Piecewise Linear Homeomorphisms for Approximation of Invertible Maps

by

Richard E. Groff

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical Engineering: Systems)
in The University of Michigan
2003

Doctoral Committee:
Professor Daniel E. Koditschek, Co-Chair
Professor Pramod P. Khargonekar, Co-Chair
Professor Debasish Dutta
Professor David Neuhoff
Professor Quentin Stout

ABSTRACT

Piecewise Linear Homeomorphisms for Approximation of Invertible Maps

by
Richard E. Groff

Co-Chairs: Daniel E. Koditschek and Pramod P. Khargonekar

Changes of coordinates play an important role in design and analysis for a wide variety of fields, including control systems, robotics, and color systems management. Though not necessarily explicitly called a change of coordinates, many other applications require simultaneous approximation of forward and inverse models from data. This thesis proposes piecewise linear homeomorphisms (PLH) as a computationally effective, finitely parameterized family of nonlinear changes of coordinates. Other approximation techniques generally require that separate approximations be computed for the forward and inverse maps, whereas piecewise linear homeomorphisms are invertible in closed form, requiring only a single model. Motivated by the industrially significant problem of color systems management identified in collaboration with Xerox Corp., this dissertation presents work on the design and analysis of algorithms to compute piecewise linear homeomorphism approximations from data.

This thesis introduces two algorithms: the MINVAR algorithm for computing continuous multidimensional piecewise linear approximations to data, and the Graph Intersection algorithm, the scalar specialization of MINVAR. A geometrically influenced parameterization of PL functions as well as a theoretical framework for proving their properties is developed. The theoretical framework facilitates the main theoretical result, a proof of local convergence for MINVAR under the condition that the data generating function is piecewise linear with the same combinatorial structure as the approximation. Numerical studies of MINVAR and the Graph Intersection algorithm show that PL approximation compares favorably against other approximation techniques in terms of error and computational cost.

The color systems management problem in electrophotography is shown to reduce to a search for the change of coordinates embodied by the print engine. PLH approximations are proposed as a replacement for the current industry standard, lookup tables. A preliminary numerical study on a set of simulated color data provided by Xerox Corp. indicates that MINVAR-generated PLH approximations compare favorably to lookup tables, providing good approximations with a more parsimonious parameterization. Further study is required, but the author remains cautiously optimistic that these methods may have an eventual impact on the printing industry.

In memory of my father

ACKNOWLEDGEMENTS

This dissertation, the culmination of my years in graduate education, would not exist but for the support, encouragement, and contributions of many individuals.

I thank my co-advisors, Dan Koditschek and Pramod Khargonekar, for their continued guidance through this process. I have benefited more from interacting with Dan and Pramod jointly than I would have from interacting with them them both separately. Our discussions have shaped my opinions on engineering and academics immeasurably. Dan's infectious enthusiasm and vision make the lab an exciting place in which to work. Pramod's scholarship, impressively broad as well as deep, has provided insight from unsuspected areas on numerous occasions. I aspire to these qualities in my own work.

I thank my committee for their input and advice. I thank our colleagues at Xerox for their collaboration, letting us in on the trade secrets of the printing industry. Special thanks go to Tracy Thieret, my closest contact at Xerox and an all-around good guy.

Working in Dan's group is an intellectually rich experience, and I would especially like to thank Bill Schwind for being a great mentor, Noah Cowan for many discussions on research and our roles as engineers and academics, and Greg Sharp for his expert programming knowledge and for always being up for a chat on some esoteric subject, usually mathematics. The lab is always abuzz with new ideas, and I thank Eric Klavins, Uluc Saranlı, Eric Westervelt, Richard Altendorfer, Hal Komsuoglu, Joel Weingarten, Gabriel Lopes, Pei-Chun Lin, Erich Staudacher, and Rahul Bagdia for many stimulating conversations. Special thanks also go to Vaughn Washington, who helped to code an early version of MINVAR.

I thank the staff of the ATL building and the EECS department for making everything run smoothly, especially Karen Alexa, Karen Coulter, Jeff Kopmanis, Wendy Harrell, and Linda Cox.

I thank the many friends and colleagues who have helped make my time outside of lab a pleasure, Pat Simen, Web Stayman, Ketan Patel, Tara Javidi, Chris Lott, Corey Chen, Neal Yakelis, and the M-Foosball club.

My family has offered unfailing love and encouragement, even while not sure what I am doing or why it takes so long. I thank my parents for sparking my intellectual curiosity and encouraging me to pursue those interests. I wish my father were still here to share this accomplishment.

Most importantly, I owe my deepest gratitude to my wife Jen, who has offered her

unwavering patience and support, enduring all the deadlines and late nights, and all the social conversations that inevitably turn to robotics or Riemannian metrics or whatnot. Throughout this process Jen always believed in me, sometimes more than I did. I am blessed to have such a wonderful partner. She is the love of my life.

TABLE OF CONTENTS

LIST OF TABLES

Table

LIST OF FIGURES

Figure

LIST OF APPENDICES

APPENDIX

NOTATION

Geometry and Triangulations

d	dimension of domain, dimension of embedding space, $\mathbb{R}^d$
$\delta(\mathbf{x}, A)$	distance between a point $\mathbf{x}$ and a set A
$\delta_s(\mathbf{x}, H_i)$	signed distance between point $\mathbf{x}$ and a hyperplane H_i (page 18)
s	a simplex of any dimension in $\mathbb{R}^d$
$\bar{s}$	a d-simplex in $\mathbb{R}^d$
$\mathcal{D}(\bar{s}, \epsilon)$	ϵ dilation of $\bar{s}$ (page 21)
$\mathcal{K}$	geomeotric simplicial complex
$\mathcal{S}$	an abstract simplicial complex
$\mathcal{T}$	a triangulation
$\mathcal{T}(P, \mathcal{S})$	a parameterized candidate triangulation (see page 28), where P is a set of n vertices, $\mathbf{p}_1, \ldots, \mathbf{p}_n \in \mathbb{R}^d$ $\mathcal{S}$ is an abstract simplicial complex with vertex set $\{1, \ldots, n\}$
$\bar{s}_i$	i^{th} d-simplex of $\mathcal{T}(P, \mathcal{S})$ (d-simplices of $\mathcal{T}(P, \mathcal{S})$ are indexed $1, \ldots, N$)
$d_{i,j}$	dimension of facet shared by $\bar{s}_i$ and $\bar{s}_j$
N_i	number of d-simplices in the star of $\mathbf{p}_i$
r_1	maximum inter-vertex distance (page 29)
r_2	minimum orthogonal distance (page 29)
r_3	dilation radius (page 29)

PL Functions

$f_{\mathcal{P}}$	a PL function (page 43) parameterized by $\mathcal{P} = (P, Q, \mathcal{S})$, where P is a set of n vertices, $\mathbf{p}_1, \ldots, \mathbf{p}_n \in \mathbb{R}^d$ Q is a set of n vertices in the codomain $\mathbf{q}_1, \ldots, \mathbf{q}_n \in \mathbb{R}^d$ $\mathcal{S}$ is an abstract simplicial complex with vertex set $\{1, \ldots, n\}$
$\bar{s}_i$	i^{th} d-simplex of $\mathcal{T}(P, \mathcal{S})$ (d-simplices of $\mathcal{T}(P, \mathcal{S})$ are indexed $1, \ldots, N$)
$\mathbf{A}_i, \mathbf{b}_i$	$f_{\mathcal{P}}\big\vert_{\bar{s}_i}(\mathbf{x}) = \mathbf{A}_i\mathbf{x} + \mathbf{b}_i$

MINVAR

$\mathcal{Z}$	set of N_s data pairs
$\mathcal{Z}_j$	$\mathcal{Z}_j \subset \mathcal{Z}$, the data lying in $\bar{s}_i$
L_j	least squares approximation to $\mathcal{Z}_j$ $L_j(\mathbf{x}) = \hat{\mathbf{A}}_j\mathbf{x} + \hat{\mathbf{b}}_j$
L^i	the set of L_j's corresponding d-simplices in $\operatorname{St}\mathbf{p}_i$
$\mathbf{H}_i, \mathbf{h}_i$	used to solve the "min var" equation (page 59)
$\mathbf{H}_{c,i}, \mathbf{h}_{c,i}$	used to solve the constrained "min var" equation (page 61)

Local Convergence Proof

$f^*_{\mathcal{P}}, \mathbf{p}^*_i$, etc.	a * indicates quantities related to the data generating function
$\Pi(f)$	best L_2 affine approximation of f
$\varphi_i(\mathbf{x})$	construction for local convergence proof (page 72)
$\psi_i(\mathbf{x})$	construction for local convergence proof (page 72)

CHAPTER 1

Introduction

Changes of coordinates play an important role in design and analysis for a wide variety of fields, including control systems, mechanics, color systems management, pattern recognition, and more. Finding an appropriate change of coordinates may make design objectives easier to represent and achieve (e.g. canonical forms in control), elucidate underlying structure in the problem (e.g. action-angle coordinates in mechanics), or may itself be the solution to the problem (e.g. color stabilization, pattern recognition).

This thesis concerns a flexible representation of nonlinear changes of coordinates and a methodology for approximating them from data. In this thesis, a change of coordinates is a homeomorphism, i.e. a continuous function with a continuous inverse. Specifically, *piecewise linear homeomorphisms* (PLH) are proposed as a computationally effective, finitely parameterized family of nonlinear changes of coordinates.

Piecewise linear homeomorphisms are a subset of the space of *continuous piecewise linear* (PL) functions. For the most part, the theoretical background and algorithms presented in this thesis apply to the larger class of PL functions. PL functions possess the notable property that their invertibility can be checked geometrically, and they can be inverted in closed form. A piecewise linear homeomorphism is an invertible PL function.

1.1 Motivating Examples

To illustrate why changes of coordinates are important and why a closed form invertible representation is useful, consider some motivating examples. First, an example taken from linear control theory shows how a linear change of coordinates is employed in the pole placement problem. Second, the industrially relevant color stabilization problem is formulated as the search for the inverse of the coordinate transformation embodied by the print engine. This application initiated our interest in representing nonlinear changes of coordinates. The section concludes with a variety of other applications that could benefit from an effective representation of nonlinear changes of coordinates.

1.1.1 Control Systems

Canonical forms are a well established concept in control theory. A space of dynamical systems, for example the space of controllable linear systems, can be partitioned into equivalence classes by similarity, the equivalence relation defined by a linear change of coordinates. Each class is represented by its canonical form, a unique system embodying a certain structure, for example the controllable canonical form. Often, synthesis and analysis tasks can be simplified by transforming a system into an appropriate canonical form. The following example from linear state space control illustrates this.

Consider the dynamical system

$$\boldsymbol{\Sigma}_1 : \quad \dot{\mathbf{x}} \;=\; \mathbf{A}\mathbf{x} + \mathbf{b}u \tag{1.1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}$ is the input, and $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{b} \in \mathbb{R}^{n\times 1}$. Many of the properties of $\boldsymbol{\Sigma}_1$, such as stability and transient response, are characterized by the eigenvalues of $\mathbf{A}$. Suppose $\boldsymbol{\Sigma}_1$ is known to be controllable, and one wishes to design a feedback control $u = \mathbf{k}\mathbf{x}$ to perform pole placement, i.e. to select the eigenvalues of the closed loop system or equivalently, to select the characteristic polynomial of $\mathbf{A}+\mathbf{b}\mathbf{k}$, given by $\det(s\mathbf{I} - \mathbf{A} - \mathbf{b}\mathbf{k})$. Suppose the desired characteristic polynomial for the closed loop system is $s^n + \hat{\alpha}_{n-1}s^{n-1} + \cdots + \hat{\alpha}_1 s + \hat{\alpha}_0$. The relationship between $\mathbf{A}$, $\mathbf{b}$, $\mathbf{k}$ and the characteristic polynomial of the closed loop system is complicated, so it may appear difficult to find $\mathbf{k}$ to achieve this desired characteristic polynomial with the system in its present form.

Fortunately, a well known theorem from linear control [Che84, Theorem 7-1] states that a linear change of coordinates $\bar{\mathbf{x}} = \mathbf{P}\mathbf{x}$ exists (and may be computed algorithmically) that transforms $\boldsymbol{\Sigma}_1$ into controllable canonical form:

$$\boldsymbol{\Sigma}_2 : \quad \dot{\bar{\mathbf{x}}} \;=\; \bar{\mathbf{A}}\bar{\mathbf{x}} + \bar{\mathbf{b}}u \tag{1.2}$$

where $\bar{\mathbf{A}} = \mathbf{P}\mathbf{A}\mathbf{P}^{-1}$ and $\bar{\mathbf{b}} = \mathbf{P}\mathbf{b}$ take the form

$$\bar{\mathbf{A}} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ -\alpha_0 & -\alpha_1 & -\alpha_2 & \cdots & -\alpha_{n-2} & -\alpha_{n-1} \end{bmatrix}, \qquad \bar{\mathbf{b}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}. \tag{1.3}$$

Thanks to the structure of $\bar{\mathbf{A}}$, its characteristic polynomial, $\det(s\mathbf{I} - \bar{\mathbf{A}})$, is given by $s^n + \alpha_{n-1}s^{n-1} + \cdots + \alpha_1 s + \alpha_0$. Moreover, since the eigenvalues of a linear system are invariant under a linear change of coordinates, this is also the characteristic polynomial of $\mathbf{A}$. Designing a feedback to achieve a specified closed loop characteristic polynomial for a system in controllable canonical form is very easy due to the special structure of $\bar{\mathbf{b}}$.

Specifically, if the feedback $u = \bar{\mathbf{k}}\bar{\mathbf{x}}$, with

$$\bar{\mathbf{k}} = \begin{bmatrix} \alpha_0 - \hat{\alpha}_0 & \alpha_1 - \hat{\alpha}_1 & \cdots & \alpha_{n-1} - \hat{\alpha}_{n-1} \end{bmatrix},$$

is applied to $\mathbf{\Sigma_2}$, then the closed loop system will have the desired characteristic polynomial, $s^n + \hat{\alpha}_{n-1}s^{n-1} + \cdots + \hat{\alpha}_1 s + \hat{\alpha}_0$. Since eigenvalues are invariant under a change of coordinates, applying the feedback $u = \mathbf{kx}$, where $\mathbf{k} = \bar{\mathbf{k}}\mathbf{P}^{-1}$, to $\mathbf{\Sigma}_1$ will give the desired closed loop characteristic polynomial for the original system!

Designing the feedback gain $\mathbf{k}$ for $\mathbf{\Sigma}_1$ directly would have been difficult. Since the design objective is invariant under coordinate transformations, a change of coordinates is applied to transform the system to control canonical form for which the design problem is trivial, and the resulting design is translated back to a design for the original system through the inverse change of coordinates. Solving the design problem for the canonical form solves the problem for all systems related to the canonical form through a (in this case linear) change of coordinates.

Several canonical forms have been studied for classes of nonlinear systems, for example Brunowsky's canonical form [Isi95]. It is generally much more difficult to transform a nonlinear system to a canonical form without significant insight into the algebraic structure of the system. Moreover, it can be very difficult to show that a given nonlinear transformation actually is a valid change of coordinates.

Linear systems theory is very powerful, aided in part by the tractability of computing linear changes of coordinates. Some problems, however, are simply not linear, such as our next example.

1.1.2 Color Systems Management

Working in collaboration with Xerox Corp. under an NSF GOALI grant, the color systems management problem initiated our interest in representing changes of coordinates. A more detailed background to this problem is provided in Chapter 7.

If one repeatedly prints a color image on a laser printer over a period of weeks and examines the resulting prints, one will notice that though each may look fine individually, there is a variation in color tone from print to print. This variation is undesirable and, for some applications, may be unacceptable, such as printing trademarked colors or high quality photography. Offset lithography, the traditional method of color printing, reproduces color more consistently but has higher fixed costs per document run [Bla83]. To gain further competitive advantage against offset lithography so that electrophotography is the clear choice for small color print jobs (less than 500 copies), the consistency of electrophotographic color reproduction must be improved. To this end, Xerox has implemented low signal level controls inside laser printers and copiers for twenty years, but significant color variation remains. The next step is to wrap a higher level feedback around the color commands to the print engine.

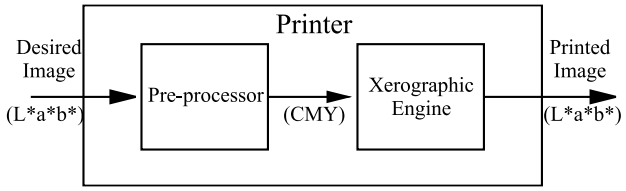

Figure 1.1: Decomposition of a printer into the print engine and preprocessor. The engine embodies the transformation T, while the preprocessor performs an approximation of the inverse $\tilde{T}^{-1}$, such that $T \circ \tilde{T}^{-1} \approx i_d$.

A laser printer can be conceptually decomposed into two components (See Figure 1.1), the print engine that physically marks the page, and the preprocessor that interprets the document description sent from the computer and gives commands to the print engine. The print engine of a color laser printer accepts commands as points in a *device dependent color space*. This color coordinate space is based on how the device physically produces a desired color. In a printer, the device dependent coordinate space is CMY, corresponding to cyan, magenta, and yellow, specifying how much of each respective toner is used to produce the commanded color. Sometimes CMYK is used, where K corresponds to black, in order to extend the color gamut, the range of reproducible colors.

Specifying the colors in a document should not require the user to know what type of toner is in the printer. Thus, when a computer sends a color document to a laser printer, the color of each pixel is specified as a point in a *device independent color space*, such as L*a*b*, a space based on the psychophysics of the eye. The L*a*b* coordinates of a printed color patch can be measured using a colorimetric sensor.

The print engine of the printer accepts color commands in CMY coordinates and print colors measured in L*a*b* coordinates. Thus, the print engine embodies a color space transformation T : CMY $\rightarrow$ L*a*b*. The transformation T is called the forward device characterization [Bal03], and, lacking a sufficient first principles model, is typically only known through color patch experiments. In order for the printer to reproduce a color specified in L*a*b* coordinates, the preprocessor should ideally have access to the inverse of the forward device characterization T^{-1} : L*a*b* $\rightarrow$ CMY, called the inverse device characterization. Using T^{-1} the preprocessor could transform a requested L*a*b* color $\mathbf{x}$ into CMY command, $T^{-1}(\mathbf{x})$, for the print engine. Then the print engine would print the color $T \circ T^{-1}(\mathbf{x}) = \mathbf{x}$, i.e. the printed color would be identical to the requested color. Since T^{-1} is not known, the preprocessor uses an approximation $\tilde{T}^{-1}$, and one hopes that $T \circ \tilde{T}^{-1} \approx i_d$, where i_d is the identity map. The approximation $\tilde{T}^{-1}$ is constructed from a few thousand color patch experiments as part of the calibration process. For low end printers, a single calibration may be performed for an entire product line at the factory, while high end printers are individually calibrated at the factory and can be recalibrated in the field by a technician. Further complicating matters, T gradually changes with time,

affected by environmental factors such as temperature, humidity, toner concentration, and photoreceptor voltages. If the approximation $\tilde{T}^{-1}$ remains fixed as T drifts, this will be observed as tone variation in the resulting prints.

Industry practice has settled on the least parsimonious of representations for $\tilde{T}^{-1}$, the look-up table (LUT) [Kan97]. The difficulty of building a lookup table for $\tilde{T}^{-1}$ is additionally complicated, since the data is sampled from T, and thus achieving uniform spacing requires either interpolation or iterative experiments. Instead, this thesis proposes to use a single PLH to approximate both the forward and inverse device characteristics simultaneously. The approximation to the forward characteristic is useful for performing feedback on the print engine, while the approximation to the inverse device characteristic is required by the preprocessor to generate print commands. The PLH is much more parsimonious than a look-up table, making it amenable to an eventual real-time calibration process, where a color sensor in the machine would be used to periodically or continually recalibrate $\tilde{T}$.

1.1.3 Other Examples

A flexible representation for changes of coordinates would be useful in many other application areas. This section presents a brief overview of a number of these areas.

Robot Navigation

Artificial potential functions have long been applied for robot navigation, see for example [Kha80, KM78, Kro84]. In these methods, the robot moves in the direction of the negative gradient of the potential, leading it to a minimum of the potential function, the encoding of the goal state. Most artificial potential methods suffer from undesired local minima in addition to the goal state. In [KR90], Koditschek and Rimon introduce navigation functions, a special class of potential functions for which the only local minimum is the global minimum. They show how to construct a navigation function for any configuration space that is a "sphere world," a subset of $\mathbb{R}^d$ whose boundary is formed from a disjoint union of a finite number of spheres. They also show that the navigation properties are invariant under a diffeomorphism[1] of the configuration space, so a navigation function can be constructed for any manifold that can be deformed to a sphere world. To extend the applicability of navigation functions, [RK91] provides a method for constructing diffeomorphisms from "star-shaped worlds" to sphere worlds. Still, many environments are neither sphere worlds nor star-shaped worlds.

Piecewise linear homeomorphisms provide a viable method for generalizing navigation functions to complicated environments. One concern might be that PLHs lack smoothness beyond continuity, but nevertheless they yield a well defined flow in the target environment, even though the vector field is not well defined everywhere.

[1] A diffeomorphism is a homeomorphism that is smooth and has a smooth inverse.

Pattern Recognition

There is a long tradition in the AI literature of what might be termed "invariant recall," attributed originally to Minsky and Papert [MP69]. In this paradigm, the problem of pattern recognition is cast as the computational effort to identify an archetypal pattern that presents itself as some "deformed" set of features. Namely, one assumes that the sensed data arises from the action upon the archetype of some unknown member of a known group of transformations. The pattern recognition task is to "factor out" the particularizing transform so as to reveal that "true" exemplar of the "concept."

