4499. DOI: 10.1109/CVPR.2018.00472.

[62] Hengshuang Zhao et al. "Point Transformer". In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 16239–16248. DOI: 10.1109/ICCV48922.2021.01595.

[63] Rana Hanocka et al. "MeshCNN: a network with an edge". In: *ACM Trans. Graph.* 38.4 (July 2019). ISSN: 0730-0301. DOI: 10.1145/3306346.3322959.

[64] Patrick Reiser et al. "Graph Neural Networks for Materials Science and Chemistry". In: *Communications Materials* 3.1 (2022), p. 93.

[65] Amil Merchant et al. "Scaling deep learning for materials discovery". In: *Nature* 624.7990 (2023), pp. 80–85.

[66] Bojun Liu et al. *Voxel-based Point Cloud Geometry Compression with Space-to-Channel Context*. 2025. arXiv: 2503.18283 [cs.CV]. URL: https://arxiv.org/abs/2503.18283.

[67] Vincent Sitzmann et al. "Implicit neural representations with periodic activation functions". In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NeurIPS '20. Vancouver, BC, Canada: Curran Associates Inc., 2020. ISBN: 9781713829546.

[68] Ben Mildenhall et al. "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis". In: *Communications of the ACM* 65.1 (Dec. 2021), pp. 99–106. ISSN: 0001-0782. DOI: 10.1145/3503250.

[69] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. "Decomposing NeRF for editing via feature field distillation". In: *Proceedings of the 36th International Conference on Neural Information Processing Systems*. NeurIPS '22. New Orleans, LA, USA: Curran Associates Inc., 2022. ISBN: 9781713871088.

[70] Itai Lang et al. "iSeg: Interactive 3D Segmentation via Interactive Attention". In: *SIGGRAPH Asia 2024 Conference Papers*. SA '24. Tokyo, Japan: Association for Computing Machinery, 2024. ISBN: 9798400711312. DOI: 10.1145/3680528.3687605.

[71] Gabriele Corso et al. *DiffDock: Diffusion Steps, Twists, and Turns for Molecular Docking*. 2023. arXiv: 2210.01776 [q-bio.BM]. URL: https://arxiv.org/abs/2210.01776.

[72] Jiaqi Guan et al. *3D Equivariant Diffusion for Target-Aware Molecule Generation and Affinity Prediction*. 2023. arXiv: 2303.03543 [q-bio.BM]. URL: https://arxiv.org/abs/2303.03543.

[73] Arne Schneuing et al. *Structure-based Drug Design with Equivariant Diffusion Models*. 2024. arXiv: 2210.13695 [q-bio.BM]. URL: https://arxiv.org/abs/2210.13695.

[74] Sudarshan Babu et al. “HyperFields: Towards zero-shot generation of NeRFs from text”. In: *Proceedings of the 41st International Conference on Machine Learning*. ICML’24. Vienna, Austria: JMLR.org, 2024.

[75] M. Przewięźlikowski et al. *HyperMAML: Few-Shot Adaptation of Deep Models with Hypernetworks*. 2024. arXiv: 2205.15745 [cs.LG]. URL: https://arxiv.org/abs/2205.15745.

[76] Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising diffusion probabilistic models”. In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NeurIPS ’20. Vancouver, BC, Canada: Curran Associates Inc., 2020. ISBN: 9781713829546.

[77] Oriel Frigo, Rémy Brossard, and David Dehaene. *Graph Context Encoder: Graph Feature Inpainting for Graph Generation and Self-supervised Pretraining*. 2021. arXiv: 2106.10124 [cs.LG]. URL: https://arxiv.org/abs/2106.10124.

[78] Thomas E. Hadfield et al. “Incorporating Target-Specific Pharmacophoric Information into Deep Generative Models for Fragment Elaboration”. In: *Journal of Chemical Information and Modeling* 62.10 (2022). PMID: 35499971, pp. 2280–2292. DOI: 10.1021/acs.jcim.1c01311.