There is an extensive literature following this approach to pattern recognition. One modern example is [TY00], which considers scalar diffeomorphisms of the interval for function matching problems. For example, a pair of parameterized curves through space, $f_i : I \to \mathbb{R}^d$, $i = 1, 2$, could be matched by finding a diffeomorphism from I to I that causes $\|f_1 - f_2\|$ to be small. [TY00] provides sufficient conditions for the existence of an optimal match, but does not provide a computational representation for matching problems, providing another potential application of piecewise linear homeomorphisms.

More applications of invertible functions

The field of robotics is rife with examples where changes of coordinates play a key role: in the representation of gaits [Rai86, SK00, WBGK03]; in sensor based manipulation [CWK02]; as well as in calibration [ZD89]. Though not necessarily explicitly called a change of coordinates, many applications require simultaneous approximation of forward and inverse models from data. Representations of scalar invertible functions are required for certain machine tool calibration problems [HNY98], for certain automobile fuel control settings [FK98], as well as for probability density estimation [Fio01]. Camera calibration with correction for lens distortion requires scalar or possibly planar homeomorphisms [Tsa87, Wen92, Hei00].

In such settings, most approximation techniques require the construction of distinct forward and inverse representations, because the approximations are not invertible in closed form. In addition to doubling the effective training effort, accuracy suffers since the approximation of the inverse is not exactly the inverse of the forward approximation. In contrast, invertibility of PL functions can be verified and even imposed geometrically, that is, by well characterized and computationally effective techniques arising from geometric insights. Moreover, if a PL function is invertible, it can be inverted in closed form. Thus, PL approximation is well suited for these sorts of applications.

1.2 Piecewise Linear Approximation and Invertible Approximations in the Literature

A substantial mathematical literature on real function approximation (see, for example, [Che66, Dav75, Lor66]), largely concerned with linear-in-parameters techniques, deals extensively with algorithms, fundamental limits, convergence rates, and families of bases in approximating functions. Recent activity has been spurred by evidence that nonlinear-in-parameters function families, for example neural networks and radial basis function networks, offer improved approximation rates in higher dimensions as compared to linear-in-parameters representations [Bar93]. Continuous piecewise linear functions may be nonlinear- or linear-in-parameters, depending on whether or not the domain triangulation, the partition of the domain, is allowed to change as part of the parameterization.

Piecewise linear functions have been addressed in a number of different settings. Dating back to the fifties, algebraic topologists have used piecewise linear homeomorphisms to classify topological spaces [Spa66, Pon52], allowing many topological properties of a space to be captured by the purely combinatorial structure of an abstract simplicial complex. Piecewise linear functions were intimately involved in their work, but computational considerations were not addressed. Today, topologists are increasingly interested in computation, with a new subdiscipline called computational topology emerging [DEG99], but the focus seems to have not yet come to rest on piecewise linear homeomorphisms. Triangulations, vital for the representation of piecewise linear homeomorphisms, have been studied in the related field known as computational geometry [Ede01, Bro79, Law86, PS85, ES96], though results for general dimension are still limited.

Interest and research in approximation using piecewise linear functions is derived primarily from three areas: piecewise polynomial approximations, spline approximations (under which finite element methods are included), and locally-parametric/nonparametric approximations. This section makes these distinctions more precise, discusses influential work from each of these areas, and concludes with an appraisal of progress in approximation of invertible functions.

1.2.1 Piecewise Polynomials

The piecewise polynomial literature addresses the problem of finding a partition of the domain and a polynomial for each cell of the partion in order to approximate an explicitly known function, almost all confined to the scalar case (with the notable exception of [TB97]). The domain partition is considered as part of the approximant's parameterization. Piecewise polynomial approximants have no continuity requirements between partition cells of the domain, i.e. generically, an approximation will be discontinuous. In applications, one generally requires that an approximation be at least continuous, but various authors note that the best piecewise polynomial approximation is often continuous or "nearly contin-

uous" [BCSW78, Nad86, TB97]. That is, under certain conditions the best partition of the domain will cause the best (possibly discontinuous) approximation, e.g. best L_2, to be continuous. Most of the results in this field are for approximation of scalar functions (from $\mathbb{R}$ to $\mathbb{R}$). A survey of these results follows.

Barrow et al. [BCSW78] present generalized convexity conditions which guarantee the existence of a unique best L_2 approximation from S_N^2, the space of second order B-splines (i.e. continuous piecewise linear) with N vertices ($N-1$ domain cells). Chow [Cho82] provides a generalization showing that equivalent conditions guarantee existence of a unique best L_2 approximation from P_N^k, the space of piecewise polynomial approximations with N vertices ($N-1$ domain cells) and k^{th} order (degree $k-1$) polynomials in each cell. S_N^2 is a submanifold of P_N^2. Note that this implies that under these generalized convexity conditions, the best piecewise discontinuous approximation will be continuous. Neither of these proofs is constructive.

In [Kio80, Kio81], Kioustelidis presents necessary conditions for an optimal piecewise polynomial, possibly discontinuous, approximation. If an optimal piecewise polynomial approximation to a continuous function exists, then the error modulus will be continuous. Motivated by a clever observation about the error modulus in neighboring cells [Phi70], Kioustelidis also presents an algorithm for computing a good, but not necessarily best, piecewise polynomial L_p approximation to a continuous function. This algorithm uses two complementary steps. The first step finds the locally optimal approximation in each cell, given a partition. The second step adjusts the partition such that the error modulus is decreased over the entire domain. The Kioustelidis algorithm is closely related in spirit to the Graph Intersection and MINVAR algorithms presented in this dissertation.

Gayle and Wolfe [GW96] apply a modified version of the Kioustelidis algorithm and re-establish the uniqueness results of Barrow et al. [BCSW78] and Chow [Cho82] under the same generalized convexity conditions. This is achieved by showing that under the generalized convexity conditions, the modified Kioustelidis algorithm becomes a contraction on the space of piecewise polynomials, and uniqueness is established through application of the contraction mapping theorem.

Baines [Bai94] presents another algorithm for computing possibly discontinuous piecewise linear and piecewise constant L_2 approximations to continuous functions. This one dimensional algorithm [Bai94] is similar to that proposed by Kioustelidis. Baines and Tourigny [TB97] propose a generalization to higher dimensions and show numerical results in two dimensions. They provide a convergence proof for the algorithm. Also, Tourigny et al. [TH98] provide a related algorithm for a moving mesh finite element (i.e. vertex spline) solution to variational problems using a similar technique. A specialization of this moving mesh algorithm is finding the best L^p, p finite and even, continuous piecewise polynomial approximation to a function.

The piecewise polynomial literature focuses on possibly discontinuous approximations

to a function that is completely known. The multidimensional algorithms of [TB97, TH98], as well as the scalar algorithms of [Bai94, Kio80, Kio81], entail steps such as root finding that intrinsically incorporate the function to be approximated. In contrast, this dissertation concerns continuous approximations to a function that is observed through a discrete set of data. With this distinction in mind, the MINVAR and Graph Intersection algorithms, presented in Chapters 4 and 6, produce continuous piecewise linear approximations and are designed to be used with discrete data or with the actual function. The MINVAR algorithm, moreover, is defined for general dimension.

1.2.2 Splines

Splines are piecewise polynomials that are guaranteed to be smooth up to some degree, that is, splines are C^r for $r \geq 1$ [dB01, SP95]. Continuous piecewise linear functions qualify as splines, but because the spline literature is especially interested in approximations that are differentiable or smoother, piecewise linear approximations seem to play a lesser role.

De Boor introduced *basis splines*, better known as B-splines, which provide a convenient way to compute scalar spline approximations [dB01]. Given a partition of the interval over which to form the approximation, the B-splines of order k (so the polynomials will be degree $k-1$) form a family of smooth basis functions, such that the support for each basis function is k consecutive partition cells. For a fixed partition, computing the least squares or best L_2 spline is a linear-in-parameters problem, and thus can be solved easily. Section 3.3 shows how to compute the least squares (multidimensional) piecewise linear approximation for a fixed triangulation, following the methodology of B-spline approximations.

Box splines are a typical method of extending splines to multiple dimensions by means of tensor products of univariate splines [dBHR93, dBD83, Chu88]. The domain partition is then a tensor product of partitions of the individual dimensions and the approximant is the sum of tensor products of scalar spline functions. A multidimensional linear box spline is multilinear, that is linear in each variable while holding the other variables constant, rather than truly linear. The box spline literature tends to deal with very uniform partitions of domain.

Another extension of splines to multiple dimensions are *vertex splines*, which use a simplicial partition or triangulation, as described in Section 2.2, to extend splines to multiple dimensions [CL87, Chu88]. Vertex splines are often used in finite element modeling [Bat96]. The piecewise linear functions presented in Chapter 3 may be considered as vertex splines, though this literature is often interested in higher order smoothness, and hence generally pays less attention to the piecewise linear case.

The multidimensional spline literature generally considers the domain partition to be fixed. In this case, best L_2 spline approximation is a linear-in-parameters problem and can be solved by standard linear least squares methods. The partitions of the domain are generally very regular and highly structured. If the L_2 approximation over a partition is

not good enough, the spline literature will typically refine the partition (this technique is covered further in the following subsection on nonparametric approximation). This stands in contrast to the piecewise polynomial literature, which provides several algorithms for moving a partition with a fixed number of cells. This dissertation covers a middle ground, continuous piecewise linear approximations for which the domain partition is moved in order to improve the approximation.

1.2.3 Locally Parametric and Nonparametric Approximations

Piecewise polynomial, spline, and hence piecewise linear, approximations fall into a larger class of approximation schemes, sometimes referred to as locally parametric, in which the approximant is a combination of basis functions with local support. Adjusting a parameter changes the approximant over some bounded region of the domain, i.e. locally rather than globally. Approximation methods over a partition of the domain, as well as schemes that use a cover rather than a partition [AMS97], fall naturally into this group. Locally parametric approximation leads to nonparametric methods, where the quality of the approximation is improved by adjusting the number of localized basis functions, as well as each basis function's local parameters.

In nonparametric regression, the function to be approximated is assumed to be drawn from a very large family of distributions that cannot be finitely parameterized in a natural way [Háj69]. The general approach in nonparametric regression is to allow the approximation algorithm to add and remove parameters adaptively in order to improve the fit to the observed data. With splines, parameters are added by refining the domain partition. An excellent example is Friedman's Multivariate Adaptive Regression Splines (MARS), which use a box spline representation and an algorithm to add and remove spline basis functions [Fri91, SHKT97].

Schaal et al. present a nonparametric technique called locally weighted progression regression in [AMS97, SA98], that uses a collection of local linear models, each with an associated Gaussian confidence region. The approximation is given as the average of the linear models weighted by the confidences. The approximation algorithm adds and removes local linear models in order to maintain a certain quality fit to the data.

The piecewise linear approximation algorithms presented in this dissertation are local approximations in which the domain partition is also adjusted in order to improve the approximation while maintaining a fixed number of parameters in the approximation. A natural extension to these algorithms would be a nonparametric approach, where parameters are added or removed by refining or coarsening the domain triangulation.

1.2.4 Approximating Invertible Functions

The topic of approximating invertible functions has risen several times in the literature, most often involving continuous scalar functions on a connected set, for which invertibility

is equivalent to strict monotonicity. In multiple dimensions there has been some interest in simultaneously fitting separate forward and inverse models to a set of data, and updating these approximations as additional data are received.

In [AM94], Abu-Mostafa proposed a method to add "hints" describing *a priori* known features of the data generating function by penalizing the squared error criterion with a term describing the "hint." One of the proposed "hints" for a function from $\mathbb{R}$ to $\mathbb{R}$ is monotonicity. The monotonicity hint involves evaluating the approximation at a number of pairs of points in the domain. The hint for a pair is zero if the points respect the desired monotonicity, and the squared difference if they violate monotonicity. The sum of the pairwise hints is added to the squared error over the data, and this quantity is then minimized via gradient descent to obtain the approximation. This method does not guarantee that the resulting scalar function will be monotone for two reasons. First, the descent algorithm is free to trade off improvements in approximation error for loss of monotonicity. Second, even if the summed hint error is zero for the approximation, the hint error is evaluated only at certain points.

In [FB01, Fio01], Fiori et al. propose approximating cumulative distribution functions with an "adaptive activation function neuron (FAN)." A FAN is the composition of a sigmodial function and a polynomial with a special structure. The sigmoid is smooth and monotonically increasing from 0 to 1. The polynomial has odd powers plus a constant term, with all coefficients nonnegative. Clearly, the polynomial will be monotonically nondecreasing, and thus a FAN is a monotonically nondecreasing function.

Both Abu-Mostafa and Fiori take advantage of the fact that strict monotonicity and invertibility are equivalent for (continuous) scalar functions. Abu-Mostafa uses this fact to modify the error function in order to bias a general scalar approximation toward an invertible approximation. Fiori restricts the approximant to a set that is guaranteed to be invertible. Unfortunately, invertibility is more difficult to characterize in dimensions higher than one. Proving that a function is invertible on its range can take a great deal of analysis [RK91], since one must check not only that the Jacobian (for a smooth function) is nonsingular, but also that injectivity is not lost due to the global structure of the function. In Chapter 3, it is shown that a continuous PL function can be checked for invertibility in a straightforward manner via a geometric interpretation of the parameterization.

In [JW99], Jordan considers learning a forward and inverse model of a dynamical system in the context of motor control tasks in robots and animals. The forward model is used to compute the feedback component of the control signal, while the inverse model is used to compute the feedforward component. Separate neural network approximations for the forward and the inverse models are computed. Since the forward model should already contain the information of the inverse model, it would be nice to have an approximant that could be easily inverted in order to avoid construction of a separate inverse model. In Chapter 3 it is shown that an invertible continuous piecewise linear function can be inverted

in closed form, thus permitting a single approximation to be constructed for problems that require a simultaneous forward and inverse model.

Given the range of systems problems that can be posed and solved as a search for an appropriate change of coordinates, the paucity of results on the topic of invertible function approximations seems surprising. In particular, the literature does not address the common need of many of the applications presented in this chapter for a computationally effective, invertible, multidimensional function approximation technique. This thesis addresses the apparent gap in the literature.

1.3 Contributions

This thesis introduces MINVAR, a novel algorithm for computing multidimensional PL approximations to data. MINVAR moves both the domain and codomain vertices of the PL function, making the approximation problem nonlinear-in-parameters. The algorithm consists of two stages. In the first stage, the data is sorted into subsets according to the d-simplices of the domain triangulation, and then the least squares affine approximation is computed for each data subset. In the second stage, the domain and codomain vertices are moved to make the continuous PL approximation closer to the (generally discontinous) least squares approximations. The movement of a domain vertex, directed by the "min var" equation, depends only on the current vertex location and on the least squares approximations corresponding to d-simplices surrounding the vertex. In addition to the basic algorithm, MINVAR is extended to allow affinely constrained vertex movement and heuristically guided retriangulation. The Graph Intersection algorithm, the scalar precurser of MINVAR is also presented.

This thesis develops a geometrically influenced representation of PL functions. The parameterization is composed of three elements: a set of domain vertices, a set of corresponding codomain vertices, and an abstract simplicial complex. If the PL function is invertible, then the abstract simplicial complex can be coupled with the domain and codomain vertices to give triangulations in the domain and codomain, respectively. Though similar parameterizations have been used in algebraic topology since the fifties, and the geometric influence is drawn heavily from computational geometry, e.g. [Ede01], to the author's best knowledge this parameterization has not been used in the approximation literature. This representation is especially important in light of this thesis's interest in piecewise linear homeomorphisms, since it allows invertibility of a PL function to be checked geometrically and permits a PL function to be inverted in closed form (that is, the parameters of the inverse are given directly by the parameters of the forward function).

The MINVAR algorithm moves both the domain and codomain vertices, making the approximation problem nonlinear-in-parameters. Thus, one should not expect global convergence. This thesis presents a local convergence proof for the MINVAR algorithm, under

the assumption that the data generating function is also piecewise linear. The structure of the proof follows the stages of the MINVAR algorithm. Assuming that the vertices of the approximation are ϵ close to the vertices of the data generating function, it is shown that the least squares approximations from the first stage of the MINVAR algorithm will be an ϵ^2 perturbation away from the true affine map over the data generating function's corresponding d-simplex. Then it is shown that the ϵ^2 perturbations in the least squares affine approximations cause the "min var" equation to move each approximation domain vertex to within an ϵ^2 perturbation of the data generating function's corresponding domain vertex. These results are combined to prove local convergence of the MINVAR algorithm. This proof subsumes an earlier proof of convergence for the Graph Intersection algorithm.

The MINVAR algorithm is implemented in C++. The greatest difficulties in implementation are the "bookkeeping" related to the combinatorial components of a PL function and dealing with degeneracies in the triangulations. Topological flipping behaves differently when too many points lie in the same affine subspace, but this degenerate condition can be difficult to check numerically. The MINVAR algorithm is being rewritten as a general purpose C++ library for evaluating and approximating PL functions.

The Graph Intersection (GI) algorithm is implemented in Matlab, and is available for download at `http://www.eecs.umich.edu/~regroff/research`. A numerical study comparing its performance on a variety of function families against three other approximation techniques, neural networks, least squares polynomial, and a gradient-descent based piecewise linear approximation, shows that GI is competetive in terms of approximation error and more efficient computationally.

The color systems management problem in electrophotography is shown to reduce to a search for the change of coordinates embodied by the print engine. PLH approximations are proposed as a replacement for the current industrial standard, a lookup table. A preliminary numerical study on a set of simulated color data provided by Xerox Corp. indicates that MINVAR-generated PLH approximations compare favorably to lookup tables, providing better approximations with a more parsimonious parameterization. Further study on actual hardware is required, but these methods may have an eventual impact on the printing industry.

1.4 Organization of Dissertation

Chapter 2 provides background material on convex and affine geometry, the formal definition of a triangulation, and related issues such as triangulating a set of points and validating the embedding of an abstract simplicial complex. These topics are required for the geometric interpretation of PL functions. Chapter 3 covers multidimensional PL functions, including their parameterization and basic properties. Chapter 3 also shows how to perform PL data interpolation, to construct the least squares PL approximation to a set

of data for a fixed domain triangulation, and to invert a PL function in closed form. Chapter 4 presents the MINVAR algorithm, a novel algorithm for computing multidimensional (from $\mathbb{R}^d$ to $\mathbb{R}^c$) PL approximations to data. Chapter 4 also presents a numerical study of MINVAR on a two dimensional test function. Chapter 5 presents a local convergence proof for the MINVAR algorithm. Chapter 6 presents the scalar precursor of the MINVAR algorithm, called the Graph Intersection algorithm, as well as a numerical study of this algorithm. Chapter 7 presents electrophotography and the color systems managment problem, and discusses a study of MINVAR's performance on a set of color data supplied by Xerox. Chapter 8 is the conclusion. Appendix A presents proofs and discussion related to Chapter 2. Appendix B presents algebraic details of a number of the calculations involved in the MINVAR algorithm, presented in Chapter 4.

CHAPTER 2

Geometry and Triangulations

This dissertation takes a strongly geometric view of *continuous piecewise linear* (PL) functions, and piecewise linear homeomorphisms (PLH) in specific. A PL function, as shown in Chapter 3, has both continuous and combinatorial parameters. The ability to check invertibility of a PL function and to invert it in closed form is elucidated through a geometric interpretation of those parameters. This chapter presents the requisite background in geometry and triangulations.

This chapter is divided into four sections. Section 2.1 provides background on affine and convex geometry. The dilation of a simplex and the interpretation of barycentric coordinates as distances both play important roles in proving properties of PL functions as well as the MINVAR local convergence proof. Section 2.2 provides a formal definition of triangulation, based upon the notion of a geometric simplicial complex. Perhaps the most important concept in this chapter is the parameterization of triangulations from Section 2.2.2, which is fundamental in the definition of multidimensional piecewise linear functions in Chapter 3. Section 2.3 presents the computational geometry problem of triangulating a set of points in $\mathbb{R}^d$. The Lawson algorithm for generating the 2-dimensional Delaunay triangulation through edge flipping (Section 2.3.3) motivated Dyn et al.'s algorithm for generating 2-dimensional data dependent triangulations for interpolation [DLR90b], described in Section 3.2.1. Coupling these ideas with local topological flipping in higher dimensions (Section 2.3.4) leads to the heuristically guided retriangulation in general dimension developed for the MINVAR algorithm (Section 4.3). The chapter concludes with Section 2.4 on validating an embedding of a triangulation, which can be used to check if a PL function is invertible (Section 3.4).

The material in this chapter is a synthesis of results from the fields of convexity, computational geometry and algebraic topology. Many of the presented results are standard in their fields. For further reading on convex and affine geometry (Section 2.1) see [Web94, Bar02] and for triangulations (Section 2.2) see [Ede01]. Some of the results are original, including the generalized interpretation of barycentric coordinates as distances, and the dilation and its properties (Section 2.1.4), which are used to prove properties of PL functions as well as

MINVAR local convergence.

To ease readability, proofs of the claims in this chapter are supplied in Appendix A rather than in the chapter itself.

2.1 Geometry

2.1.1 Affine Subspaces

An *affine subspace* $V \subseteq \mathbb{R}^d$ is a linear subspace $L \subseteq \mathbb{R}^d$ translated by some $\mathbf{x}_o \in \mathbb{R}^d$, i.e. $V = L + \mathbf{x}_o$.[1] The dimension of V is $\dim(V) = \dim(L)$. An affine subspace in $\mathbb{R}^d$ with dimension $d-1$ is called a *hyperplane*. A finite set of points $\mathcal{U} \subseteq \mathbb{R}^d$ is *affinely independent* if for $i = 1, \ldots, d$, no affine subspace of dimension i contains more that $i+1$ points from $\mathcal{U}$. The next two claims show the connection between affine independence and linear independence as well as provide a computational method to check affine independence.

Claim 2.1. *$\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_m \in \mathbb{R}^d$ are affinely independent if and only if the vectors $\mathbf{p}_1 - \mathbf{p}_m, \mathbf{p}_2 - \mathbf{p}_m, \ldots, \mathbf{p}_{m-1} - \mathbf{p}_m$ are linearly independent.*

It follows from Claim 2.1 that at most $d+1$ points in $\mathbb{R}^d$ can be affinely independent. The following claim offers another method of checking if a set of points is affinely independent.