[79] Maxime Langevin et al. “Scaffold-Constrained Molecular Generation”. In: *Journal of Chemical Information and Modeling* 60.12 (2020). PMID: 33301333, pp. 5637–5646. DOI: 10.1021/acs.jcim.0c01015.

[80] Krzysztof Maziarz et al. *Learning to Extend Molecular Scaffolds with Structural Motifs*. 2024. arXiv: 2103.03864 [cs.LG]. URL: https://arxiv.org/abs/2103.03864.

[81] Xiao Zhang et al. *Residual Connections Harm Generative Representation Learning*. 2025. arXiv: 2404.10947 [cs.CV]. URL: https://arxiv.org/abs/2404.10947.

[82] Xiao Zhang and Michael Maire. *Structural Adversarial Objectives for Self-Supervised Representation Learning*. 2023. arXiv: 2310.00357 [cs.CV]. URL: https://arxiv.org/abs/2310.00357.

[83] Xiao Zhang, David Yunis, and Michael Maire. *Deciphering 'What' and 'Where' Visual Pathways from Spectral Clustering of Layer-Distributed Neural Representations*. 2024. arXiv: 2312.06716 [cs.CV]. URL: https://arxiv.org/abs/2312.06716.

[84] Xingyi Yang and Xinchao Wang. "Diffusion Model as Representation Learner". In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023, pp. 18892–18903. DOI: 10.1109/ICCV51070.2023.01736.

[85] Tianhong Li et al. "MAGE: MAsked Generative Encoder to Unify Representation Learning and Image Synthesis". In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 2142–2152. DOI: 10.1109/CVPR52729.2023.00213.

[86] Laurianne David et al. "Molecular representations in AI-driven drug discovery: a review and practical guide". In: *Journal of Cheminformatics* 12.1 (Sept. 2020), p. 56. ISSN: 1758-2946. DOI: 10.1186/s13321-020-00460-5.

[87] David Weininger. "SMILES, a Chemical Language and Information System. 1. Introduction to Methodology and Encoding Rules". In: *Journal of Chemical Information and Computer Sciences* 28.1 (1988), pp. 31–36. DOI: 10.1021/ci00057a005.

[88] David Weininger, Arthur Weininger, and Joseph L. Weininger. "SMILES. 2. Algorithm for Generation of Unique SMILES Notation". In: *Journal of Chemical Information and Computer Sciences* 29.2 (1989), pp. 97–101. DOI: 10.1021/ci00062a008.

[89] David Weininger. "SMILES. 3. DEPICT. Graphical Depiction of Chemical Structures". In: *Journal of Chemical Information and Computer Sciences* 30.3 (1990), pp. 237–243. DOI: 10.1021/ci00067a005.

[90] Stephen R Heller et al. “InChI, the IUPAC International Chemical Identifier”. In: *Journal of Cheminformatics* 7 (May 2015). ISSN: 1758-2946.

[91] Mario Krenn et al. “Self-referencing Embedded Strings (SELFIES): A 100% Robust Molecular String Representation”. In: *Machine Learning: Science and Technology* 1.4 (Oct. 2020), p. 045024. DOI: 10.1088/2632-2153/aba947.

[92] Mario Krenn et al. “SELFIES and the Future of Molecular String Representations”. In: *Patterns* 3.10 (2022), p. 100588. ISSN: 2666-3899. DOI: 10.1016/j.patter.2022.100588.

[93] Christiane O. Dietrich-Buchecker and Jean-Pierre Sauvage. “A Synthetic Molecular Trefoil Knot”. In: *Angewandte Chemie International Edition in English* 28.2 (Feb. 1989), pp. 189–192. ISSN: 0570-0833. DOI: 10.1002/anie.198901891.

[94] Stephen D. P. Fielden, David A. Leigh, and Steffen L. Woltering. “Molecular Knots”. In: *Angewandte Chemie International Edition* 56.37 (Aug. 2017), pp. 11166–11194. ISSN: 1521-3773. DOI: 10.1002/anie.201702531.