Claim 2.2. *Let $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_m \in \mathbb{R}^d$. Then $\mathbf{p}_1, \ldots, \mathbf{p}_m$ are affinely independent if and only if*

$$\sum_{i=1}^{m} \alpha_i \mathbf{p}_i = 0 \quad \text{and} \quad \sum_{i=1}^{m} \alpha_i = 0 \qquad \text{where} \quad \alpha_i \in \mathbb{R} \tag{2.1}$$

holds only when $\alpha_i = 0$ for all i, or equivalently in matrix notation if

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \cdots & \mathbf{p}_m \\ 1 & 1 & \cdots & 1 \end{bmatrix} \tag{2.2}$$

has a trivial null space.

The matrix $\mathbf{P}$ contains the points in homogeneous form. This matrix appears quite often in computations, for example of barycentric coordinates and piecewise linear functions.

The *affine hull* of a set $\mathcal{U} \subseteq \mathbb{R}^d$, denoted $\mathrm{aff}(\mathcal{U})$, is the intersection of all affine subspaces containing $\mathcal{U}$,

$$\mathrm{aff}(\mathcal{U}) = \bigcap_{\substack{\mathcal{U} \subseteq V \\ V \text{ aff. sbsp.}}} V.$$

Since the intersection of an arbitrary number of affine subspaces in $\mathbb{R}^d$ is also an affine subspace ([Web94] Theorem 1.2.3), $\mathrm{aff}(\mathcal{U})$ is the smallest affine subspace containing $\mathcal{U}$. A

[1] $L + \mathbf{x}_o$ denotes the vector sum $L + \{\mathbf{x}_o\}$. For $A \subset \mathbb{R}^d$ and $B \subset \mathbb{R}^d$, the *vector sum* of A and B, written $A + B$, is a subset of $\mathbb{R}^d$, defined as $A + B = \{x + y \,|\, x \in A,\, y \in B\}$.

further result of Claim 2.1 is that the affine hull of $m+1$ affinely independent points is an m dimensional affine subspace.

Consider $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_m \in \mathbb{R}^d$. Then $\mathbf{p}$, given by

$$\mathbf{p} = \sum_{i=1}^{m} \alpha_i \mathbf{p}_i \qquad \text{where} \quad \alpha_i \in \mathbb{R}, \quad \sum_{i=1}^{m} \alpha_i = 1,$$

is called an *affine combination* of $\mathbf{p}_1, \ldots, \mathbf{p}_m$. The set of all affine combinations of $\mathbf{p}_1, \ldots, \mathbf{p}_m$ is called the *affine set generated by* $\mathbf{p}_1, \ldots, \mathbf{p}_m$. The following claim shows that the affine set generated is equivalent to the affine hull of the points.

Claim 2.3. *The affine set generated by $\mathbf{p}_1, \ldots, \mathbf{p}_m \in \mathbb{R}^d$ is the affine hull of the points,* $\text{aff}\{\mathbf{p}_1, \ldots, \mathbf{p}_m\}$.

2.1.2 Barycentric Coordinates

Let $\mathbf{p}_1, \ldots, \mathbf{p}_{k+1}$ be affinely independent points in $\mathbb{R}^d$. We say that $\mathbf{p}_1, \ldots, \mathbf{p}_{k+1}$ is an *affine basis* for $V = \text{aff}\{\mathbf{p}_1, \ldots, \mathbf{p}_{k+1}\}$. As a result of Claim 2.1, V is a k-dimensional affine subspace. By Claim 2.3, V is the affine set generated by $\mathbf{p}_1, \ldots, \mathbf{p}_{k+1}$, thus any point $\mathbf{x} \in V$ can be written as an affine combination of the $\mathbf{p}_i$'s, i.e. $\mathbf{x} = \sum_{i=1}^{k+1} \alpha_i \mathbf{p}_i$, where $\sum_{i=1}^{k+1} \alpha_i = 1$. The coefficients α_i are called the *barycentric coordinates* of $\mathbf{x}$ with respect to the affine basis $\mathbf{p}_1, \ldots, \mathbf{p}_{k+1}$.

Claim 2.4. *Let $\mathbf{p}_1, \ldots, \mathbf{p}_{k+1} \in \mathbb{R}^d$ be affinely independent. Let $\mathbf{x} \in \text{aff}\{\mathbf{p}_1, \ldots, \mathbf{p}_{k+1}\}$. The barycentric coordinates of $\mathbf{x}$ with respect to $\mathbf{p}_1, \ldots, \mathbf{p}_{k+1}$ are unique.*

If we are given an affine basis for $\mathbb{R}^d$ it is straightforward to compute the barycentric coordinates as follows.

Claim 2.5. *Let $\mathbf{p}_1, \ldots, \mathbf{p}_{d+1} \in \mathbb{R}^d$ be affinely independent. Let $\mathbf{x} \in \mathbb{R}^d$. The barycentric coordinates of $\mathbf{x}$ with respect to $\mathbf{p}_1, \ldots, \mathbf{p}_{d+1}$ are given by*

$$\boldsymbol{\alpha} = \mathbf{P}^{-1} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}, \tag{2.3}$$

where

$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \cdots \\ \alpha_{d+1} \end{bmatrix}, \qquad \mathbf{P} = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \cdots & \mathbf{p}_{d+1} \\ 1 & 1 & & 1 \end{bmatrix}. \tag{2.4}$$

Claim 2.5 can also be used to compute barycentric coordinates with respect to an affine basis of an affine subspace of $\mathbb{R}^d$. Let $\mathbf{p}_1, \ldots, \mathbf{p}_k \in \mathbb{R}^d$ be affinely independent. To compute the barycentric coordinates of $\mathbf{x} \in \text{aff}\{\mathbf{p}_1, \ldots, \mathbf{p}_k\}$, choose points $\mathbf{p}_{k+1}, \ldots, \mathbf{p}_{d+1} \in \mathbb{R}^d$ such that $\mathbf{p}_1, \ldots, \mathbf{p}_{d+1}$ are affinely independent and apply Claim 2.5 to find the barycentric coordinates of $\mathbf{x}$ with respect to $\mathbf{p}_1, \ldots, \mathbf{p}_{d+1}$. Since barycentric coordinates are unique

by Claim 2.4, it follows that $\alpha_{k+1} = \cdots = \alpha_{d+1} = 0$, so $\alpha_1, \ldots, \alpha_k$ are the barycentric coordinates of $\mathbf{x}$ with respect to $\mathbf{p}_1, \ldots, \mathbf{p}_k$.

The rest of this section will consider barycentric coordinates with respect to an affine basis for $\mathbb{R}^d$. Let $\mathbf{p}_1, \ldots, \mathbf{p}_{d+1}$ be such an affine basis. Let H_i be the hyperplane that contains $\mathbf{p}_1, \ldots, \mathbf{p}_{i-1}, \mathbf{p}_{i+1}, \ldots, \mathbf{p}_{d+1}$, i.e. $H_i = \text{aff}\left(\{\mathbf{p}_1, \ldots, \mathbf{p}_{d+1}\} \setminus \{\mathbf{p}_i\}\right)$. We say that H_i is the *hyperplane opposing* $\mathbf{p}_i$ (Recall that a hyperplane is an affine subspace with dimension $d-1$.). It follows from the discussion above that if $\mathbf{x} \in H_i$, then its i^{th} barycentric coordinate will be zero, $\alpha_i = 0$.

The barycentric coordinates of $\mathbf{x} \in \mathbb{R}^d$ are affine in $\mathbf{x}$ by Claim 2.5. Also, $\alpha_i = 0$ if $\mathbf{x} \in H_i$, and $\alpha_i = 1$ if $\mathbf{x} = \mathbf{p}_i$. From these facts it is not too surprising that the α_i may be interpreted as a signed, scaled distance from H_i. The following definitions and claim make this concept concrete.

The distance between a point $\mathbf{x} \in \mathbb{R}^d$ and a nonempty set $A \subseteq \mathbb{R}^d$ is a well defined quantity [Web94] given by

$$\delta(\mathbf{x}, A) := \inf_{\mathbf{z} \in A} \|\mathbf{x} - \mathbf{z}\|, \tag{2.5}$$

where $\|\cdot\|$ is the standard Euclidean norm. In the case of a hyperplane H, the distance $\delta(\mathbf{x}, H)$ can be computed directly given an implicit representation of H. Let $(\mathbf{a}, c)$ be an implicit representation for H, that is, $H = \left\{\mathbf{z} \in \mathbb{R}^d | \mathbf{a}^{\mathrm{T}}\mathbf{z} + c = 0\right\}$. A simple calculation shows that the distance of a point $\mathbf{x} \in \mathbb{R}^d$ to H is given by

$$\delta(\mathbf{x}, H) = \frac{\left|\mathbf{a}^{\mathrm{T}}\mathbf{x} + c\right|}{\|\mathbf{a}\|}.$$

Barycentric coordinates are related to a "signed distance." Let H_i be the hyperplane opposing $\mathbf{p}_i$. Let $\mathbf{x} \in \mathbb{R}^d$. We define $\delta_s(\mathbf{x}, H_i)$ as the *signed distance* of $\mathbf{x}$ from H_i. That is, $|\delta_s(\mathbf{x}, H_i)| = \delta(\mathbf{x}, H_i)$, and $\delta_s(\mathbf{x}, H_i) > 0$ for $\mathbf{x}$ on the same side of H_i as $\mathbf{p}_i$, and $\delta_s(\mathbf{x}, H_i) < 0$ for $\mathbf{x}$ on the opposite side of H_i. The following claim demonstrates the relationship between barycentric coordinate α_i, signed distance $\delta_s(\mathbf{x}, H_i)$, and the (positive) scaling factor $\delta(\mathbf{p}_i, H_i)$.

Claim 2.6. *Let* $\mathbf{x} \in \mathbb{R}^d$. *Let* $\boldsymbol{\alpha} = \begin{bmatrix}\alpha_1 & \cdots & \alpha_{d+1}\end{bmatrix}^{\mathrm{T}}$ *be the barycentric coordinates of* $\mathbf{x}$ *with respect to the affinely independent points* $\mathbf{p}_1, \ldots, \mathbf{p}_{d+1} \in \mathbb{R}^d$. *Then* $\delta_s(\mathbf{x}, H_i) = \alpha_i \delta(\mathbf{p}_i, H_i)$.

Figure 2.1 illustrates the relationship between barycentric coordinates and distances. Claim 2.6 interprets each of the barycentric coordinates of $\mathbf{x}$ as the distance from $\mathbf{x}$ to the affine hull of d elements of the affine basis.

What about the (unsigned) distance of $\mathbf{x}$ to the affine hull of $k < d$ elements of the affine basis? In this case, a quadratic form of the barycentric coordinates provides the desired distance, as demonstrated by the following claim.

Claim 2.7. *Let* $\mathbf{x} \in \mathbb{R}^d$. *Let* $\boldsymbol{\alpha} = \begin{bmatrix}\alpha_1 & \cdots & \alpha_{d+1}\end{bmatrix}^{\mathrm{T}}$ *be barycentric coordinates of* $\mathbf{x}$ *with respect to the affinely independent points* $\mathbf{p}_1, \ldots, \mathbf{p}_{d+1}$. *The distance from* $\mathbf{x}$ *to the*

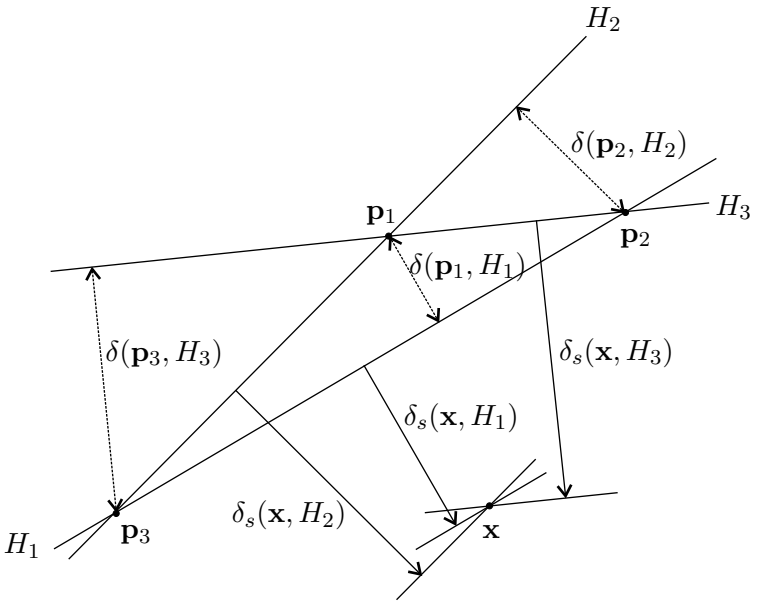

Figure 2.1: Illustration of barycentric coordinates as distances. Let $\alpha_1, \alpha_2, \alpha_3$ be the barycentric coordinates of the point $\mathbf{x}$ with respect to $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$. Claim 2.6 shows that the signed distance between $\mathbf{x}$ and H_i is given by $\delta_s(\mathbf{x}, H_i) = \alpha_i \delta(\mathbf{p}_i, H_i)$. In this example, $\alpha_1 < 0$ and $\alpha_2, \alpha_3 > 0$.

affine subspace $A = \text{aff}(\{\mathbf{p}_1, \ldots \mathbf{p}_k\})$, *is given by* $\delta(\mathbf{x}, A) = \sqrt{\bar{\boldsymbol{\alpha}}_s{}^{\mathrm{T}} G \bar{\boldsymbol{\alpha}}_s}$, *where* $\bar{\boldsymbol{\alpha}}_s = \begin{bmatrix} \alpha_{k+1} & \alpha_{k+2} & \cdots & \alpha_{d+1} \end{bmatrix}^{\mathrm{T}}$ *and* $G \in \mathbb{R}^{(d-k+1)\times(d-k+1)}$, *defined in Equation A.6, is a positive definite matrix whose entries depend only on* $\mathbf{p}_1, \ldots, \mathbf{p}_k$.

The proof of Claim 2.7 is an application of Gram determinants. This result plays an important role in the proof of Lemma 5.1.

2.1.3 Simplices

A set $C \subseteq \mathbb{R}^d$ is said to be *convex* if whenever $\mathbf{p}_1, \mathbf{p}_2 \in C$, then the line segment $[\mathbf{p}_1, \mathbf{p}_2]$ given by

$$[\mathbf{p}_1, \mathbf{p}_2] = \left\{ \mathbf{p} \in \mathbb{R}^d \mid \mathbf{p} = (1-\alpha)\mathbf{p}_1 + \alpha \mathbf{p}_2,\ \alpha \in [0,1] \right\}$$

is in C. i.e. $[\mathbf{p}_1, \mathbf{p}_2] \subseteq C$. Figure 2.2 illustrates some convex and nonconvex sets.

The *convex hull* of a set $S \subseteq \mathbb{R}^d$, denoted $\text{conv}(S)$, is the intersection of all convex sets containing S,

$$\text{conv}(S) = \bigcap_{\substack{S \subseteq C \\ C \text{ convex}}} C.$$

Since the intersection of an arbitrary number of convex sets is convex (Theorem 2.1.3 in [Web94]), the convex hull of a set is convex. Thus, $\text{conv}(S)$ is the smallest convex set containing S.

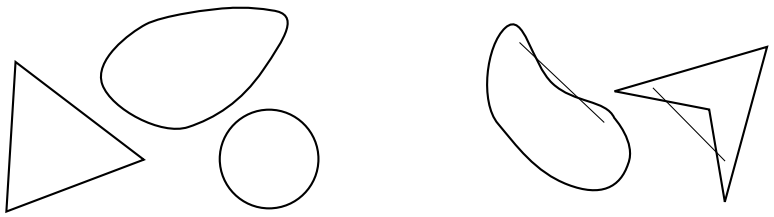

Figure 2.2: The sets on the left are convex. The sets on the right are not convex, with a line indicating where they fail.

Let $\mathbf{p}_1, \ldots, \mathbf{p}_m \in \mathbb{R}^d$. Let $\alpha_1, \ldots, \alpha_m \in \mathbb{R}$ such that $\sum_{i=1}^m \alpha_i = 1$ and for all i, $\alpha_i \geq 0$. Then $\mathbf{p} = \sum_{i=1}^m \alpha_i \mathbf{p}_i$ is a *convex combination* of the points $\mathbf{p}_1, \ldots, \mathbf{p}_m$.

Claim 2.8. *Let $S \subset \mathbb{R}^d$. Let A be the set of all convex combinations of points in S,*

$$A = \left\{ \mathbf{x} \in \mathbb{R}^d \,\middle|\, \mathbf{x} = \sum_{i=1}^k \alpha_i \mathbf{p}_i, \;\; \sum_{i=1}^k \alpha_i = 1 \;,\;\; \alpha_i \geq 0, \;\; \mathbf{p}_i \in S \right\}. \tag{2.6}$$

Then $A = \text{conv}(S)$.

A face of a convex set $A \subset \mathbb{R}^d$ is a convex subset $B \subseteq A$ such that whenever $\alpha\mathbf{x} + (1-\alpha)\mathbf{y} \in B$, where $\mathbf{x}, \mathbf{y} \in A$ and $0 < \alpha < 1$, then $\mathbf{x}, \mathbf{y} \in B$ [Web94]. Every convex set A has faces $\emptyset$ and A, which are called improper faces. All other faces are called proper faces. A face of dimension 0 is called an *extreme point*. An equivalent characterization for an extreme point is that the point $\mathbf{a}$ in convex set A is an extreme point if and only if whenever $\mathbf{a} = \frac{1}{2}(\mathbf{x} + \mathbf{y})$ for $\mathbf{x}, \mathbf{y} \in A$, then $\mathbf{x} = \mathbf{y} = \mathbf{a}$.

A *simplex* is the convex hull of a set of affinely independent points. Figure 2.3 illustrates simplices of various dimension. Let the simplex s be the convex hull of $k+1$ affinely independent points. The *dimension* of s is $\dim(s) := \dim(\text{aff}(s)) = k$, and we call s a k-simplex. As a result of Claim 2.1, a set of at most $d+1$ points in $\mathbb{R}^d$ can be affinely independent points in $\mathbb{R}^d$, and thus there are simplices of dimension $-1, 0, 1, \ldots, d$, where by convention $\emptyset$ is considered a simplex with $\dim(\emptyset) := -1$.

The extreme points of a simplex are called *vertices*, and the set of vertices for a simplex s is written as $\text{vert}(s)$.

Claim 2.9. *Let $\mathbf{p}_1, \ldots, \mathbf{p}_{k+1} \in \mathbb{R}^d$ be affinely independent. The simplex $s = \text{conv}(\{\mathbf{p}_1, \ldots, \mathbf{p}_{k+1}\})$ has the vertices $\text{vert}(s) = \{\mathbf{p}_1, \ldots, \mathbf{p}_{k+1}\}$.*

This claim is a result of the Krein-Milman Theorem [Web94], which states that any compact convex set in $\mathbb{R}^d$ is the convex hull of its extreme points. We can think of $\text{vert}(\cdot)$ as a pseudo-inverse of $\text{conv}(\cdot)$, since for a set of affinely independent points, $\{\mathbf{p}_1, \ldots, \mathbf{p}_{k+1}\} = \text{vert}(\text{conv}(\{\mathbf{p}_1, \ldots, \mathbf{p}_{k+1}\}))$. Note that $\dim(s) = \text{card}(\text{vert}(s)) - 1$, where $\text{card}(\cdot)$ is cardinality.

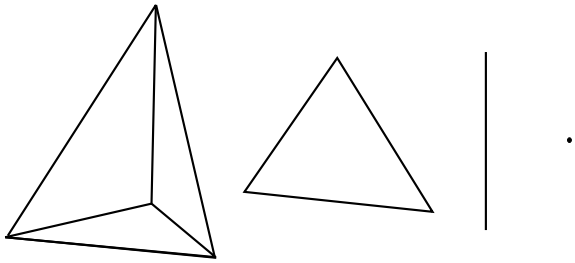

Figure 2.3: Examples of 3-,2-,1-, and 0-simplices, from left to right.

The faces of a simplex are also simplices. If s is a k-simplex and $\mathcal{V} \subseteq \text{vert}(s)$ with $\text{card}(\mathcal{V}) = l + 1$, then $\text{conv}(\mathcal{V})$ is an l-simplex face of s. A k-simplex has $\binom{k+1}{l+1}$ faces of dimension l. In $\mathbb{R}^d$, a $d-1$ dimensional face of a d-simplex is more specifically called a *facet*. The structure of the faces admits a partial order of simplices. Given two simplices s_1, s_2, we say that $s_1 \preceq s_2$ if and only if $\text{vert}(s_1) \subseteq \text{vert}(s_2)$, that is, s_1 is a face of s_2.

Let $\mathbf{\Delta}_d$ be the *standard d-simplex*, defined as

$$\mathbf{\Delta}_d := \left\{ \alpha \in \mathbb{R}^{d+1} | \alpha_i \geq 0, \sum_{i=1}^{d+1} \alpha_i = 1 \right\}. \tag{2.7}$$

$\mathbf{\Delta}_d$ is the convex hull of the $d+1$ standard basis vectors $\mathbf{e}_1, \ldots, \mathbf{e}_{d+1}$ for $\mathbb{R}^{d+1}$. Note that if $\mathbf{p}_1, \ldots, \mathbf{p}_{k+1} \in \mathbb{R}^d$ and $\alpha \in \mathbf{\Delta}_k$, then $\sum_{i=1}^{k+1} \alpha_i \mathbf{p}_i$ is a convex combination of the $\mathbf{p}_i$'s.

In this dissertation we will predominantly be interested in d-simplices, so we adopt the convention to distinguish d-simplices from simplices of unspecified dimension. For simplices in $\mathbb{R}^d$, an s indicates a simplex of any dimension 0 to d, while the over-bar, such as $\bar{s}$, indicates a d-simplex.

The following claim shows how to calculate the signed volume of a d-simplex. Note that the matrix $\mathbf{P}$ is also used to calculate the barycentric coordinates of a point (Equation 2.3). The sign of the volume indicates whether the transformation of the simplex to the standard d-simplex is orientation preserving (positive volume) or orientation reversing (negative volume).

Claim 2.10. *Let $\mathbf{p}_1, \ldots, \mathbf{p}_{d+1} \in \mathbb{R}^d$ be affinely independent. The signed volume of the simplex $\bar{s} = \text{conv}(\{\mathbf{p}_1, \ldots, \mathbf{p}_{d+1}\})$ is $V(\bar{s}) = \frac{1}{d!} \det \tilde{\mathbf{P}} = \frac{1}{d!} \det \mathbf{P}$, where*

$$\tilde{\mathbf{P}} = \begin{bmatrix} \mathbf{p}_1 - \mathbf{p}_{d+1} & \mathbf{p}_2 - \mathbf{p}_{d+1} & \cdots & \mathbf{p}_d - \mathbf{p}_{d+1} \end{bmatrix} \qquad \mathbf{P} = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \cdots & \mathbf{p}_{d+1} \\ 1 & 1 & & 1 \end{bmatrix}.$$

2.1.4 The Dilation and its Properties

The dilation is a geometric construction developed for the proof of Lemma 5.1 in order to measure how the approximation's mismatched triangulation affects the least squares fits.

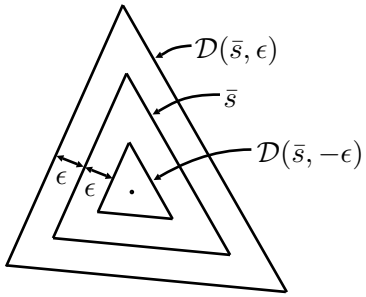

Figure 2.4: Dilations of 2-simplex. The 2-simplex $\bar{s}$ with its ϵ and $-\epsilon$ dilations. The point at the center is where the dilation degenerates to a single point.

The following discussion will define the dilation of a simplex and establish some geometric facts about it.

Let $\bar{s}$ be a d-simplex with $\text{vert}(\bar{s}) = \{\mathbf{p}_1, \ldots, \mathbf{p}_{d+1}\}$. The ϵ dilation of $\bar{s}$, written $\mathcal{D}(\bar{s}, \epsilon)$, is defined as

$$\mathcal{D}(\bar{s}, \epsilon) := \left\{ x = \sum_{j=1}^{d+1} \alpha_j \mathbf{p}_j \middle| \sum_{j=1}^{d+1} \alpha_j = 1 \text{ with } \alpha_j \geq \frac{-\epsilon}{\delta(\mathbf{p}_j, H_j)} \right\}, \tag{2.8}$$

where H_j is the hyperplane opposing $\mathbf{p}_j$, and $\delta(\cdot, \cdot)$ is the distance between a point and a set, defined in Equation 2.5. Figure 2.4 illustrates the dilation of a 2-simplex. $\mathcal{D}(\bar{s}, \epsilon)$ is well defined for

$$\epsilon \geq -\left(\sum_{j=1}^{d+1} 1/\delta(\mathbf{p}_j, H_j) \right)^{-1}.$$

When equality holds, $\mathcal{D}(\bar{s}, \epsilon)$ is a single point, otherwise it is a d-simplex with facets parallel to $\bar{s}$, but distance $|\epsilon|$ away, with $\bar{s} \subseteq \mathcal{D}(\bar{s}, \epsilon)$ for $\epsilon > 0$ and $\mathcal{D}(\bar{s}, \epsilon) \subseteq \bar{s}$ for $\epsilon < 0$. These properties are established by the following claim.

Claim 2.11. *Let $\bar{s} \subseteq \mathbb{R}^d$ be a d-simplex with vertices $\mathbf{p}_1, \ldots, \mathbf{p}_{d+1}$. Let*

$$\epsilon_{\min} = -\left(\sum_{i=1}^{d+1} 1/\delta(\mathbf{p}_i, H_i) \right)^{-1}, \tag{2.9}$$

where H_i is the opposing hyperplane to $\mathbf{p}_i$. $\mathcal{D}(\bar{s}, \epsilon_{\min})$ is a single point. For $\epsilon > \epsilon_{\min}$, $\mathcal{D}(\bar{s}, \epsilon)$ is a d-simplex, with faces parallel to and translated distance $|\epsilon|$ away from the faces of $\bar{s}$. For $\epsilon_{\min} \leq \epsilon \leq 0$, $\mathcal{D}(\bar{s}, \epsilon) \subseteq \bar{s}$, while for $\epsilon \geq 0$, $\bar{s} \subseteq \mathcal{D}(\bar{s}, \epsilon)$.

The specific result needed for Lemma 5.1 requires a relationship between incident d-simplices expressed using the dilation. The following claim establishes the necessary result.

Claim 2.12. *Let $\bar{s}_a, \bar{s}_b \subseteq \mathbb{R}^d$ be incident d-simplices, that is $\bar{s}_a \cap \bar{s}_b = s_{ab}$, where $s_{ab} \preceq \bar{s}_a, \bar{s}_b$ is a $(k-1)$-simplex, $1 \leq k < d+1$. Let* $\text{vert}(\bar{s}_a) = \{\mathbf{p}_1, \ldots, \mathbf{p}_{d+1}\}$, $\text{vert}(\bar{s}_b) =$

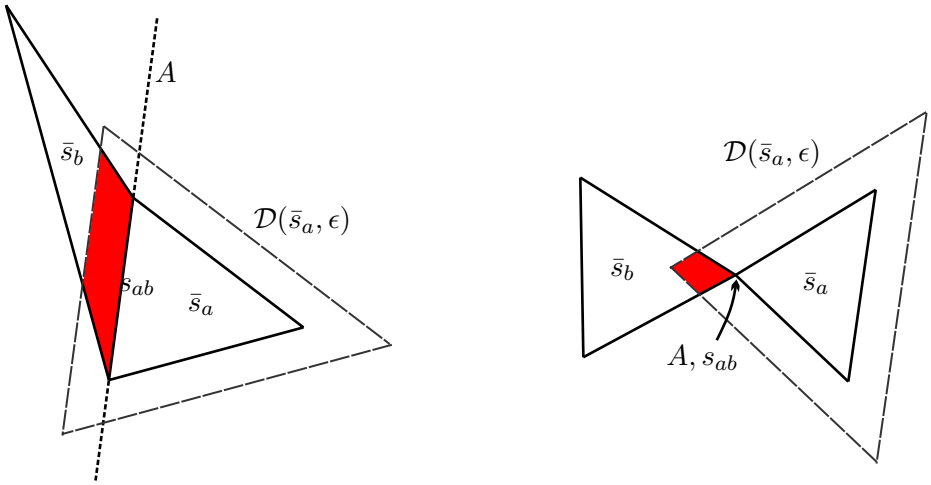

Figure 2.5: Two examples of Claim 2.12 in $\mathbb{R}^2$. Any $\mathbf{x} \in \mathcal{D}(\bar{s}_a, \epsilon) \cap \bar{s}_b$ (the shaded region) will satisfy $\delta(\mathbf{x}, A) < \kappa_{a,b}\epsilon$, where $\kappa_{a,b}$ is a constant determined by $\bar{s}_a$ and $\bar{s}_b$ and $A = \text{aff } s_{ab}$. This result holds for arbitrary dimension and for shared faces of any dimension.

$\{\mathbf{p}_1, \ldots, \mathbf{p}_k, \mathbf{q}_{k+1}, \ldots, \mathbf{q}_{d+1}\}$, *and* $\text{vert}(s_{ab}) = \{\mathbf{p}_1, \ldots, \mathbf{p}_k\}$. *Let* $A = \text{aff } s_{ab}$. *Then* $\exists \kappa_{a,b} > 0$ *such that* $\forall \epsilon > 0$, *if* $\mathbf{x} \in \mathcal{D}(\bar{s}_a, \epsilon) \cap \bar{s}_b$ *then* $\delta(\mathbf{x}, A) < \kappa_{a,b}\epsilon$.

Figure 2.5 illustrates Claim 2.12 with a pair of examples in $\mathbb{R}^2$.

2.2 Triangulations

The intuitive concept of "triangulation" as a partition of some space into simplices is found in many fields, such as algebraic topology, computational geometry, graphics, and finite element analysis. Unfortunately there is no uniform definition of triangulation across fields, and even within computational geometry there is no uniform formal definition of triangulation [Ede01]. Formal definitions of triangulation are built on the definition of simplicial complex from algebraic topology. A simplicial complex, defined below, is a collection of simplices and their faces that have been "glued" together in an appropriate way. In algebraic topology, a triangulation of a topological space $\mathcal{X}$ is a simplicial complex $\mathcal{K}$ coupled with a homeomorphism[2] between the underlying space of the simplicial complex $|\mathcal{K}|$ and $\mathcal{X}$ [Spa66, Hat01]. In this case, the triangulation captures the combinatorial features of the space, but in $\mathcal{X}$ the simplices have been deformed through the homeomorphism so that they are no longer necessarily geometric simplices. A more constrained notion of triangulation is used in computational geometry. The definition of triangulation presented in Section 2.2.2 is akin to that used in computational geometry, but slightly more general than most definitions in that the underlying space is allowed to be nonconvex. First it is necessary to

[2]Recall that a homeomorphism is a continuous map that has a continuous inverse.

define simplicial complex.

2.2.1 Simplicial Complexes

The definition of simplicial complex presented here, adapted from [Ede01], emphasizes the geometric structure of the complex and is hence less abstract than some of the traditional literature in algebraic topology circa the 1950s, such as example [Spa66]. This definition also shares much in common with the generalization of simplicial complexes, called $\mathbf{\Delta}$-complexes, found in modern algebraic topology literature such as [Hat01]. The discussion of the differences between simplicial- and $\mathbf{\Delta}$-complexes may be found in A.3.1 in the appendix.

A *geometric simplicial complex* is a collection $\mathcal{K}$ of simplices in $\mathbb{R}^d$ satisfying

1. if $s_1 \in \mathcal{K}$ and $s_2 \preceq s_1$, then $s_2 \in \mathcal{K}$
2. if $s_1, s_2 \in \mathcal{K}$, then $s_1 \cap s_2 \preceq s_1, s_2$.

We generally refer to a geometric simplicial complex simply as "simplicial complex," only including "geometric" when necessary to distinguish it from an abstract simplicial complex, defined below. Condition 1 for a simplicial complex says that if a simplex is in the complex, then all its faces must also be in the complex. This gives simplicial complexes a recursive structure. Condition 2 says that simplices in the complex may intersect only at shared faces and in no other way. A collection of simplices that violates condition 2 is said to have an "improper intersection." Figure 2.6 illustrates a geometric simplicial complex as well as several of the operations described below.

The *dimension* of a simplicial complex $\mathcal{K}$ is

$$\dim \mathcal{K} := \sup_{s \in \mathcal{K}} \dim s. \tag{2.10}$$

The *vertex set* of a simplicial complex is

$$\operatorname{vert}(\mathcal{K}) := \left\{ \mathbf{x} \in \mathbb{R}^d \,|\mathbf{x} \in \operatorname{vert}(s), s \in \mathcal{K} \right\}. \tag{2.11}$$

The *underlying space* of a simplicial complex is

$$|\mathcal{K}| := \left\{ \mathbf{x} \in \mathbb{R}^d \,|\mathbf{x} \in s, s \in \mathcal{K} \right\}. \tag{2.12}$$

A *subcomplex* is a subset of a simplicial complex that is itself a simplicial complex. The *closure* of a subset $\mathcal{L} \subseteq \mathcal{K}$ is the smallest subcomplex that contains $\mathcal{L}$,

$$\operatorname{Cl} \mathcal{L} := \left\{ s \in \mathcal{K} \middle| s \leq s', s' \in \mathcal{L} \right\} \tag{2.13}$$

The *star* of a simplex s is the set of all simplices that contain s,

$$\operatorname{St} s := \left\{ s' \in \mathcal{K} \middle| s \leq s' \right\}. \tag{2.14}$$

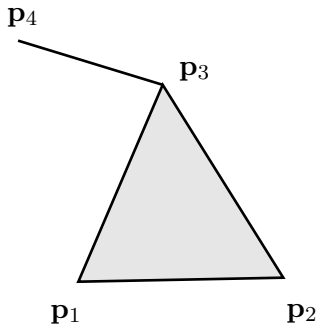

Figure 2.6: An example of a simplicial complex $\mathcal{K}$ in $\mathbb{R}^2$.

$\mathcal{K} = \{\{\mathbf{p}_1\}, \{\mathbf{p}_2\}, \{\mathbf{p}_3\}, \{\mathbf{p}_4\}, s_1, s_2, s_3, s_4, s_5\}$, where

$s_1 = \text{conv}(\{\mathbf{p}_1, \mathbf{p}_2\})$, $s_2 = \text{conv}(\{\mathbf{p}_1, \mathbf{p}_3\})$, $s_3 = \text{conv}(\{\mathbf{p}_2, \mathbf{p}_3\})$,

$s_4 = \text{conv}(\{\mathbf{p}_3, \mathbf{p}_4\})$, $s_5 = \text{conv}(\{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\})$.

Examples of operations on simplicial complex $\mathcal{K}$,

$\dim \mathcal{K} = 2$

$\text{vert}(\mathcal{K}) = \{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4\}$

$\text{Cl}(\{s_5\}) = \{\{\mathbf{p}_1\}, \{\mathbf{p}_2\}, \{\mathbf{p}_3\}, s_1, s_2, s_3, s_5\}$

$\text{Cl}(\{\{\mathbf{p}_4\}, s_4\}) = \{\{\mathbf{p}_3\}, \{\mathbf{p}_4\}, s_4\}$

$\text{St}(\{\mathbf{p}_1\}) = \{\{\mathbf{p}_1\}, s_1, s_2, s_3\}$

$\text{St}(s_1) = \{s_5\}$

$\text{Lk}(\{\mathbf{p}_1\}) = \{s_2, s_3\}$

$\text{Lk}(d_5) = \{\{\mathbf{p}_4\}\}$

The star is not in general a subcomplex. The *link* of a simplex s,

$$\operatorname{Lk} s := \left\{ s' \in \operatorname{Cl}\operatorname{St} s \middle| s \cap s' = \emptyset \right\}. \tag{2.15}$$

is the set of all simplices in the closure of the star that do not intersect s.

An *abstract simplicial complex* is a collection $\mathcal{S}$ of finite sets satisfying if $\alpha \in \mathcal{S}$ and $\beta \subseteq \alpha$ then $\beta \in \mathcal{S}$. Moreover if $\beta \subseteq \alpha$, then we say that β is a face of α and write $\beta \preceq \alpha$. The sets in $\mathcal{S}$ are also called simplices (we use the term "abstract simplices" in cases where the context is not clear), and the dimension of $\alpha \in \mathcal{S}$ is $\dim \alpha := \operatorname{card}(\alpha) - 1$. The *vertex set* of an abstract simplicial complex is the set $\left\{ x \middle| x \in \alpha, \alpha \in \mathcal{S} \right\}$. The concepts of subcomplex, closure, star and link may be similarly extended to abstract simplicial complexes.

Notice that given a geometric simplicial complex $\mathcal{K}$, the collection of finite sets

$$\{\{\operatorname{vert}(s)\} | s \in \mathcal{K}\} \tag{2.16}$$

is an abstract simplicial complex. The abstract simplicial complex represents the combinatorial information about the underlying space of the geometric simplicial complex.[3]

A geometric simplicial complex $\mathcal{K}$ in $\mathbb{R}^d$ can be parameterized [4] by the pair $(P, \mathcal{S})$, where P is an indexed set of n unique points in $\mathbb{R}^d$,

$$P = \left\{ \mathbf{p}_1, \quad \mathbf{p}_2, \quad \ldots, \quad \mathbf{p}_n \right\},$$

and $\mathcal{S}$ is an abstract simplicial complex with vertex set $\{1, 2, \ldots, n\}$. Let

$$\mathcal{K}(P, \mathcal{S}) = \left\{ \operatorname{conv}(P(\alpha)) \middle| \alpha \in \mathcal{S} \right\},$$

where[5] $P(\alpha) = \{ \mathbf{p}_i \in P \,|\, i \in \alpha \}$. The following claim gives the conditions on P and $\mathcal{S}$ such that $\mathcal{K}(P, \mathcal{S})$ is a simplicial complex.

Claim 2.13. *$\mathcal{K}(P, \mathcal{S})$ is a geometric simplicial complex if and only if*

1. *$\forall \alpha \in \mathcal{S}, P(\alpha)$ are affinely independent*
2. *if $s_1, s_2 \in \mathcal{K}(P, \mathcal{S})$, then $s_1 \cap s_2 \preceq s_1, s_2$.*

Moreover, if these properties hold, then $\operatorname{vert}(\mathcal{K}(P, \mathcal{S})) = P$.

If $\mathcal{K}(P, \mathcal{S})$ is a geometric simplicial complex, that is, if it is a collection of well-shaped simplices and there are no improper intersections of the simplices, then we say that $\mathcal{K}(P, \mathcal{S})$ is an *embedding* of $\mathcal{S}$.

[3] In [Spa66], a simplicial complex is defined as what we call an abstract simplicial complex here. A discussion of the differences between the traditional definition and the more geometric definition may be found in A.3.1 in the appendix.

[4] This is not formally a parameterization, because there are some pairs $(P, \mathcal{S})$ for which $\mathcal{K}(P, \mathcal{S})$ is not a geometric simplicial complex. However, for any geometric simplicial complex $\mathcal{K}$, we can write down a pair $(P, \mathcal{S})$ such that $\mathcal{K} = \mathcal{K}(P, \mathcal{S})$.

[5] Formally, an indexed set P of n points in $\mathbb{R}^d$ is a map $P : \{1, 2, \ldots, n\} \to \mathbb{R}^d$. The i^{th} member of P is $P(i)$, which we generally write as $\mathbf{p}_i$ for notational convenience. Here we extend the notion of P to sets. Let $\alpha \subseteq \{1, 2, \ldots, n\}$, then $P(\alpha) := \left\{ P(i) \middle| i \in \alpha \right\}$

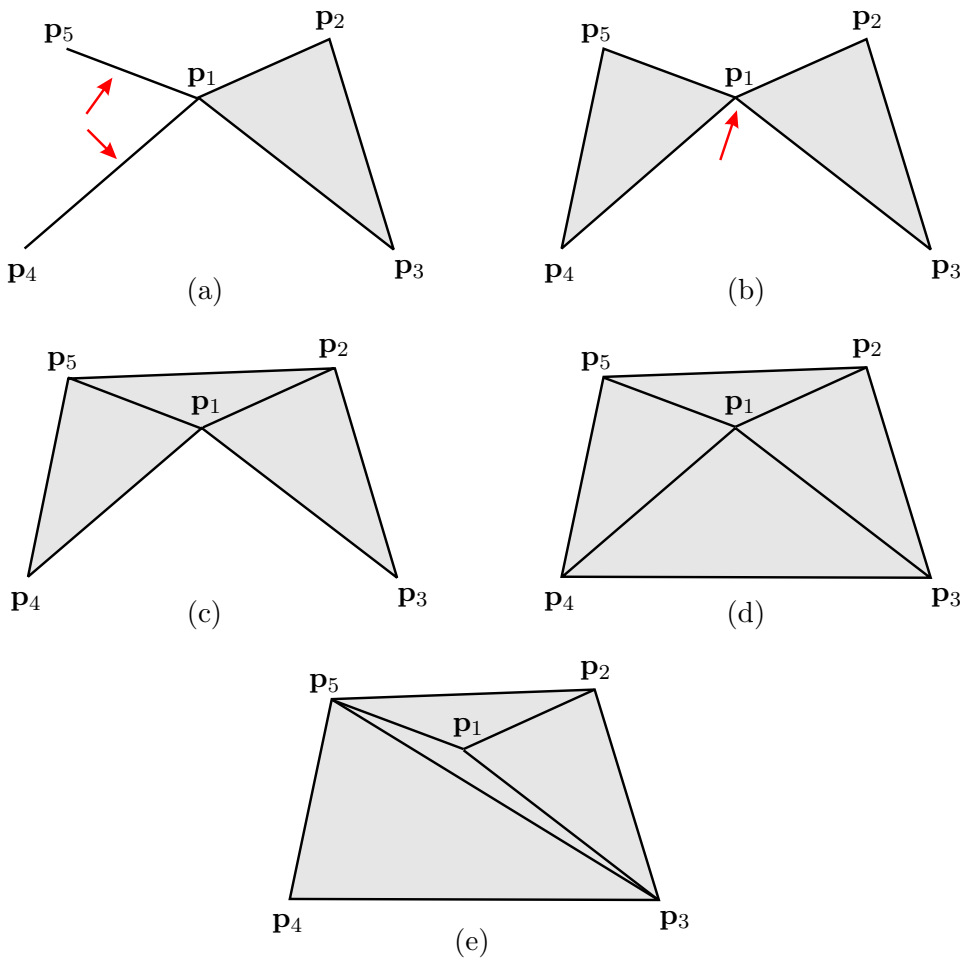

Figure 2.7: Examples of geometric simplicial complexes. These five different geometric simplicial complexes in $\mathbb{R}^2$ all have the same set of vertices, namely $P = \{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4, \mathbf{p}_5\}$, but different combinatorial structures. The combinatorial structures are:

$\mathcal{S}_a = \{\ \{1,2,3\}, \{1,2\}, \{1,3\}, \{1,4\}, \{1,5\}, \{2,3\}, \{1\}, \{2\}, \{3\}, \{4\}, \{5\}\ \}$

$\mathcal{S}_b = \{\ \{1,2,3\}, \{1,4,5\}, \{1,2\}, \{1,3\}, \{1,4\}, \{1,5\}, \{2,3\}, \{4,5\}, \{1\}, \{2\}, \{3\}, \{4\}, \{5\}\ \}$

$\mathcal{S}_c = \{\ \{1,2,3\}, \{1,2,5\}, \{1,4,5\}, \{1,2\}, \{1,3\}, \{1,4\}, \{1,5\}, \{2,3\}, \{2,5\},$
$\{4,5\}, \{1\}, \{2\}, \{3\}, \{4\}, \{5\}\ \}$

$\mathcal{S}_d = \{\ \{1,2,3\}, \{1,2,5\}, \{1,3,4\}, \{1,4,5\}, \{1,2\}, \{1,3\}, \{1,4\}, \{1,5\}, \{2,3\},$
$\{2,5\}, \{3,4\}, \{4,5\}, \{1\}, \{2\}, \{3\}, \{4\}, \{5\}\ \}$

$\mathcal{S}_e = \{\ \{1,2,3\}, \{1,2,5\}, \{1,3,5\}, \{3,4,5\}, \{1,2\}, \{1,3\}, \{1,5\}, \{2,3\},$
$\{2,5\}, \{3,4\}, \{3,5\}, \{4,5\}, \{1\}, \{2\}, \{3\}, \{4\}, \{5\}\ \}$

The complexes in (c), (d), and (e) are also triangulations. The complexes in (a) and (b) are not triangulations because their underlying spaces fail to be manifolds with boundary, as indicated by the arrows. The triangulations in (d) and (e) have the same underlying space, but different combinatorial structure. Note that for a triangulation, the rest of the combinatorial structure can be deduced from the d-simplices of $\mathcal{S}$.