[95] M. S. Smyth and J. H. Martin. “X-ray Crystallography”. In: *Molecular Pathology* 53.1 (Feb. 2000), pp. 8–14. ISSN: 1366-8714. DOI: 10.1136/mp.53.1.8.

[96] Cláudio F. Tormena. “Conformational Analysis of Small Molecules: NMR and Quantum Mechanics Calculations”. In: *Progress in Nuclear Magnetic Resonance Spectroscopy* 96 (2016), pp. 73–88. ISSN: 0079-6565. DOI: 10.1016/j.pnmrs.2016.04.001.

[97] Thomas A. Halgren. “Merck Molecular Force Field. I. Basis, Form, scope, Parameterization, and Performance of MMFF94”. In: *Journal of Computational Chemistry* 17.5-6 (1996), pp. 490–519. DOI: 10.1002/(SICI)1096-987X(199604)17:5/6<490::AID-JCC1>3.0.CO;2-P.

[98] Philipp Pracht, Fabian Bohle, and Stefan Grimme. “Automated Exploration of the Low-energy Chemical Space with Fast Quantum Chemical Methods”. In: *Physical*

Chemistry Chemical Physics 22.14 (2020), pp. 7169–7192. ISSN: 1463-9084. DOI: 10.1039/c9cp06869d.

[99] Noel M O'Boyle et al. "Confab - Systematic Generation of Diverse Low-energy Conformers". In: *Journal of Cheminformatics* 3.1 (Mar. 2011). ISSN: 1758-2946. DOI: 10.1186/1758-2946-3-8.

[100] Paul C. D. Hawkins et al. "Conformer Generation with OMEGA: Algorithm and Validation Using High Quality Structures from the Protein Databank and Cambridge Structural Database". In: *Journal of Chemical Information and Modeling* 50.4 (2010). PMID: 20235588, pp. 572–584. DOI: 10.1021/ci100031x.

[101] Mikko J. Vainio and Mark S. Johnson. "Generating Conformer Ensembles Using a Multiobjective Genetic Algorithm". In: *Journal of Chemical Information and Modeling* 47.6 (2007). PMID: 17892278, pp. 2462–2474. DOI: 10.1021/ci6005646.

[102] J Santeri Puranen, Mikko J Vainio, and Mark S Johnson. "Accurate Conformation-dependent Molecular Electrostatic Potentials for High-throughput in Silico Drug Discovery". In: *Journal of Computational Chemistry* 31.8 (June 2010).

[103] Sereina Riniker and Gregory A. Landrum. "Better Informed Distance Geometry: Using What We Know To Improve Conformation Generation". In: *Journal of Chemical Information and Modeling* 55.12 (2015). PMID:26575315, pp. 2562–2574. DOI: 10.1021/acs.jcim.5b00654.

[104] *RDKit: Open-source Chemoinformatics*. 2025. URL: https://www.rdkit.org.

[105] Simon Axelrod and Rafael Gómez-Bombarelli. "GEOM, Energy-annotated Molecular Conformations for Property Prediction and Molecular Generation". In: *Scientific Data* 9.1 (Apr. 2022), p. 185. ISSN: 2052-4463. DOI: 10.1038/s41597-022-01288-4.

[106] Gregor Simm and Jose Miguel Hernandez-Lobato. "A Generative Model for Molecular Distance Geometry". In: *Proceedings of the 37th International Conference on Machine*

Learning. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, July 2020, pp. 8949–8958.

[107] Chence Shi et al. "Learning Gradient Fields for Molecular Conformation Generation". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meilă and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, July 2021, pp. 9558–9568.

[108] Minkai Xu et al. "An End-to-End Framework for Molecular Conformation Generation via Bilevel Programming". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meilă and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, July 2021, pp. 11537–11547.

[109] Shitong Luo et al. "Predicting Molecular Conformation via Dynamic Graph Score Matching". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 19784–19795.

[110] Elman Mansimov et al. "Molecular Geometry Prediction using a Deep Generative Graph Neural Network". In: *Scientific Reports* 9.1 (Dec. 2019). ISSN: 2045-2322. DOI: 10.1038/s41598-019-56773-5.

[111] Octavian-Eugen Ganea et al. "GeoMol: Torsional Geometric Generation of Molecular 3D Conformer Ensembles". In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer et al. 2021.

[112] Jinhua Zhu et al. *Direct Molecular Conformation Generation*. 2022. arXiv: 2202.01356 [cs.AI]. URL: https://arxiv.org/abs/2202.01356.

[113] Minkai Xu et al. "GeoDiff: A Geometric Diffusion Model for Molecular Conformation Generation". In: *International Conference on Learning Representations*. 2022.

[114] Kirk Swanson, Jake Lawrence Williams, and Eric M Jonas. "Von Mises Mixture Distributions for Molecular Conformation Generation". In: *Proceedings of the 40th In-*

ternational Conference on Machine Learning. Ed. by Andreas Krause et al. Vol. 202. Proceedings of Machine Learning Research. PMLR, July 2023, pp. 33319–33342.

[115] Toshiki Ochiai et al. "Variational autoencoder-based chemical latent space for large molecular structures with 3D complexity". In: *Communications Chemistry* 6.1 (Nov. 2023). ISSN: 2399-3669. DOI: 10.1038/s42004-023-01054-6.

[116] Jaechang Lim et al. "Molecular generative model based on conditional variational autoencoder for de novo molecular design". In: *Journal of Cheminformatics* 10.1 (July 2018). ISSN: 1758-2946. DOI: 10.1186/s13321-018-0286-7.

[117] Ryan J Richards and Austen M Groener. *Conditional β-VAE for De Novo Molecular Generation*. 2022. arXiv: 2205.01592 [q-bio.BM]. URL: https://arxiv.org/abs/2205.01592.

[118] Minkai Xu et al. *Learning Neural Generative Dynamics for Molecular Conformation Generation*. 2021. arXiv: 2102.10240 [cs.LG]. URL: https://arxiv.org/abs/2102.10240.

[119] Yiheng Zhu et al. *MolHF: A Hierarchical Normalizing Flow for Molecular Graph Generation*. 2023. arXiv: 2305.08457 [cs.LG]. URL: https://arxiv.org/abs/2305.08457.

[120] Eyal Rozenberg and Daniel Freedman. *Semi-Equivariant Continuous Normalizing Flows for Target-Aware Molecule Generation*. 2022. arXiv: 2211.04754 [cs.LG]. URL: https://arxiv.org/abs/2211.04754.

[121] Bruno Macedo, Inês Ribeiro Vaz, and Tiago Taveira Gomes. "MedGAN: optimized generative adversarial network with graph convolutional networks for novel molecule design". In: *Scientific Reports* 14.1 (Jan. 2024). ISSN: 2045-2322. DOI: 10.1038/s41598-023-50834-6.

[122] Andrew E. Blanchard, Christopher Stanley, and Debsindhu Bhowmik. "Using GANs with adaptive training data to search for new molecules". In: *Journal of Cheminformatics* 13.1 (Feb. 2021). ISSN: 1758-2946. DOI: 10.1186/s13321-021-00494-3.

[123] Ziqiao Zhang et al. "GANs for Molecule Generation in Drug Design and Discovery". In: *Generative Adversarial Learning: Architectures and Applications*. Springer International Publishing, 2022, pp. 233–273. ISBN: 9783030913908. DOI: 10.1007/978-3-030-91390-8_11.

[124] Arne Schneuing et al. "Structure-based drug design with equivariant diffusion models". In: *Nature Computational Science* 4.12 (Dec. 2024), pp. 899–909. ISSN: 2662-8457. DOI: 10.1038/s43588-024-00737-x.