2.2.2 Definition of Triangulation

There is no uniform formal definition of triangulation in computational geometry [Ede01], though it is generally a geometric simplicial complex with a few extra conditions. For example, in [ES96] a triangulation is a simplicial complex whose underlying space is the convex hull of the vertex set. This definition is well suited to the computational geometry problem of computing a triangulation of a set of vertices, discussed in Section 2.3, but the requirement for a convex underlying space is stricter than desired for the present purpose. The key concept in the definition of triangulation used here is that a triangulation has good local volume properties everywhere. The underlying space has no "thin" spots. Figure 2.7 provides five geometric simplicial complexes, three of which are triangulations and two of which are not.

A *triangulation* $\mathcal{T}$ is a k-dimensional geometric simplicial complex in $\mathbb{R}^d$ for which the underlying space is a connected k-manifold with boundary. In this dissertation, we will primarily be interested in triangulations for which the underlying space is a simply-connected d-manifold with boundary embedded in $\mathbb{R}^d$.

Since a triangulation is a geometric simplicial complex, it can be parameterized in the same manner. We write $\mathcal{T}(P, \mathcal{S})$ rather than $\mathcal{K}(P, \mathcal{S})$ to indicate that the potential geometric simplicial complex generated by the pair $(P, \mathcal{S})$ would be a triangulation. As with geometric simplicial complexes, if $\mathcal{T}(P, \mathcal{S})$ is a triangulation, then we say that $\mathcal{T}(P, \mathcal{S})$ is an *embedding* of $\mathcal{S}$.

Given that $\mathcal{K}(P, \mathcal{S})$ is a geometric simplicial complex, whether it is also a triangulation is determined entirely by $\mathcal{S}$. Unfortunately the author is unaware of any proven sufficient conditions on $\mathcal{S}$ to yield a triangulation, although there are a number of straightforward necessary conditions.

Claim 2.14. *Let $\mathcal{K}(P, \mathcal{S})$ be a geometric simplicial complex that is also a triangulation, that is $|\mathcal{K}(P, \mathcal{S})|$ is a connected k-manifold with boundary. Then*

- *$\forall \alpha \in \mathcal{S}$, $\exists$ a k-simplex $\alpha' \in \mathcal{S}$ such that α is a face of α'*
- *For any $\alpha, \beta \in \mathcal{S}$, $\exists$ k-simplices $\gamma_1, \ldots, \gamma_n \in \mathcal{S}$ such that $\alpha \cap \gamma_1 \neq \emptyset$, $\beta \cap \gamma_n \neq \emptyset$, and $\gamma_i \cap \gamma_{i+1}$ is $(k-1)$-simplex for each $i = 1, \ldots, n-1$.*

2.2.3 Notation and Geometric Constants Related to Triangulations

In this dissertation we are primarily concerned with d-dimensional triangulations in $\mathbb{R}^d$. From Claim 2.14, it is sufficient to list the d-simplices of a triangulation. All other simplices in the complex are faces of the d-simplices. For this reason, when writing about a triangulation $\mathcal{T}(P, \mathcal{S})$, we will treat $\mathcal{S}$ as if it is specified by an indexed list of N_T abstract d-simplices, $\bar{\alpha}_i$, $i = 1, \ldots, N_T$. The corresponding geometric d-simplices are $\bar{s}_i$, $i = 1, \ldots, N_T$, where $\bar{s}_i = \text{conv}(P(\bar{\alpha}_i))$. The following notation is a convenient way to indicate the dimension of

the shared face between two d-simplices of $\mathcal{T}(P, \mathcal{S})$,

$$d_{i,j} := \dim(\bar{s}_i \cap \bar{s}_j) = \operatorname{card}(\operatorname{vert}(\bar{s}_i \cap \bar{s}_j)) - 1, \tag{2.17}$$

Another bit of useful notation is the number of d-simplices in $\operatorname{St}\{\mathbf{p}_i\}$,

$$N_i = \sum_{\bar{s}_j \in \operatorname{St} \mathbf{p}_i} 1. \tag{2.18}$$

Several characteristic measurements of a triangulation are used in establishing facts about PL functions and proving convergence of the MINVAR algorithm. These constants may be interpreted geometrically as minima or maxima of different measures of the "radii" of d-simplices in the parameterized triangulation $\mathcal{T}(P, \mathcal{S})$.

The maximum inter-vertex distance is the maximum distance between vertices in a d-simplex,

$$r_1(P, \mathcal{S}) := \max_{\substack{\bar{s}_i \in \mathcal{T}(P,\mathcal{S}) \\ \mathbf{p}_j, \mathbf{p}_k \in \bar{s}_i}} \|\mathbf{p}_j - \mathbf{p}_k\| . \tag{2.19}$$

The minimum orthogonal distance is the minimum distance from a vertex to the hyperplane containing one of its opposing facets,

$$r_2(P, \mathcal{S}) := \min_{\substack{\bar{s}_i \in \mathcal{T}(P,\mathcal{S}) \\ \mathbf{p}_j \in \operatorname{vert}(\bar{s}_i)}} \delta(\mathbf{p}_j, \operatorname{aff}(\operatorname{vert}(\bar{s}_i) \setminus \{\mathbf{p}_j\})) . \tag{2.20}$$

The dilation radius is given by

$$r_3(P, \mathcal{S}) = \min_{\bar{s}_i \in \mathcal{T}(P,\mathcal{S})} \sup \left\{ \epsilon \,\middle|\, \mathcal{D}(\bar{s}_i, \epsilon) \subseteq |\operatorname{Cl} \operatorname{St} \bar{s}_i| \right\}, \tag{2.21}$$

where $\mathcal{D}(\cdot, \cdot)$ is the dilation, defined in Section 2.1.4. The dilation radius gives a bound on how far a d-simplex can be dilated while guaranteeing that it does not expand out past its neighbors. A single thin d-simplex in the triangulation can cause the dilation radius to be small. The property of the dilation radius is expressed by the following claim.

Claim 2.15. *Let $0 < \epsilon < r_3$. Let $\bar{s}_i, \bar{s}_j \in \mathcal{T}(P, \mathcal{S})$. Then $\mathcal{D}(\bar{s}_i, \epsilon) \cap \bar{s}_j \neq \emptyset$ if and only if $\bar{s}_i \cap \bar{s}_j \neq \emptyset$.*

The proof follows trivially from the definition.

2.3 Triangulation of a Set of Points

Finding a triangulation of a set of points is a standard problem in computational geometry and is conveniently stated using the terminology of the previous section: Given $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_N\}$, $\mathbf{p}_i \in \mathbb{R}^d$, find an abstract simplicial complex $\mathcal{S}$ such that $\mathcal{T}(P, \mathcal{S})$ is a triangulation with $|\mathcal{T}(P, \mathcal{S})| = \operatorname{conv}(P)$. $\mathcal{T}(P, \mathcal{S})$ is called a triangulation of P. In another similar problem, called constrained triangulation, a set of facets in addition to a set of

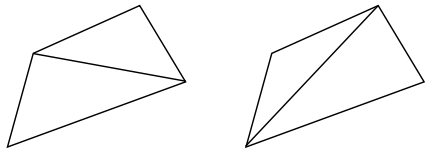

Figure 2.8: Edge flipping in two dimensions. These four points in two dimensions have two possible triangulations, shown above. In two dimensions it is possible to change between triangulations by "edge flipping," exchanging two triangles which combine to form a convex quadrilateral for the two triangles which have the opposite corners connected. A generalization to higher dimensions exists, but the number of d-simplices after a flip does not necessarily remain constant.

points is given, and one must find an abstract simplicial complex such that $\mathcal{T}(P, \mathcal{S})$ is a triangulation of P and $\mathcal{T}(P, \mathcal{S})$ contains the desired facets.

Typically one assumes that the points of P are in general position in $\mathbb{R}^d$, which in this case means that no $d+2$ points lie on a common hypersphere or hyperplane. This assumption can be relaxed, but in this section we assume at least that $\text{conv}(P)$ has non-empty interior. That is, the points in P do not all lie in the same hyperplane.

By Claim 2.14, it is sufficient to list the d-simplices in $\mathcal{S}$. Using this fact, we will represent $\mathcal{S}$ as an indexed list of its d-simplices. In general a given set of points P has many different triangulations. That is, many different abstract simplicial complexes $\mathcal{S}$ yield a triangulation $\mathcal{T}(P, \mathcal{S})$ satisfying $|\mathcal{T}(P, \mathcal{S})| = \text{conv}(P)$. In the two-dimensional case, one can change from one combinatorial structure to another through "edge flipping," illustrated in Figure 2.8. A generalization of edge flipping exists for higher dimensions, discussed in Section 2.3.4. Generally one wants to pick a specific triangulation from among the many possible triangulations of a set of points. The Delaunay triangulation is one of the most popular triangulations.

2.3.1 Delaunay Triangulation

The Delaunay triangulation is a canonical triangulation of a set of points in which the triangles are as "fat" as possible. The Delaunay triangulation may be characterized in several equivalent ways. The definition that most readily generalizes to higher dimensions is the circumcircle (or circumsphere for higher dimensions) criterion. Let P be an indexed set of points in $\mathbb{R}^d$ and let $\mathcal{S}$ be such that $\mathcal{T}(P, \mathcal{S})$ is a triangulation of P. For any d-simplex, there is a unique hypersphere passing through the $d+1$ points of $\text{vert}(\bar{s})$, called the circumsphere. Claim A.1 shows how to compute the circumsphere for a set of points. A triangulation is *Delaunay* if the circumsphere of each d-simplex $\bar{s} \in \mathcal{T}(P, \mathcal{S})$ contains no points from P other than $\text{vert}(\bar{s})$.

For the two dimensional case, an equivalent characterization of Delaunay is that of all

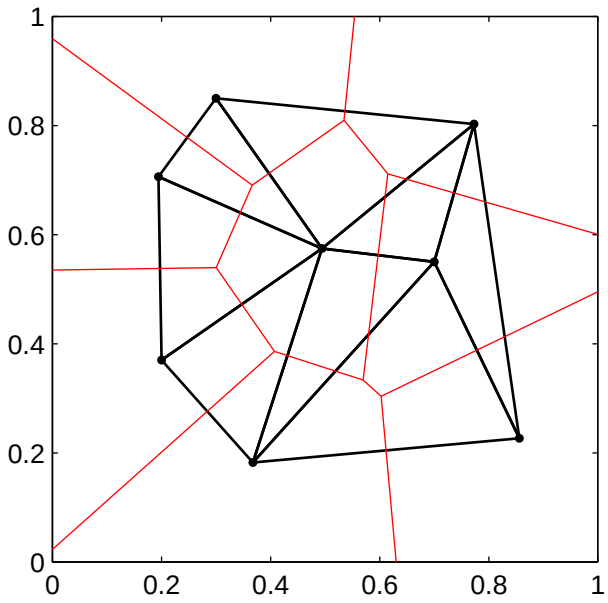

Figure 2.9: The Delaunay triangulation and its dual Voronoi tessellation. The Delaunay triangulation is shown in thick, dark lines and the dual Voronoi tessellation is shown in thin, light lines for this set of eight points in $\mathbb{R}^2$

possible triangulations, it is the triangulation for which the smallest angle of all the angles in all the triangles is as large as possible. There are also other equivalent characterizations, for example [Raj94] shows Delaunay as a solution to a combinatorial optimization problem.

The Delaunay triangulation is the dual of the Voronoi tessellation (also known as the "Voronoi diagram," and "Dirichlet" or "nearest neighbor" tessellation) [Bro79, Bow81, Wat81]. The Voronoi tessellation divides $\mathbb{R}^d$ into polygonal cells. The Voronoi tessellation has one cell for each vertex $\mathbf{p}_i \in P$ and the cell is the set of all points that are closer to $\mathbf{p}_i$ than to any other vertex in P. The Voronoi tessellation can be constructed from the Delaunay triangulation and vice versa. An example of a planar Delaunay triangulation and its dual Voronoi tessellation is shown in Figure 2.9.

The Delaunay triangulation is unique if the vertices are in general position. In this case, general position means that no $d+2$ points lie on a common sphere or on a common hyperplane.

2.3.2 Delaunay Triangulation via Lifting to Paraboloid

In [Bro79], Brown shows that the Voronoi tessellation, and hence the Delaunay triangulation, of a set of points in $\mathbb{R}^d$ may be computed by projecting the points in a particular way to $\mathbb{R}^{d+1}$, computing the convex hull of the projected points and then projecting the facets of the convex hull back down to $\mathbb{R}^d$. Since there are efficient algorithms and codes for

computing the convex hull in general dimension, projection provides an effective method for computing the Delaunay triangulation in higher dimensions. The projection employed by Brown is an inversion, but the modern preference is to project to a paraboloid. The algorithm for computing the Delaunay triangulation of a set of points by projecting to the paraboloid is:

Let $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$, $\mathbf{p}_i \in \mathbb{R}^d$ be in general position.

1. Project onto the paraboloid: $\mathbf{p}_i \longrightarrow \tilde{\mathbf{p}}_i = \left[\mathbf{p}_i{}^{\mathrm{T}} \quad \|\mathbf{p}_i\|^2\right]^{\mathrm{T}}$
2. Compute the convex hull of the projected points (i.e. the facets of the convex hull with outward pointing normal)
3. $\mathbf{p}_{i_1}, \mathbf{p}_{i_2}, \ldots, \mathbf{p}_{i_{d+1}}$ is a d-simplex of the Delaunay triangulation of P if and only if $\tilde{\mathbf{p}}_{i_1}, \tilde{\mathbf{p}}_{i_2}, \ldots, \tilde{\mathbf{p}}_{i_{d+1}}$ are the vertices of a facet of the convex hull of the projected points with outward normal vector pointed downward (negative in the $n+1$ coordinate).

A proof of correctness for this algorithm is supplied by Claim A.3 in the appendix. Projecting to the paraboloid will appear again in the discussion of topological flipping, the generalization of edge flipping.

2.3.3 Optimality of the Delaunay Triangulation

The Delaunay triangulation of a set of points can be interpreted as a combinatorial optimization problem. One of the first observed properties of the planar Delaunay triangulation is that of all triangulations of the convex hull of the point set, the Delaunay triangulation maximizes the minimum angle of any triangle (2-simplex) in the triangulation [Law77]. In this sense, the planar Delaunay triangulation can be considered the triangulation that maximizes the minimum angle. The following subsection shows how a lexicographic order can be placed on planar triangulations, and local flipping can be applied to reach the Delaunay triangulation [Law77]. The local flipping algorithm motivated the algorithm for computing a two dimensional data dependent triangulation [DLR90b] (presented in Section 3.2) as well as the heuristically guided retriangulation for MINVAR (Section 4.3).

The generalization of this optimality criterion to $\mathbb{R}^d$ is stated in terms of the maximum min-containment radius rather than maximum minimum angle. The min-containment radius is the radius of the smallest sphere that contains a given d-simplex. The maximum min-containment radius for a triangulation is the greatest min-containment radius for all d-simplices in that triangulation. For a given set of points, the Delaunay triangulation has the smallest maximum min-containment radius among all other triangulations [Raj91]. Unfortunately, this generalized property does not give rise to a lexicographic ordering of d-dimensional triangulations.

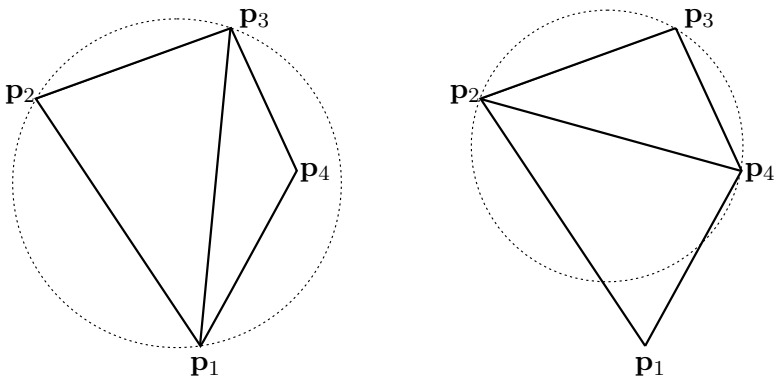

Figure 2.10: The locally Delaunay criterion. *Left*, two triangles $\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3$ and $\mathbf{p}_1\mathbf{p}_3\mathbf{p}_4$ share the edge $\mathbf{p}_1\mathbf{p}_3$. The circumcircle for $\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3$ is shown. Since $\mathbf{p}_4$ lies within the circumcircle, the edge $\mathbf{p}_1\mathbf{p}_3$ is not locally Delaunay, but the edge is flippable. *Right*, the edge $\mathbf{p}_1\mathbf{p}_3$ has been flipped to the edge $\mathbf{p}_2\mathbf{p}_4$. The circumcircle of triangle $\mathbf{p}_2\mathbf{p}_3\mathbf{p}_4$ does not contain $\mathbf{p}_1$, so the edge is locally Delaunay.

2D Delaunay Triangulation by Local Flipping

Delaunay triangulations have been studied in great detail in two dimensions for several decades. See [dBvKOS97, PS85], for example. Lawson found that the planar Delaunay triangulation could be constructed from an arbitrary triangulation through edge flipping [Law77, dBvKOS97]. A triangulation of a set of n points in $\mathbb{R}^2$ has a fixed number of triangles given by $2n - 2 - k$, where k is the number of points on the convex hull. A triangulation can be labeled with an angle vector with $3(2n - 2 - k)$ entries, listing the angles of all the triangles in the triangulation, ordered from smallest angle to largest. A lexicographic ordering of these angle vectors gives an ordering to the triangulations of the point set, with the greatest triangulation corresponding to the Delaunay triangulation. (Since, as described above, the Delaunay triangulation has the largest minimum angle.)

Consider an edge of a planar triangulation with endpoints $\mathbf{p}_i, \mathbf{p}_j$ shared by two triangles, one with vertices $\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k$ and one with vertices $\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_l$. The edge is said to be *locally Delaunay* (called the circle criterion by Lawson) if the circumcircle of $\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k$ does not contain $\mathbf{p}_l$. The locally Delaunay criterion is illustrated in Figure 2.10. An edge which is not locally Delaunay can always be flipped. Flipping from a non- locally Delaunay edge to a locally Delaunay edge yields a triangulation that is greater in the lexicographical order.

Given an initial triangulation of the point set, the following procedure generates the Delaunay triangulation:

1. While there remains an edge which is not locally Delaunay
   Flip edge

A proof that this procedure terminates and gives the Delaunay triangulation is provided in [Law77] and [dBvKOS97]. This procedure motivates the data dependent triangulations for

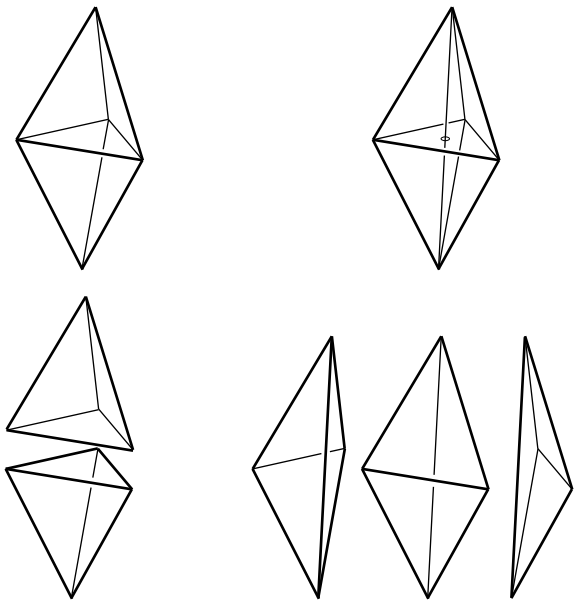

Figure 2.11: An example of topological flipping in $\mathbb{R}^3$. These five points in $\mathbb{R}^3$ have two possible triangulations. The triangulation on the left consists of two tetrahedra, while the triangulation on the right consists of three tetrahedra.

interpolation discussed in Chapter 3, and for the heuristically guided local retriangulation performed in the MINVAR algorithm. Unfortunately, this procedure does not generalize to higher dimensions in a straightforward fashion.

2.3.4 Local Topological Flipping in Higher Dimensions

In the two-dimensional case, one can move between triangulations of the set of points by "edge flipping," illustrated in Figure 2.8. In edge-flipping two triangles are removed and replaced by two other triangles without affecting the triangulation outside of the convex hull of the four points. Local changes to the triangulation can be used to find the Delaunay triangulation of a set of points given an initial triangulation, as presented above. Local changes in triangulation can also be used to improve interpolation of input-output data [DLR90b, DLR90a], discussed in section 3.2.

Local changes analogous to edge flipping exist for higher dimensions, but the changes are more complicated. In two dimensions, an edge flip replaces two triangles with two other triangles, but in higher dimension the number of d-simplices before and after the "flip" may change. Lawson first described local flipping for d-dimensional triangulations in [Law86]. Lawson shows that a set of $d+2$ points in $\mathbb{R}^d$ has at most two triangulations. Lawson uses "signature sets" to examine these triangulations.

Theorem (Theorem 1 from [Law86]). *Let P be a set of $d+2$ points $\mathbf{p}_1, \ldots, \mathbf{p}_{d+2}$ in $\mathbb{R}^d$*

not lying entirely in any hyperplane. There is a partitioning of the index set $1, 1, \ldots, d+2$ *into three sets* σ_0, σ_1 *and* σ_2 *and a set of numbers* c_i *satisfying*

$$\sum_{i \in \sigma_1} c_i \mathbf{p}_i = \sum_{i \in \sigma_2} c_i \mathbf{p}_i,$$

$$\sum_{i \in \sigma_1} c_i = \sum_{i \in \sigma_2} c_i = 1,$$

$$c_i = 0, \quad i \in \sigma_0,$$

$$c_i > 0, \quad i \in \sigma_1 \cup \sigma_2$$

The numbers c_i *are uniquely determined by* P*. The sets* σ_0*,* σ_1*,* σ_2 *are also unique with the understanding that the labeling of* σ_1 *and* σ_2 *could be arbitrarily interchanged.*

The sets σ_0, σ_1, and σ_2 are the signature sets of the $d+2$ points in P. Triangulations of P are given by the signature sets as follows:

Theorem (Theorem 2 from [Law86]). *Let* P *be as above. There are at most two distinct triangulations of the convex hull of* P*, namely* $\mathcal{T}(P, \mathcal{S}_1)$ *and* $\mathcal{T}(P, \mathcal{S}_2)$*, where the abstract* d*-simplices of* $\mathcal{S}_i$ *are given by* $\{1, \ldots, d+2\} \setminus \{j\}$ *for* $j \in \sigma_i$

When there are two triangulations of the $d+2$ points, one may switch between these two triangulations without affecting the convex hull. If the triangulation of these vertices given by σ_1 is a subcomplex of a larger d-dimensional triangulation in $\mathbb{R}^d$, then we can locally substitute the triangulation given σ_2 without affecting the rest of the triangulation. This is called a "topological flip." It is more complicated than standard edge-flipping, since a topological flip can change the number of d-simplices in the triangulation. Figure 2.11 shows two different triangulations for 5 generic points in $\mathbb{R}^3$, in which case one triangulation has two 3-simplices and the other has three 3-simplices. In [ES96], Edelsbrunner notes results of Lawson can be interpreted as a result of Radon's theorem from classical convex geometry [Web94, Theorem 2.2.5]. Conditions for finding flippable subcomplexes of a triangulation will be presented later in this section.

Further from Lawson [Law86], one can enumerate all possible significantly different configurations of $d+2$ points by finding possible ways of assigning values to $\text{card}(\sigma_0)$, $\text{card}(\sigma_1)$, and $\text{card}(\sigma_2)$ satisfying

$$\text{card}(\sigma_2) \geq \text{card}(\sigma_1) \geq 1,$$

$$\text{card}(\sigma_2) \geq 2,$$

$$\text{card}(\sigma_0) \geq 0,$$

$$\text{card}(\sigma_0) + \text{card}(\sigma_1) + \text{card}(\sigma_2) = d + 2.$$

Table 2.1 lists the possible assignments of signature set cardinalities for $d = 2, 3, 4$. Assignments with $\text{card}(\sigma_1) = 1$ are not admissible triangulations, since they represent a single

$d=2$			$d=3$			$d=4$		
$\text{card}(\sigma_0)$	$\text{card}(\sigma_1)$	$\text{card}(\sigma_2)$	$\text{card}(\sigma_0)$	$\text{card}(\sigma_1)$	$\text{card}(\sigma_2)$	$\text{card}(\sigma_0)$	$\text{card}(\sigma_1)$	$\text{card}(\sigma_2)$
1	1	2	2	1	2	3	1	2
0	2	2	1	2	2	2	2	2
0	1	3	1	1	3	2	1	3
			0	2	3	1	2	3
			0	1	4	1	1	4
						0	3	3
						0	2	4
						0	1	5

Table 2.1: Enumeration of the cardinalities of the signature sets for significantly different configurations of $d+2$ points in $\mathbb{R}^d$ (From [Law86])

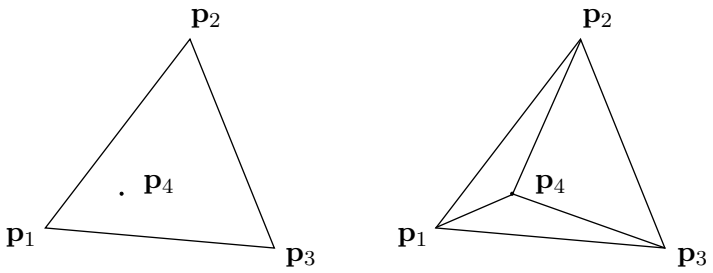

Figure 2.12: Inserting a point in a 2-dimensional triangulation via topological flipping. A configuration of four points in $\mathbb{R}^2$ with signature sets $\sigma_0 = \emptyset$, $\sigma_1 = \{4\}$ and $\sigma_2 = \{1,2,3\}$, with triangulations corresponding to σ_1 on the left and σ_2 on the right. The triangulation corresponding to σ_1 is inadmissible since $\mathbf{p}_4 \in \bar{s} = \text{conv}(\{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\})$ but $\mathbf{p}_4$ is not a vertex of $\bar{s}$. Even though this triangulation is not admissible, flipping can be used to add a vertex to (flip σ_1 to σ_2) or remove a vertex from (flip σ_2 to σ_1) a triangulation.

d-simplex in which the $(d+2)^{th}$ point lies inside. Figure 2.12 gives an example of this case. Flipping between the inadmissible and admissible triangulation can be used to add or remove a vertex from a triangulation.

Configurations where $\text{card}(\sigma_0) > 0$ are degenerate in the sense that $r+2$ of the points lie in an r-dimensional affine subspace for some $r < d$ (i.e. the points are not in general position). In this case, the facets of the convex hull of the points may change without changing the convex hull itself, as illustrated in Figure 2.13. If such a configuration occurs as a subcomplex of a larger triangulation, then the subcomplex cannot be flipped independently of simplices outside the subcomplex that share the degenerate faces.

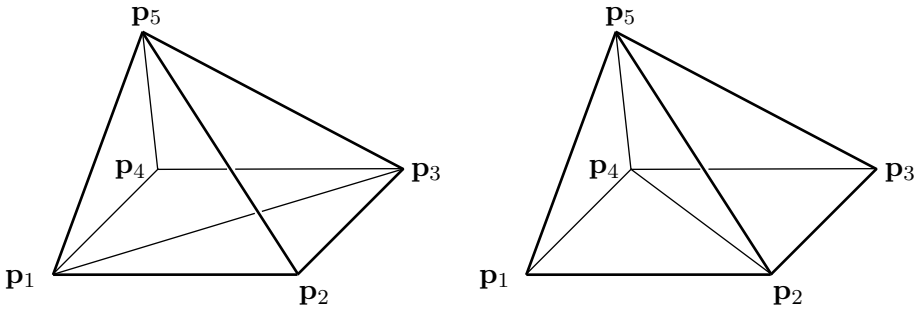

Figure 2.13: A degenerate flip in $\mathbb{R}^3$. A configuration of five points in $\mathbb{R}^3$ with $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4$ coplanar. The signature sets are $\sigma_0 = \{5\}$, $\sigma_1 = \{2, 4\}$ and $\sigma_2 = \{1, 3\}$, with triangulations corresponding to σ_1 on the left and σ_2 on the right. Notice that the facets on the convex hull are different in the two triangulations, though the convex hull itself does not change.

Local Triangulations Via Lifting to Paraboloid

In [ES96], Edelsbrunner shows that the two triangulations of a set of $d + 2$ points in $\mathbb{R}^d$ that do not all lie in a hyperplane may be computed by lifting the points to the paraboloid and computing the convex hull, as in computing the Delaunay triangulation in Section 2.3.2. The points are lifted to the paraboloid by $\mathbf{p}_i \rightarrow \tilde{\mathbf{p}}_i = [\mathbf{p}_i{}^{\mathrm{T}} \ \ \|\mathbf{p}_i\|^2]^{\mathrm{T}}$ for $i = 1, \ldots, d+2$. Since the $\mathbf{p}_i$'s do not all lie in a hyperplane $\mathbb{R}^d$, the lifted points $\tilde{\mathbf{p}}_i \in \mathbb{R}^{d+1}$ are affinely independent. Since the $d + 2$ points are affinely independent points in $\mathbb{R}^{d+1}$, it follows that $\bar{s} = \mathrm{conv}(\{\tilde{\mathbf{p}}_1, \ldots, \tilde{\mathbf{p}}_{d+2}\})$ is a $(d + 1)$-simplex and the convex hull of any subset of $d+1$ vertices, $\{\tilde{\mathbf{p}}_1, \ldots, \tilde{\mathbf{p}}_{d+2}\} \setminus \{\tilde{\mathbf{p}}_j\}$, is a facet of $\bar{s}$. The facet's (outward pointing) normal vector points away from the halfspace containing $\tilde{\mathbf{p}}_j$. As in Section 2.3.2, facets with downward pointing normal vectors (negative in the $(d+1)^{th}$ coordinate) correspond to the d-simplices of the Delaunay triangulation of $\mathbf{p}_1, \ldots, \mathbf{p}_{d+2}$. Facets with upward pointing normal vectors (positive in the $(d + 1)^{th}$ coordinate) correspond to the d-simplices of the second triangulation. (Note that for more than $d+2$ points, the facets with upward pointing normal vectors would not correspond to a valid triangulation.) Facets with normals that point neither upward nor downward ($(d + 1)^{th}$ coordinate is zero) are degenerate and do not belong to either triangulation, corresponding to the signature set σ_0. This occurs when $r + 2$ of the points $\mathbf{p}_i$ lie in an affine subspace of dimension r, for some $r < d$. This is a thin set in the space of point configurations, but unfortunately designs often put large numbers of points in the same hyperplane, for example on the domain boundary.

Flippability of a Subcomplex

The discussion above applies to configurations of $d + 2$ points in $\mathbb{R}^d$, but the objective is to use these results to change the combinatorial structure $\mathcal{S}_1$ for some d-dimensional triangulation in $\mathbb{R}^d$, $\mathcal{T}(P, \mathcal{S}_1)$ with $\mathrm{card}(P) > d + 2$ to get a new combinatorial structure

$\mathcal{S}_2$ such that $\mathcal{T}(P, \mathcal{S}_2)$, where $\mathcal{S}_2$ is the same as $\mathcal{S}_1$ except for a local change of the form above. This requires a method of finding a subcomplex of $\mathcal{T}(P, \mathcal{S}_1)$ that has the proper configuration to be flipped. Edelsbrunner proposed the following definition for *flippability* when the points of P are in general position, i.e. no $r+2$ points lie in an affine subspace of dimension r for some $r < d$:

Let s be a $(d-1)$-simplex (i.e. a facet) of a triangulation $\mathcal{T}(P, \mathcal{S})$, and let $\bar{s}_1$ and $\bar{s}_2$ be the incident d-simplices, if they exist. Let $P' = \text{vert}(\bar{s}_1) \cup \text{vert}(\bar{s}_2)$ and let $\mathcal{S}'$ be the collection $\mathcal{S}' = \{\alpha \in \mathcal{S} | \, \mathbf{p}_i \in P' \text{ for all } i \in \alpha\}$. Then $\mathcal{T}(P', \mathcal{S}')$ is the triangulation of P' induced by $\mathcal{S}$. The subcomplex $\mathcal{T}(P', \mathcal{S}')$ is *flippable* if $\text{conv}(P') = |\mathcal{T}(P', \mathcal{S}')|$. If the subcomplex is flippable, one also says that s is flippable. [ES96]

Given a triangulation $\mathcal{T}(P, \mathcal{S}_1)$, if the points of P are in general position and one finds a flippable subcomplex $\mathcal{T}(P', \mathcal{S}')$, then the alternate triangulation of P' can be computed by lifting to the paraboloid as discussed above, giving the new subcomplex $\mathcal{T}(P', \mathcal{S}'')$. The new triangulation is then $\mathcal{T}(P, \mathcal{S}_2)$, where $\mathcal{S}_2 = (\mathcal{S} \setminus \mathcal{S}') \cup \mathcal{S}''$.

2.3.5 Regular Triangulation via Incremental Topological Flipping

Any 2-dimensional triangulation can be transformed into a Delaunay triangulation through the edge-flipping algorithm as presented in Section 2.3.3. For a 3-dimensional triangulation, Joe [Joe89] shows that an analogous algorithm can get stuck. That is, there will be simplices that do not satisfy the local Delaunay property but are not part of a flippable subcomplex. A straightforward flipping algorithm does not work. In [Raj91], Rajan shows that if a point is inserted in a Delaunay triangulation of general dimension, then there is always a sequence of flips that converts this new triangulation to a Delaunay triangulation, and presents an algorithm to incrementally build a Delaunay triangulation through point insertion and local flipping.

Regular triangulations [Lee91] are a generalization of Delaunay, in which each point has a weight associated with it. The weights control the triangulation, and when all weights are the same, the regular triangulation is the Delaunay triangulation. For a set of points in general position in $\mathbb{R}^d$, a set of weights gives rise to a unique regular triangulation.

In [ES96], Edelsbrunner presents an algorithm that computes the regular triangulation of a set of points in general position in $\mathbb{R}^d$ using incremental topological flipping. When inserting a new point $\mathbf{p}_i$ into the triangulation, it is shown that only non-regular facets in $\text{lk}\, \mathbf{p}_i$, the link of the inserted point, must be flipped in order to transform the triangulation back to a regular triangulation. Edelsbrunner's algorithm couples this fact with a clever method to determine the d-simplex in which the inserted vertex lies, providing overall an $O(n \log n + n^{\lceil d/2 \rceil})$ algorithm. The method to locate the inserted point uses a directed acyclic graph for locating the d-simplex to represent the successive triangulations generated as vertices are inserted. A variant of this algorithm is implemented as part of the MIN-

VAR algorithm in order to generate initial triangulations and perform heuristically guided retriangulation.

2.3.6 Practical Considerations

The algorithms in this section assume that the points to be triangulated are in general position. That is, no $d+2$ points lie on a common sphere or on a common hyperplane. When the input points are not in general position, they are said to be degenerate. In the input space of n points in $\mathbb{R}^d$, i.e. $\mathbb{R}^{nd}$, the set of degenerate configurations has measure zero. If configurations were chosen randomly, then degenerate configurations would be vanishingly rare, but often the set of points to be triangulated is not chosen randomly. For example, in the following chapter on PL functions, a triangulation represents the domain of a PL function. Often (e.g. due to design of experiments) the domain is a hypercube with multiple vertices on each face, which is a degenerate configuration and requires special care in constructing a triangulation. As with many geometric algorithms, much of the difficulty in constructing triangulations is addressing the special cases caused by degenerate input. There are many codes for generating triangulations in two and three dimensions that deal with degeneracies, driven by interest in computer graphics and finite element modeling. In contrast there is very little that reliably deals with degenerate point sets in general dimension.

Consider a degenerate set of n points in which $d+2$ (or more) points lie on a common hypersphere. In this case, the Delaunay triangulation, which would be unique if the points were in general position, might not be unique. When computing the Delaunay triangulation by lifting to the paraboloid (Section 2.3.2), this type of degeneracy appears as a face of the convex hull that is a polytope rather than a simplex. This polytope can be partitioned into simplices in several ways, resulting in several different triangulations that are equivalently Delaunay. One of these triangulations must be chosen. This choice is arbitrary since each of the resulting triangulations is equivalently Delaunay.

Consider a degenerate set of n points in which $r+2$ points (or more) lie on an r dimensional affine subspace, $r \leq d$. This type of degeneracy is more difficult to handle since it can lead to a combinatorial structure that does not yield a valid triangulation with the input points. Consider a flippable subcomplex of $d+2$ points, for which $r+2$, $r < d$, of the points lie in an r-dimensional affine subspace. In this case, $\text{card}(\sigma_0) > 0$, where σ_0 is the signature set described in Section 2.3.4. When computing the two possible triangulations of the $d+2$ points by lifting to the paraboloid, the degeneracy causes facet(s) whose normal vector is orthogonal to the lifting axis. That is, the facet's normal vector points neither upward nor downward. When this type of degenerate subcomplex occurs, it cannot be flipped without simultaneously flipping adjoining degenerate subcomplexes. This form of degeneracy is difficult to detect systematically due to finite precision arithmetic, especially in the case where

multiple subcomplexes must be flipped simultaneously. A facet's normal vector[6] is a continuous object. The normal vector is classified as pointing upward, downward, or orthogonal, which is a combinatorial classification, and the classification of the facets determines two possible triangulations. Due to finite precision arithmetic, a normal vector will generally never be perfectly orthogonal, even if the points are degenerate. Misclassification leads to very different combinatorial structures. If points which are degenerate appear numerically to be nondegenerate, or alternatively if nondegenerate points appear degenerate, then an improper combinatorial structure may be chosen that does not yield a valid triangulation. In the QHULL code [BDH96] (the algorithm is designed for data in general position), the effects of misdiagnosed degeneracies are often observed as improper d-simplices with zero volume. If the improper simplices are on the boundary, then they can be pruned without damaging the triangulation, but if the improper simplices are in the interior (not on the boundary) of the triangulation, then pruning breaks the combinatorial structure of the triangulation.

There are several ways to address degeneracies. One method is to perturb the input set to gain a set of points in general position. The QHULL code optionally performs such a perturbation, which they call "joggling," by adding small random floating point values to the input points. The size of the perturbation is determined heuristically. Simulation of Simplicity (SoS) [EM90] is another perturbation based technique, developed by Edelsbrunner and Mücke, which provides a general method for addressing degenerate cases in geometric algorithms. SoS represents perturbations symbolically, computing the solution for a set of nondegenerate points infinitesimally close to the degenerate point set. This requires exact arithmetic, which is considerably more computationally intensive than machine native finite precision arithmetic, but removes the need for heuristics to determine the proper size for perturbation. Perturbation techniques in triangulation algorithms yield a combinatorial structure that forms a valid triangulation with the perturbed points, but that might not form a valid triangulation with the original set of points. For example, some of the simplices may be improper, i.e. have zero volume. Postprocessing is needed in this case, but is not currently well understood in general dimension.

The triangulation code developed for use in the MINVAR algorithm identifies degeneracies numerically and treats these cases explicitly. The code works robustly in $\mathbb{R}^3$, and addresses some, but not all, degeneracies in higher dimensions. Like many problems in geometry, the most significant difficulties in triangulation arise in from the degenerate cases. Dealing with degeneracies in triangulations is a topic of ongoing interest, both in the present work as well as in the computation geometry more generally.

[6]Other quantities can be used to diagnose the degeneracy, but the normal vector is convenient for the present discussion.

2.4 Validating an Embedding of an Abstract Simplicial Complex

In checking invertibility of PL functions, the situation arises where one wishes to test whether a set of vertices forms a triangulation when coupled with a particular abstract simplicial complex, which we call an embedding of the abstract simplicial complex. Let P and $\mathcal{S}$ be given such that $\mathcal{T}(P, \mathcal{S})$ is a triangulation, i.e. $\mathcal{T}(P, \mathcal{S})$ is an embedding of $\mathcal{S}$. Given another set of points Q with the same number of points as P, can we validate that $\mathcal{T}(Q, \mathcal{S})$ is an embedding of $\mathcal{S}$? This is a complementary problem to triangulating a set of points, discussed in the previous section. In generating a triangulation of a set of points, a fixed set of vertices P is provided and one wishes to determine an abstract simplicial complex $\mathcal{S}$ such that $\mathcal{T}(P, \mathcal{S})$ is a triangulation. To validate an embedding of an abstract simplicial complex, an abstract simplicial complex (which is know to form a triangulation with *some* unspecified set of vertices) and a set of vertices are given, and it must be determined whether these together give a triangulation.

The naive approach to validating an embedding is to directly check the conditions of Claim 2.13, that is, check that the appropriate sets of vertices are affinely independent and that there are no improper intersections of the simplices in $\mathcal{T}(Q, \mathcal{S})$. It is given that $\mathcal{T}(P, \mathcal{S})$ is a triangulation, for some set P, so it is unnecessary to check whether the underlying space of $\mathcal{T}(Q, \mathcal{S})$ is a manifold with boundary. $\mathcal{T}(Q, \mathcal{S})$ will be a good triangulation so long as it is a good geometric simplicial complex. That is, it is sufficient to check that $\mathcal{T}(Q, \mathcal{S})$ is a collection of well formed simplices and that there are no improper intersections between the simplices. Furthermore, since $\mathcal{T}(P, \mathcal{S})$ is a triangulation, it suffices to check only for improper intersections among the d-simplices of $\mathcal{T}(Q, \mathcal{S})$. Exhaustively checking pairs of d-simplices for improper intersections is quadratic in the number of d-simplices. Improvements could be made, for example by arranging the d-simplices in a variant of a $k-d$ tree in order to avoid checking for intersections of d-simplices that are not nearby.