[125] Clemens Isert, Kenneth Atz, and Gisbert Schneider. "Structure-based drug design with geometric deep learning". In: *Current Opinion in Structural Biology* 79 (2023), p. 102548. ISSN: 0959-440X. DOI: 10.1016/j.sbi.2023.102548.

[126] Emiel Hoogeboom et al. "Equivariant Diffusion for Molecule Generation in 3D". In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri et al. Vol. 162. Proceedings of Machine Learning Research. PMLR, July 2022, pp. 8867–8887.

[127] Minkai Xu et al. *Geometric Latent Diffusion Models for 3D Molecule Generation*. 2023. arXiv: 2305.01140 [cs.LG]. URL: https://arxiv.org/abs/2305.01140.

[128] Clement Vignac et al. *MiDi: Mixed Graph and 3D Denoising Diffusion for Molecule Generation*. 2023. arXiv: 2302.09048 [cs.LG]. URL: https://arxiv.org/abs/2302.09048.

[129] Pedro O. Pinheiro et al. "3D molecule generation by denoising voxel grids". In: *Proceedings of the 37th International Conference on Neural Information Processing Systems*. NeurIPS '23. New Orleans, LA, USA: Curran Associates Inc., 2023.

[130] Matthieu Kirchmeyer, Pedro O. Pinheiro, and Saeed Saremi. *Score-based 3D molecule generation with neural fields*. 2025. arXiv: 2501.08508 [cs.LG]. URL: https://arxiv.org/abs/2501.08508.

[131] Yuyang Wang et al. *Swallowing the Bitter Pill: Simplified Scalable Conformer Generation*. 2024. arXiv: 2311.17932 [physics.chem-ph]. URL: https://arxiv.org/abs/2311.17932.

[132] Chao Hu et al. "ScaffoldGVAE: scaffold generation and hopping of drug molecules via a variational autoencoder based on multi-view graph neural networks". In: *Journal of Cheminformatics* 15.1 (Oct. 2023). ISSN: 1758-2946. DOI: 10.1186/s13321-023-00766-0.

[133] Zhirui Liao et al. "Sc2Mol: a scaffold-based two-step molecule generator with variational autoencoder and transformer". In: *Bioinformatics* 39.1 (Dec. 2022), btac814. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btac814.

[134] Walid Ahmad et al. *ChemBERTa-2: Towards Chemical Foundation Models*. 2022. arXiv: 2209.01712 [cs.LG]. URL: https://arxiv.org/abs/2209.01712.

[135] Gengmo Zhou et al. "Uni-Mol: A Universal 3D Molecular Representation Learning Framework". In: *The 11th International Conference on Learning Representations*. 2023.

[136] Rohan Chabra et al. *Deep Local Shapes: Learning Local SDF Priors for Detailed 3D Reconstruction*. 2020. arXiv: 2003.10983 [cs.CV]. URL: https://arxiv.org/abs/2003.10983.

[137] AmirEhsan Khorashadizadeh et al. *LoFi: Neural Local Fields for Scalable Image Reconstruction*. 2024. arXiv: 2411.04995 [cs.CV]. URL: https://arxiv.org/abs/2411.04995.

[138] Chuhan Chen et al. “Implicit Neural Head Synthesis via Controllable Local Deformation Fields”. In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 416–426. DOI: 10.1109/CVPR52729.2023.00048.

[139] David Ha, Andrew Dai, and Quoc V. Le. *HyperNetworks*. 2016. arXiv: 1609.09106 [cs.LG]. URL: https://arxiv.org/abs/1609.09106.

[140] Vinod Kumar Chauhan et al. “A brief review of Hypernetworks in Deep Learning”. In: *Artificial Intelligence Review* 57.9 (Aug. 2024). ISSN: 1573-7462. DOI: 10.1007/s10462-024-10862-8.

[141] Jonathan Lorraine et al. “ATT3D: Amortized Text-to-3D Object Synthesis”. In: *The International Conference on Computer Vision (ICCV)* (2023).