CHAPTER 3

Piecewise Linear Functions: Representation and Computation

In this dissertation, PL is used to indicate a *continuous* piecewise linear function[1]. This chapter presents a parameterization of PL functions that has a nice geometric interpretation, especially helpful for identifying properties such as invertibility. This representation of PL functions has been used in algebraic topology for decades [Spa66], but has not been widely used in approximation or other areas interested in computation. In dimensions greater than one, PL functions have a nontrivial combinatorial component. The parameterization explicitly separates the combinatorial and continuous parameters of the PL functions.

This chapter is divided into four sections. Section 3.1 presents the parameterization of PL functions, and proves some basic properties about them. One important result is Claim 3.1, which notes a relationship between the Jacobian of a PL function over domain simplices that share a common face. This relationship is used in the generalization of the "jump in normal derivative" criterion for heuristic retriangulation (Section 4.3) as well as in the proof of local convergence of the MINVAR algorithm. Claim 3.3 shows that a PL function is continuous in its continuous parameters, an important observation necessary for the proof of Corollary 5.1. Also, the proof of Claim 3.3 uses the same construction applied in the proof of Lemma 5.1, critical for proving the main result of Chapter 5. Section 3.2 covers interpolation of scattered data with PL functions, and includes a discussion of previous work on 2-dimensional data dependent triangulations for interpolation, which heavily influenced the heuristically guided retriangulation for MINVAR (Section 4.3). Section 3.3 shows how to compute the least squares PL approximation given a fixed domain triangulation, a standard problem from the spline literature. This result is included in part for the sake of comparison and in part to be used as a possible postprocessing step in the MINVAR algorithm. In contrast, the MINVAR algorithm adapts both the domain triangulation in addition to the

[1] "Piecewise linear" is widely used to describe a function that is affine on each cell of a partition of the domain partition. A piecewise linear function *is* locally linear over each cell, so while it may be more descriptive to use the term "piecewise affine" to describe such a function, this dissertation follows convention and uses the term piecewise linear.

other parameters. The chapter concludes with Section 3.4 covering how to check if a PL function is invertible and how to invert the function in closed form, which are major benefits of PL functions.

The material in this chapter is largely a reinterpretation of standard results in terms of the geometrically influenced parameterization of PL functions presented in Section 3.1. The parameterization is particularly nice for understanding the closed form invertibility properties of PL functions, presented in Section 3.4. Claims 3.1 and 3.3 in Section 3.1 are straightforward properties about PL functions, though the author is uncertain whether they previously appeared in the literature.

3.1 Parameterization of Piecewise Linear Functions

An affine function $f : \mathbb{R}^d \to \mathbb{R}^c$ is generally written as $f(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}$. It might seem reasonable to represent a PL function as some partition of the domain with a matrix and vector $(\mathbf{A}_i, \mathbf{b}_i)$ for each cell of the partition, however the requirement of continuity places constraints on the $(\mathbf{A}_i, \mathbf{b}_i)$ pairs as well as on the partition of the domain. A much more parsimonious parameterization is possible.

A PL function[2] $f_{\mathcal{P}} : D \to C$, $D \subset \mathbb{R}^d$, $C \subset \mathbb{R}^c$, is parameterized by a triplet $\mathcal{P} = (P, Q, \mathcal{S})$. P is an indexed set of n points in the domain and Q is an indexed set of n points in the codomain,

$$P = \Big\{\mathbf{p}_1, \quad \mathbf{p}_2, \quad \ldots, \quad \mathbf{p}_n\Big\} \qquad Q = \Big\{\mathbf{q}_1, \quad \mathbf{q}_2, \quad \ldots, \quad \mathbf{q}_n\Big\},$$

and $\mathcal{S}$ is an abstract simplicial complex with $\mathrm{vert}(\mathcal{S}) = \{1, \ldots, n\}$, such that $\mathcal{T}(P, \mathcal{S})$ is a triangulation with $|\mathcal{T}(P, \mathcal{S})| = D$. The triplet $\mathcal{P} = (P, Q, \mathcal{S})$ defines a continuous piecewise linear function $f_{\mathcal{P}}$ such that $f_{\mathcal{P}}(\mathbf{p}_i) = \mathbf{q}_i$, and for any d-simplex $\bar{s}_i \in \mathcal{T}(P, \mathcal{S})$, $f_{\mathcal{P}}(\mathbf{x})$ is affine on $\bar{s}$. For an abstract d-simplex in $\mathcal{S}$, $\{i_1, \ldots, i_{d+1}\}$, there is a corresponding d-simplex $\bar{s}_i \in \mathcal{T}(P, \mathcal{S})$ with $\mathrm{vert}(\bar{s}_i) = \{\mathbf{p}_{i_1}, \mathbf{p}_{i_2}, \ldots, \mathbf{p}_{i_{d+1}}\}$. For $\mathbf{x} \in \bar{s}_i$, the PL function is given by

$$f_{\mathcal{P}}\big|_{\bar{s}_i}(\mathbf{x}) = \mathbf{C}_i \mathbf{D}_i{}^{-1} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}, \tag{3.1}$$

where

$$\mathbf{D}_i = \begin{bmatrix} \mathbf{p}_{i_1} & \mathbf{p}_{i_2} & \cdots & \mathbf{p}_{i_{d+1}} \\ 1 & 1 & & 1 \end{bmatrix}, \tag{3.2}$$

$$\mathbf{C}_i = \begin{bmatrix} \mathbf{q}_{i_1} & \mathbf{q}_{i_2} & \cdots & \mathbf{q}_{i_{d+1}} \end{bmatrix}. \tag{3.3}$$

By Claim 2.5 the rightmost two factors of Equation 3.1, $\mathbf{D}_i{}^{-1}[\mathbf{x}^{\mathrm{T}}\ 1]^{\mathrm{T}}$, give the barycentric coordinates of $\mathbf{x}$ with respect to the vertices of $\bar{s}_i$. $f_{\mathcal{P}}\big|_{\bar{s}_i}(\mathbf{x})$ is the corresponding affine combination of $\mathbf{q}_{i_1}, \ldots, \mathbf{q}_{i_{d+1}}$. It follows that the image of $\bar{s}_i$ under $f_{\mathcal{P}}$ is $\mathrm{conv}(\{\mathbf{q}_{i_1}, \mathbf{q}_{i_2}, \cdots, \mathbf{q}_{i_{d+1}}\})$,

[2]Though this dissertation predominantly concerns PL functions where $c = d$, this chapter will present a more general definition of PL functions.

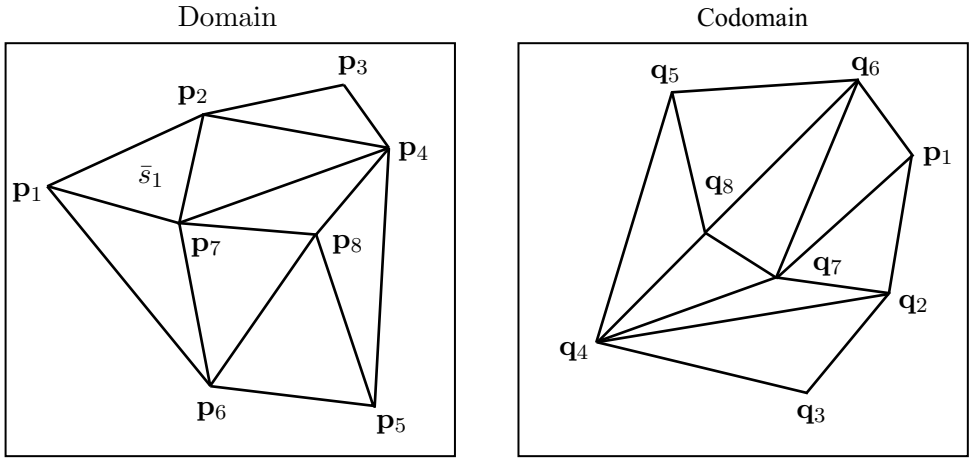

Figure 3.1: Visualization of a 2-dimensional PL function, $f_{\mathcal{P}}$. The domain triangulation $\mathcal{T}(P, \mathcal{S})$ is shown on the left. $f_{\mathcal{P}}$ is affine over each 2-simplex (triangle) in the domain, for example $f_{\mathcal{P}}$ restricted to $\bar{s}_1$ is given by

$$f_{\mathcal{P}}\big|_{\bar{s}_1}(\mathbf{x}) = \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 & \mathbf{q}_7 \end{bmatrix} \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \mathbf{p}_7 \\ 1 & 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}.$$

Since $\mathcal{T}(Q, \mathcal{S})$ shown on the right, is also a valid triangulation, the PL function is invertible. The function may be inverted by simply switching the domain and range.

which gives a hint at the structure that allows PL functions to be inverted in closed form, presented in Section 3.4. From this barycentric coordinate interpretation, clearly $f_{\mathcal{P}}\big|_{\bar{s}_i}(\mathbf{p}_{i_j}) = \mathbf{q}_{i_j}$. Another fact that falls out of the barycentric coordinate interpretation of Equation 3.1 is that only the correspondences between the $\mathbf{p}_{i_k}$'s and $\mathbf{q}_{i_k}$'s matter. $f_{\mathcal{P}}\big|_{\bar{s}_i}$ remains unaffected if the same column permutation is applied to $\mathbf{D}_i$ and $\mathbf{C}_i$.

Through algebraic manipulation of Equation 3.1 one can equivalently write

$$f_{\mathcal{P}}|_{\bar{s}_i}(\mathbf{x}) = \mathbf{A}_i\mathbf{x} + \mathbf{b}_i \tag{3.4}$$

or

$$f_{\mathcal{P}}|_{\bar{s}_i}(\mathbf{x}) = \mathbf{A}_i(\mathbf{x} - \mathbf{p}_{i_j}) + \mathbf{q}_{i_j}, \tag{3.5}$$

where $\mathbf{p}_{i_j} \in \text{vert}(\bar{s}_i)$, $\mathbf{q}_{i_j}$ is the corresponding codomain vertex, and

$$\begin{aligned} \mathbf{A}_i &= \mathbf{C}_i{\mathbf{D}_i}^{-1} \begin{bmatrix} \mathbf{I}_{d\times d} \\ \mathbf{0}_{1\times d} \end{bmatrix}, \\ \mathbf{b}_i &= \mathbf{C}_i{\mathbf{D}_i}^{-1} \begin{bmatrix} \mathbf{0}_{d\times 1} \\ 1 \end{bmatrix}. \end{aligned} \tag{3.6}$$

In some cases it is more convenient to use one representation rather than another, but Equation 3.1 makes the clearest connection to the underlying geometry.

We now verify that the functions $f_{\mathcal{P}}\big|_{\bar{s}_i}$ of this piecewise definition[3] paste together to give a well defined continuous piecewise linear function. For a given d-simplex $\bar{s}_i$, $f_{\mathcal{P}}\big|_{\bar{s}_i}$ is clearly affine, and hence continuous from the definition above. The d-simplices of $\mathcal{T}(P, \mathcal{S})$ form a cover of the domain D, intersecting each other along faces. By the pasting lemma (Theorem 7.3 of [Mun75]), in order for these functions $f_{\mathcal{P}}\big|_{\bar{s}_i}$ to paste together to form a continuous function, one only needs to check that $f_{\mathcal{P}}\big|_{\bar{s}_i}(\mathbf{x}) = f_{\mathcal{P}}\big|_{\bar{s}_j}(\mathbf{x})$ for $\mathbf{x} \in \bar{s}_i \cap \bar{s}_j$. This fact, as well as a closely related fact applied later in this chapter as well as in the convergence proof of Chapter 5, is established by the following claim. As a preliminary definition, the function g is called the *extension of* $f_{\mathcal{P}}\big|_{\bar{s}}$ *to* $\mathbb{R}^d$ if g is affine and $g(\mathbf{x}) = f_{\mathcal{P}}\big|_{\bar{s}}(\mathbf{x})$ for $\mathbf{x} \in \bar{s}$.

Claim 3.1. *Let $(P, Q, \mathcal{S})$ be the parameterization of piecewise linear function. Let $\bar{s}_i, \bar{s}_j \in \mathcal{T}(P, \mathcal{S})$ be such that $\bar{s}_i \cap \bar{s}_j \neq \emptyset$. Let $f_{\mathcal{P}}\big|_{\bar{s}_i}$ and $f_{\mathcal{P}}\big|_{\bar{s}_j}$ be defined as in Equation 3.1, and let f_i and f_j be their respective extensions to $\mathbb{R}^d$.*
i). Then $f_i(\mathbf{x}) = f_j(\mathbf{x})$, for $\mathbf{x} \in$ aff $(\bar{s}_i \cap \bar{s}_j)$
ii). Let f_i and f_j be given by $f_i(\mathbf{x}) = \mathbf{A}_i\mathbf{x} + \mathbf{b}_i$ and $f_j(\mathbf{x}) = \mathbf{A}_j\mathbf{x} + \mathbf{b}_j$. Let L be the linear subspace parallel to aff$(\bar{s}_i \cap \bar{s}_j)$. *Then $L \subseteq \mathcal{N}(\mathbf{A}_i - \mathbf{A}_j)$.*

Proof. The extensions f_i and f_j are also given by Equation 3.1, except that they hold over all of $\mathbb{R}^d$. Let k be the number of vertices that $\bar{s}_i$ and $\bar{s}_j$ share. One can find column permutations of $\mathbf{D}_i$, $\mathbf{C}_i$ and $\mathbf{D}_j$, $\mathbf{C}_j$ such that $\mathbf{D}_i$ and $\mathbf{D}_j$ have the same first k columns and $\mathbf{C}_i$ and $\mathbf{C}_j$ have the same first k columns. This permutation leaves f_i and f_j unchanged. Consider $\mathbf{x} \in$ aff$(\bar{s}_i \cap \bar{s}_j)$. Since barycentric coordinates are unique by Claim 2.4, the barycentric coordinates of $\mathbf{x}$ with respect to vert$(\bar{s}_i)$ and with respect to vert$(\bar{s}_j)$, given by $\mathbf{D}_i[\mathbf{x}^{\mathrm{T}}\ 1]^{\mathrm{T}}$ and $\mathbf{D}_j[\mathbf{x}^{\mathrm{T}}\ 1]^{\mathrm{T}}$ respectively, will be equal and only the first k coordinates will be nonzero, which correspond to vertices of $\bar{s}_i \cap \bar{s}_j$. It follows that $f_i(\mathbf{x}) = f_j(\mathbf{x})$.

Let $\mathbf{A}_i, \mathbf{b}_i$ and $\mathbf{A}_j, \mathbf{b}_j$ be such that $f_i(\mathbf{x}) = \mathbf{A}_i\mathbf{x} + \mathbf{b}_i$ and $f_j(\mathbf{x}) = \mathbf{A}_j\mathbf{x} + \mathbf{b}_j$. Let $\mathbf{x} \in$ aff$(\bar{s}_i \cap \bar{s}_j)$. Since *(i)* holds independently of the specific representation of f_i and f_j, it follows that $\mathbf{A}_i\mathbf{x} + \mathbf{b}_i = \mathbf{A}_j\mathbf{x} + \mathbf{b}_j$. Let $\mathbf{x}_0 \in$ aff$(\bar{s}_i \cap \bar{s}_j)$ and let $\mathbf{y}_0 = \mathbf{A}_i\mathbf{x}_0 + \mathbf{b}_i = \mathbf{A}_j\mathbf{x}_0 + \mathbf{b}_j$. Then $\mathbf{A}_i(\mathbf{x} - \mathbf{x}_0) + \mathbf{y}_0 = \mathbf{A}_j(\mathbf{x} - \mathbf{x}_0) + \mathbf{y}_0$, or equivalently $(\mathbf{A}_i - \mathbf{A}_j)(\mathbf{x} - \mathbf{x}_0) = \mathbf{0}$. From the definition of affine subspace, $\mathbf{x} - \mathbf{x}_0 \in L$, where L is the linear subspace parallel to aff$(\bar{s}_i \cap \bar{s}_j)$. Thus $L \subseteq \mathcal{N}(A_i - A_j)$. □

The parameters of a PL function fall naturally into two groups, the continuous parameters, P and Q, and the combinatorial parameters, $\mathcal{S}$. A PL function is continuous in its continuous parameters, which is a simple corollary of Claim 3.3, to be presented shortly. In order to set up this claim, consider two continuous piecewise linear functions, $f^1_{\mathcal{P}}$ parameterized by $\mathcal{P}_1 = (P_1, Q_1, \mathcal{S})$ and $f^2_{\mathcal{P}}$ parameterized by $\mathcal{P}_2 = (P_2, Q_2, \mathcal{S})$, such that $|\mathcal{T}(P_1, \mathcal{S})| = |\mathcal{T}(P_2, \mathcal{S})|$. A superscript denotes whether an entity corresponds to $f^1_{\mathcal{P}}$ or $f^2_{\mathcal{P}}$,

[3]Using the restriction notation $f_{\mathcal{P}}\big|_{\bar{s}_i}$ anticipates that the piecewise definition of Equation 3.1 does indeed give a continuous piecewise linear function. This abuse permits a more straightforward introduction of parameterized PL functions.

for example $P_1 = \{\mathbf{p}_1^1, \ldots, \mathbf{p}_n^1\}$, $Q_1 = \{\mathbf{q}_1^1, \ldots, \mathbf{q}_n^1\}$, $\bar{s}_i^1$ is the ith d-simplex of $\mathcal{T}(P_1, \mathcal{S})$, $f_{\mathcal{P}}^1\big|_{\bar{s}_1^1}(\mathbf{x}) = \mathbf{A}_i^1(\mathbf{x} - \mathbf{p}_{i_j}^1) + \mathbf{q}_{i_j}^1$, etc.

Before stating the result on continuity of PL in the vertices, it is useful to recall that in this chapter and throughout this dissertation, vector norms are the standard Euclidean norm and matrix norms are the induced two norm, unless otherwise noted. The L_∞ norm appears in the statement of the continuity result, and the following fact is useful for relating the vector 2-norm to the L_∞ norm.

Claim 3.2. *The L_∞ norm for the continuous function $f : U \to V$, $U \subset \mathbb{R}^d, V \subset \mathbb{R}^c$ is*

$$\|f\|_\infty = \operatorname*{ess\,sup}_{\mathbf{x} \in U} \|f(\mathbf{x})\|_\infty .$$

If

$$\sup_{\mathbf{x} \in U} \|f(\mathbf{x})\|_2 < \epsilon$$

then $\|f\|_\infty < \epsilon$

Proof. This fact follows from the fact that $\|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_2$ for $\mathbf{x} \in \mathbb{R}^d$. □

Claim 3.3 (Continuity in vertices). *Let $c > 0$. Let $0 < \epsilon < r_3$ be given, where r_3 is the dilation radius of $\mathcal{T}(P_2, \mathcal{S})$ given by Equation 2.21.*
If $\|\mathbf{p}_i^1 - \mathbf{p}_i^2\| < \epsilon$ and $\|\mathbf{q}_i^1 - \mathbf{q}_i^2\| < c\epsilon$ for $i = 1, \ldots, n$, then

$$\left\|f_{\mathcal{P}}^1 - f_{\mathcal{P}}^2\right\|_\infty < c'\epsilon,$$

where $c' = c'(\mathcal{P}_2, c)$ is given by Equation 3.7.