[142] Yinbo Chen and Xiaolong Wang. “Transformers as meta-learners for implicit neural representations”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 170–187.

[143] Prafulla Dhariwal and Alex Nichol. “Diffusion models beat GANs on image synthesis”. In: *Proceedings of the 35th International Conference on Neural Information Processing Systems*. NeurIPS ’21. Red Hook, NY, USA: Curran Associates Inc., 2021. ISBN: 9781713845393.

[144] Grace Luo et al. “Diffusion hyperfeatures: searching through time and space for semantic correspondence”. In: *Proceedings of the 37th International Conference on Neural Information Processing Systems*. NeurIPS ’23. New Orleans, LA, USA: Curran Associates Inc., 2023.

[145] Alexander Quinn Nichol and Prafulla Dhariwal. “Improved Denoising Diffusion Probabilistic Models”. In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meilă and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, July 2021, pp. 8162–8171.

[146] Mariana Pegrucci Barcelos et al. "Lead Optimization in Drug Discovery". In: *Research Topics in Bioactivity, Environment and Energy: Experimental and Theoretical Tools*. Cham: Springer International Publishing, 2022, pp. 481–500. ISBN: 978-3-031-07622-0. DOI: 10.1007/978-3-031-07622-0_19.

[147] William L. Jorgensen. "Efficient Drug Lead Discovery and Optimization". In: *Accounts of Chemical Research* 42.6 (2009). PMID: 19317443, pp. 724–733. DOI: 10.1021/ar800236t.

[148] Stephanie Kay Ashenden. "Chapter 6 - Lead optimization". In: *The Era of Artificial Intelligence, Machine Learning, and Data Science in the Pharmaceutical Industry*. Ed. by Stephanie Kay Ashenden. Academic Press, 2021, pp. 103–117. ISBN: 978-0-12-820045-2. DOI: 10.1016/B978-0-12-820045-2.00007-6.

[149] Terry Kenakin. "Predicting therapeutic value in the lead optimization phase of drug discovery". In: *Nature Reviews Drug Discovery* 2.6 (June 2003), pp. 429–438. ISSN: 1474-1784. DOI: 10.1038/nrd1110.

[150] Noel M O'Boyle et al. "Open Babel: An Open Chemical Toolbox". In: *Journal of Cheminformatics* 3.1 (Oct. 2011). ISSN: 1758-2946. DOI: 10.1186/1758-2946-3-33.

[151] Zhenqin Wu et al. *MoleculeNet: A Benchmark for Molecular Machine Learning*. 2018. arXiv: 1703.00564 [cs.LG]. URL: https://arxiv.org/abs/1703.00564.

[152] Colin A. Grambow et al. "CREMP: Conformer-rotamer ensembles of macrocyclic peptides for machine learning". In: *Scientific Data* 11.1 (Aug. 2024). ISSN: 2052-4463. DOI: 10.1038/s41597-024-03698-y.

[153] Janani Durairaj et al. "PLINDER: The protein-ligand interactions dataset and evaluation resource". In: *bioRxiv* (2024). DOI: 10.1101/2024.07.17.603955. eprint: https://www.biorxiv.org/content/early/2024/07/19/2024.07.17.603955.1.full.pdf.

[154] Loring W. Tu. *An Introduction to Manifolds*. Second Edition. Springer New York, 2011. ISBN: 9781441974006. DOI: 10.1007/978-1-4419-7400-6.

[155] Afonso S. Bandeira, Philippe Rigollet, and Jonathan Weed. *Optimal rates of estimation for multi-reference alignment*. 2018. arXiv: 1702.08546 [math.ST]. URL: https://arxiv.org/abs/1702.08546.

[156] A. B. Tsybakov. *Introduction to Nonparametric Estimation*. Springer, 2009. DOI: 10.1007/b13794.

[157] Yang You et al. *Large Batch Optimization for Deep Learning: Training BERT in 76 minutes*. 2020. arXiv: 1904.00962 [cs.LG]. URL: https://arxiv.org/abs/1904.00962.