Proof. The d-simplices of $\mathcal{T}(P_1, \mathcal{S})$ are a cover for the domain, so $\left\|f_{\mathcal{P}}^1 - f_{\mathcal{P}}^2\right\|_\infty$ can be rewritten as the maximum of the infinity norm over the N d-simplices of $\mathcal{T}(P_1, \mathcal{S})$,

$$\left\|f_{\mathcal{P}}^1 - f_{\mathcal{P}}^2\right\|_\infty = \max_{i=1,\ldots,N} \left\|(f_{\mathcal{P}}^1 - f_{\mathcal{P}}^2)|_{\bar{s}_i^1}\right\|_\infty .$$

Let $f_{\mathcal{P}}^2\big|_{\bar{s}_i^2}$ be given by $f_{\mathcal{P}}^2\big|_{\bar{s}_i^2}(\mathbf{x}) = \mathbf{A}_i^2(\mathbf{x} - \mathbf{p}_{i_j}^2) + \mathbf{q}_{i_j}^2$ for $\mathbf{p}_{i_j}^2 \in \operatorname{vert}(\bar{s}_i^2)$. Let φ_i be the extension of $f_{\mathcal{P}}^2\big|_{\bar{s}_i^2}$ to $\mathbb{R}^d$, $\varphi_i(\mathbf{x}) = \mathbf{A}_i^2(\mathbf{x} - \mathbf{p}_{i_j}^2) + \mathbf{q}_{i_j}^2$. Let $\psi_i(\mathbf{x}) = f_{\mathcal{P}}^2(\mathbf{x}) - \varphi_i(\mathbf{x})$. Then φ is an affine function and ψ_i is a piecewise linear function with $\psi_i\big|_{\bar{s}_i^2} \equiv \mathbf{0}$. Then,

$$\begin{aligned} \left\|(f_{\mathcal{P}}^1 - f_{\mathcal{P}}^2)|_{\bar{s}_i^1}\right\|_\infty &= \left\|(f_{\mathcal{P}}^1 - \varphi_i - \psi_i)|_{\bar{s}_i^1}\right\|_\infty \\ &\leq \left\|(f_{\mathcal{P}}^1 - \varphi_i)|_{\bar{s}_i^1}\right\|_\infty + \left\|\psi_i|_{\bar{s}_i^1}\right\|_\infty \end{aligned}$$

These terms will be individually bounded. The function $f_{\mathcal{P}}^1 - \varphi_i$ is affine on $\bar{s}_i^1$, so then the function $\left\|f_{\mathcal{P}}^1(\mathbf{x}) - \varphi_i(\mathbf{x})\right\|$ is convex on $\bar{s}_i^1$. Moreover $\bar{s}_i^1$ is a compact convex set, so

$\left\|f^1_{\mathcal{P}}(\mathbf{x}) - \varphi_i(\mathbf{x})\right\|$ will take its maximum on $\bar{s}^1_i$ at a vertex of $\bar{s}^1_i$. Thus,

$$\begin{aligned}
\left\|(f^1_{\mathcal{P}} - \varphi_i)|_{\bar{s}^1_i}\right\|_\infty &= \max_{\mathbf{p}^1_j \in \mathrm{vert}(\bar{s}^1_i)} \left\|f^1_{\mathcal{P}}(\mathbf{x}) - \varphi_i(\mathbf{x})\right\| \\
&= \max_{\mathbf{p}^1_j \in \mathrm{vert}(\bar{s}^1_i)} \left\|\mathbf{A}^1_i(\mathbf{p}^1_j - \mathbf{p}^1_j) + \mathbf{q}^1_j - [\mathbf{A}^2_i(\mathbf{p}^1_j - \mathbf{p}^2_j) + \mathbf{q}^2_j]\right\| \\
&\leq \max_{\mathbf{p}^1_j \in \mathrm{vert}(\bar{s}^1_i)} \left\|\mathbf{q}^1_j - \mathbf{q}^2_j\right\| + \left\|\mathbf{A}^2_i\right\| \left\|\mathbf{p}^1_j - \mathbf{p}^2_j\right\| \\
&< \left(c + \left\|\mathbf{A}^2_i\right\|\right)\epsilon
\end{aligned}$$

It remains to bound $\left\|\psi_i|_{\bar{s}^1_i}\right\|_\infty$. Recall that ψ_i is a continuous piecewise linear function and $\psi_i|_{\bar{s}^2_i} \equiv 0$. Since $\left\|\mathbf{p}^1_j - \mathbf{p}^2_j\right\| < \epsilon$, it follows that $\bar{s}^1_i \subseteq \mathcal{D}(\bar{s}^2_i, \epsilon)$, where $\mathcal{D}(\cdot,\cdot)$ is the dilation, defined in Section 2.1.4. Moreover since $\epsilon < r_3$, by Claim 2.15, $\mathcal{D}(\bar{s}^2_i, \epsilon) \cap \bar{s}^2_k \neq \emptyset$ if and only if $\bar{s}^2_k \cap \bar{s}^2_i \neq \emptyset$, or equivalently $\bar{s}^2_k \in \mathrm{St}\,\bar{s}^2_i$. These facts allow $\left\|\psi_i|_{\bar{s}^1_i}\right\|_\infty$ to be rewritten as

$$\left\|\psi_i|_{\bar{s}^1_i}\right\|_\infty = \max_{\bar{s}^2_k \in \mathrm{St}\,\bar{s}^2_i} \left\|\psi_i|_{\bar{s}^1_i \cap \bar{s}^2_k}\right\|_\infty .$$

Let $\bar{s}^2_k \in \mathrm{St}\,\bar{s}^2_i$ and let $\mathbf{p}^2_j \in \mathrm{vert}(\bar{s}^2_k \cap \bar{s}^2_i)$. Then we can write $\psi_i|_{\bar{s}^2_k}(\mathbf{x}) = (\mathbf{A}^2_k - \mathbf{A}^2_i)(\mathbf{x} - \mathbf{p}^2_j)$, and by Claim 3.1, $\psi_i|_{\bar{s}^2_k \cap \bar{s}^2_i}(\mathbf{x}) \equiv \mathbf{0}$. Thus for $\mathbf{x} \in \bar{s}^1_i \cap \bar{s}^2_k$,

$$\begin{aligned}
\|\psi_i(\mathbf{x})\| &= \left\|(\mathbf{A}^2_k - \mathbf{A}^2_i)(\mathbf{x} - \mathbf{x}_k)\right\| \\
&\leq \left\|\mathbf{A}^2_k - \mathbf{A}^2_i\right\| \delta\big(\mathbf{x}, \mathrm{aff}(\bar{s}^2_k \cap \bar{s}^2_i)\big) .
\end{aligned}$$

But $\bar{s}^1_i \cap \bar{s}^2_k \subset \mathcal{D}(\bar{s}^2_i, \epsilon) \cap \bar{s}^2_k$, so by Claim 2.12, $\delta\big(\mathbf{x}, \mathrm{aff}(\bar{s}^2_k \cap \bar{s}^2_i)\big) < \kappa_{i,k}\epsilon$, where $\kappa_{i,k}$ is a constant depending on $\mathcal{P}_2$ given by Equation A.15. Thus,

$$\left\|\psi_i|_{\bar{s}^1_i}\right\|_\infty \leq \left[\max_{\bar{s}^2_k \in \mathrm{St}\,\bar{s}^2_i} \left\|\mathbf{A}^2_k - \mathbf{A}^2_i\right\| \kappa_{i,k}\right] \epsilon$$

It follows that

$$\left\|(f^1_{\mathcal{P}} - f^2_{\mathcal{P}})|_{\bar{s}^1_i}\right\|_\infty < \left(c + \left\|\mathbf{A}^2_i\right\| + \max_{\bar{s}^2_k \in \mathrm{St}\bar{s}^2_i} \left\|\mathbf{A}^2_k - \mathbf{A}^2_i\right\| \kappa_{i,k}\right) \epsilon.$$

Thus $\left\|f^1_{\mathcal{P}} - f^2_{\mathcal{P}}\right\|_\infty < c'\epsilon$, where

$$c' = \max_i \left(c + \left\|\mathbf{A}^2_i\right\| + \max_{\bar{s}^2_k \in \mathrm{St}\bar{s}^2_i} \left\|\mathbf{A}^2_k - \mathbf{A}^2_i\right\| \kappa_{i,k}\right). \tag{3.7}$$

□

3.2 Interpolation with PL Functions

Interpolating a data set is a convenient method to predict a functional relationship for a data set. Consider a set of data pairs $\mathcal{Z} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N_d}$ produced by a data generating

function $f^* : \mathbb{R}^d \to \mathbb{R}^c$, $\mathbf{y}_i = f^*(\mathbf{x}_i)$. A continuous function f interpolates $\mathcal{Z}$ if $f(\mathbf{x}_i) = \mathbf{y}_i$ for $i = 1, \ldots, N_d$. By assuming that the data generating function comes from a certain class of functions, bounds may be placed on the error between the data generating and interpolating functions.

It is straightforward to construct a piecewise linear interpolating function for the data set $\mathcal{Z}$. Let $P = \{\mathbf{p}_i\}_{i=1}^{N_d}$ where $\mathbf{p}_i = \mathbf{x}_i$ and let $Q = \{\mathbf{q}_i\}_{i=1}^{N_d}$ where $\mathbf{q}_i = \mathbf{y}_i$, and find $\mathcal{S}$ such that $\mathcal{T}(P, \mathcal{S})$ is a triangulation. $\mathcal{S}$ could be computed, for example, using code for computing either the Delaunay or regular triangulation of P, in which case $|\mathcal{T}(P, \mathcal{S})| = \text{conv}(P)$.

Of course, in general there are many different possible combinatorial structures $\mathcal{S}$ that could be used with a given data set. Early work, such as [BA76, Gre75], on linear interpolation of functions from a Sobolev space[4] over a subset of $\mathbb{R}^2$ found that linear interpolation error bounds depended on the smallest angle in a triangle being bounded away from 0 or more critically the largest angle in a triangle being bounded away from π. For this reason, the Delaunay triangulation of P is often used because of the property from Section 2.3.3 that the Delaunay triangulation has the smallest maximum min-containment radius. In other words, the d-simplices of a Delaunay triangulation tend to be "fat."

In [Rip92], Rippa demonstrates that the bounds on linear interpolation error can be tightened if information about the directionality of the second derivative of the function is known. If the second derivative is high in one direction and low in others, then it is good to have long skinny triangles that are thin in the direction of the second derivative instead of fat triangles. This leads to the concept of "data dependent triangulations," which take into account Q as well as P in order to determine $\mathcal{S}$.

3.2.1 Data Dependent Triangulation

"Data dependent triangulations" are introduced by Dyn et al. in [DLR90a, DLR90b]. A data dependent triangulation chooses the combinatorial structure $\mathcal{S}$ of the PL interpolator based on Q as well as P. Specifically, the combinatorial structure $\mathcal{S}$ is chosen to minimize a "data dependent criterion."[5] Dyn et al. consider exclusively functions from $\mathbb{R}^2$ to $\mathbb{R}$. They propose several data dependent criteria for which the intuitive effect is to penalize the change in slope of the PL interpolator when moving from one triangle to another. The

[4] A Sobolev space is a subspace of L_p, for which the mixed derivatives up to some order are also in L_p. The Sobolev Hilbert space is often written as H^l. Functions in H^l as well as their mixed derivatives up to order l are in L_2. The norm for this space is the Sobolev norm, which is similar to the L_2 norm but also takes into derivatives account of the functions.

[5] A criterion that involves the codomain points Q does not necessarily depend on those points. The roughness criterion $R(P, Q, \mathcal{S}) = |f_{\mathcal{P}}|_{H^1}$ where $|\cdot|_{H^1}$ is the Sobolev semi-norm, given by

$$|g|_{H^1}^2 = \sum_{\bar{s}_i / in \mathcal{T}(P,\mathcal{S})} |g|_{H^1}^2 , \quad \text{where} \quad |g|_{H^1}^2 = \int_{\bar{s}_i} \left[\left(\frac{\partial g}{\partial x} \right)^2 + \left(\frac{\partial g}{\partial y} \right)^2 \right] dx\, dy$$

for a domain that is a subset of $\mathbb{R}^2$, seems reasonable. The appearance of $f_{\mathcal{P}}$ in the criterion indicates that Q is used in computing the criterion, but in [Rip90] Rippa shows that the minimal roughness triangulation is actually the Delaunay triangulation of P, so the minimal roughness triangulation does not depend on Q!

optimization of $\mathcal{S}$ is similar to Lawson's method for computing the 2-dimensional Delaunay triangulation, described in Section 2.3.3. The Dyn et al. algorithm for data dependent triangulations [DLR90a, DLR90b] is as follows:

1. Compute an initial triangulation of P, e.g. Delaunay
2. While there exists an edge for which flipping will reduce the data dependent criterion
   Flip edge

The two most successful data dependent criteria that Dyn et al. discovered are called "angle between normals" (ABN) and "jump in normal derivatives" (JND). Since their work considers linear interpolation of data from $\mathbb{R}^2$ to $\mathbb{R}$, the graph of the interpolated function over a triangle is a plane in $\mathbb{R}^3$. To evaluate an edge for the ABN criterion, first compute the unit normal vectors to the planes of the two triangles incident to the edge. The angle between the normal vectors can be computed easily from their inner product. An edge should be flipped if it reduces the angle between the normal vectors, or equivalently, if it reduces the magnitude of the inner product. For the JND criterion, let the PL functions over two triangles incident on an edge be given by $f_1(\mathbf{x}) = {\mathbf{c}_1}^{\mathrm{T}}\mathbf{x} + d_1$ and $f_2(\mathbf{x}) = {\mathbf{c}_2}^{\mathrm{T}}\mathbf{x} + d_2$, where $\mathbf{c}_i \in \mathbb{R}^2$ and $d_i \in \mathbb{R}$, and let $\mathbf{n} \in \mathbb{R}^2$ be the unit vector perpendicular to the edge. Dyn et al. define the JND criterion as $\left|\mathbf{n}^{\mathrm{T}}(\mathbf{c}_1 - \mathbf{c}_2)\right|$. By Claim 3.1 *(ii)*, the vector $\mathbf{c}_1 - \mathbf{c}_2$ will be parallel to $\mathbf{n}$, so the JND criterion could be written more simply as $\|\mathbf{c}_1 - \mathbf{c}_2\|$. An edge is flipped if it reduces the JND criterion.

Several difficulties arise in generalizing data dependent triangulations to higher dimensions. For $d > 2$, the number of d-simplices in a triangulation is not constant and flipping a subcomplex (rather than an edge) is more complex. For example, a generic flip in $\mathbb{R}^3$ converts between two tetrahedra and three tetrahedra. The data dependent criterion must take this into account. The present criteria take advantage of the structure of $\mathbb{R}^2$, e.g. only a single edge is modified during a flip. Should the data dependent criterion for a subcomplex in $\mathbb{R}^d$ be the sum of a local criterion over each interior facet in the complex (of which there are different numbers for the two configurations) or perhaps just the local criterion evaluated at the worst facet? Generally this dissertation focuses on functions from $\mathbb{R}^d$ to $\mathbb{R}^c$ (or to $\mathbb{R}^d$ more specifically) rather than to $\mathbb{R}$. How should effects on various codomain coordinates be traded off?

Data dependent triangulations, and specifically the JND criterion, serve as the inspiration for the local retriangulation performed in the MINVAR algorithm, presented in Section 4.3.

3.3 Least Squares PL for a Fixed Domain Triangulation

Since an interpolating function is required to match the data at all the sample points, interpolation does not provide noise rejection or "smoothing." Moreover, interpolation

requires that the entire data set be held, whereas it may be preferable to "summarize" the data with a low number of parameters. This gives rise to the approximation problem. Consider again a set of data pairs $\mathcal{Z} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N_d}$ produced by a data generating function $f^* : \mathbb{R}^d \to \mathbb{R}^c$, $\mathbf{y}_i = f^*(\mathbf{x}_i)$. Typically one wants to find an approximation f to the data set that minimizes the mean square error, given by

$$M.S.E. = \frac{1}{N_d} \sum_{i=1}^{N_d} \|\mathbf{y}_i - f(\mathbf{x}_i)\|^2 . \tag{3.8}$$

Suppose $\mathcal{T}(P, \mathcal{S})$ is a triangulation with n vertices and one wishes to find a piecewise linear approximation $f_{\mathcal{P}}$, parameterized by $\mathcal{P} = (P, Q, \mathcal{S})$, to the data set that minimizes the mean square error. That is, the only free parameters are the codomain vertices Q. In this case, it is well known, e.g. [Jul99, dB01, Chu88], that the approximation problem is linear-in-parameters and can be solved by standard linear least squares. The following discussion describes the solution.

Let

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \cdots & \mathbf{p}_n \end{bmatrix}, \tag{3.9}$$

$$\mathbf{Q} = \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 & \cdots & \mathbf{q}_n \end{bmatrix}. \tag{3.10}$$

Let the operator $\pi : |\mathcal{T}(P, \mathcal{S})| \to \mathbb{R}^n$ be such that

(i) $\pi_i(\mathbf{x}) \geq 0$ for $i = 1, \ldots, n$,
(ii) $\sum_{i=1}^{n} \pi_i(\mathbf{x}) = 1$, and
(iii) $\{i | \pi_i(\mathbf{x}) \neq 0\} \in \mathcal{S}$,
(iv) $\mathbf{x} = \mathbf{P}\, \pi(\mathbf{x})$

where $\pi_i(\mathbf{x})$ indicates the i^{th} component of $\pi(\mathbf{x})$. That is, $\pi(\mathbf{x})$ are the barycentric coordinates of $\mathbf{x}$ with respect to the vertices of a simplex of $\mathcal{T}(P, \mathcal{S})$ to which $\mathbf{x}$ belongs. $\pi(\cdot)$ is well defined due to the uniqueness of barycentric coordinates, Claim 2.4. Using this operator, the global definition of the PL function parameterized by $\mathcal{P} = (P, Q, \mathcal{S})$ can be written as

$$f_{\mathcal{P}}(\mathbf{x}) = \mathbf{Q}\, \pi(\mathbf{x}), \tag{3.11}$$

for all $\mathbf{x} \in |\mathcal{T}(P, \mathcal{S})|$. It is useful to compare this definition with the piecewise definition of Equation 3.1. Notice that the codomain vertices $\mathbf{q}_i$ appear linearly in the above equation, and since in the present formulation the $\mathbf{q}_i$'s are the only free parameters, $f_{\mathcal{P}}$ is called linear-in-parameters. Each element $\pi_i(\mathbf{x})$ of $\pi(\mathbf{x})$ may be interpreted as a scalar valued PL basis function parameterized by $(P, \hat{Q}_i, \mathcal{S})$, where $\hat{q}_i = 1$ and $\hat{q}_j = 0$ for $j \neq i$. π_i has a nonlinear dependence on the domain vertices, and thus $f_{\mathcal{P}}$ has a nonlinear dependence on the domain vertices, so if they were included as parameters, the linear-in-parameter structure would be broken. Let

$$\mathbf{B} = \begin{bmatrix} \pi(\mathbf{x}_1) & \pi(\mathbf{x}_2) & \cdots & \pi(\mathbf{x}_{N_d}) \end{bmatrix}, \tag{3.12}$$

$$\mathbf{Y} = \begin{bmatrix} \mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_{N_d} \end{bmatrix}. \tag{3.13}$$

Then the least squares problem is given by

$$\arg\min_{\mathbf{Q}} \sum_{i=1}^{N_d} \|\mathbf{y}_i - f_{\mathcal{P}}(\mathbf{x}_i)\|^2 \ = \arg\min_{\mathbf{Q}} \|\mathbf{Y} - \mathbf{QB}\|_F^2 \ , \tag{3.14}$$

where the $\|\cdot\|_F$ is the Frobenius norm. This is the classic least squares problem. If $\mathbf{B}$ has rank n, a requirement on the richness of the data set, then $\mathbf{Q}$ is given by

$$\mathbf{Q} = \mathbf{YB}^{\mathrm{T}}(\mathbf{BB}^{\mathrm{T}})^{-1}.$$

3.4 Inverting PL functions

One of the most compelling properties of piecewise linear functions is the ability to check invertibility and invert in closed form. Let $f_{\mathcal{P}}$ be a piecewise linear function parameterized by $\mathcal{P} = (P, Q, \mathcal{S})$. If $\mathcal{T}(Q, \mathcal{S})$ is a triangulation, then the piecewise linear function is one-to-one and thus invertible on its range. Moreover, the inverse, ${f_{\mathcal{P}}}^{-1}$, is a piecewise linear function parameterized by $\mathcal{P}^{-1} = (Q, P, \mathcal{S})$. These facts are established in this section.

Consider a continuous function $f : A \to \mathbb{R}^c$, $A \subset \mathbb{R}^d$ with nonempty interior. f is invertible if and only if it is one-to-one and onto. Since a function is always onto its range, if f is just one-to-one, then f is invertible on its range. That is, there exists $f^{-1} : C \to A$, $C = f(A)$, such that $f^{-1}(f(\mathbf{x})) = \mathbf{x}$ for all $\mathbf{x} \in A$ and $f(f^{-1}(\mathbf{y})) = \mathbf{y}$ for all $\mathbf{y} \in C$. One may check that a PL function $f_{\mathcal{P}}$ is one-to-one by determining whether $\mathcal{T}(Q, \mathcal{S})$ is a triangulation. This fact is established by Claim 3.5, but first one more fact about triangulations is needed.

Claim 3.4. *Let $\mathcal{T}$ be a triangulation. Let $\mathbf{x} \in |\mathcal{T}|$. There exists a unique $s \in \mathcal{T}$ such that $\mathbf{x} \in s$ and $s \leq s'$ for all $s' \in \mathcal{T}$ such that $\mathbf{x} \in s'$*

Proof. Let

$$s = \bigcap_{s' \in \mathcal{T},\, \mathbf{x} \in s'} s'.$$

It follows from the definition of triangulation that $s \in \mathcal{T}$. By construction $\mathbf{x} \in s$, moreover $s \leq s'$ for any other $s' \in \mathcal{T}$ that contains $\mathbf{x}$. □

Claim 3.5. *Let $f_{\mathcal{P}}$ be a piecewise linear function parameterized by $\mathcal{P} = (P, Q, \mathcal{S})$. Then $\mathcal{T}(Q, \mathcal{S})$ is a triangulation if and only if $f_{\mathcal{P}}$ is one-to-one.*

Proof. Let $\mathcal{T}(C, \mathcal{S})$ be a triangulation. Let $f_{\mathcal{P}}(\mathbf{x}) = f_{\mathcal{P}}(\mathbf{y})$ for $\mathbf{x}, \mathbf{y} \in \mathcal{T}(P, \mathcal{S})$. Since $\mathcal{T}(Q, \mathcal{S})$ is a triangulation, by Claim 3.4 there exists a unique $s \in \mathcal{T}(Q, \mathcal{S})$ such that $f_{\mathcal{P}}(\mathbf{x}) \in s$ and $s \leq s'$ for all other $s' \in \mathcal{T}(Q, \mathcal{S})$ such that $f_{\mathcal{P}}(\mathbf{x}) \in s'$. Let $\mathrm{vert}(s) = \{\mathbf{q}_{j_1}, \ldots, \mathbf{q}_{j_k}\}$. Let $\hat{s} \in \mathcal{T}(P, \mathcal{S})$ with $\mathrm{vert}(\hat{s}) = \{\mathbf{p}_{j_1}, \ldots, \mathbf{p}_{j_k}\}$. Since $f_{\mathcal{P}}(\mathbf{x}) \in s$ and $f_{\mathcal{P}}(\mathbf{y}) = f_{\mathcal{P}}(\mathbf{x})$, it follows that $\mathbf{x}, \mathbf{y} \in \hat{s}$. Let $\bar{s}$ be a d-simplex such that $\hat{s} \leq \bar{s}$. Since $\mathcal{T}(Q, \mathcal{S})$ is a triangulation, $f_{\mathcal{P}}(\bar{s})$ is also a d-simplex, and hence the linear part of $f_{\mathcal{P}}|_{\bar{s}}$ must be full rank. It follows that $\mathbf{x} = \mathbf{y}$. Thus $f_{\mathcal{P}}$ is one-to-one.

Let $f_{\mathcal{P}}$ be one-to-one. Let $\alpha \in \mathcal{S}$. Then the points $Q(\alpha)$ must be affinely independent, otherwise $f_{\mathcal{P}}$ would not be one-to-one. Thus each simplex $\hat{s} \in \mathcal{T}(P, \mathcal{S})$ is mapped by $f_{\mathcal{P}}$ into a corresponding simplex $s \in \mathcal{T}(Q, \mathcal{S})$. Since $f_{\mathcal{P}}$ is continuous and one-to-one, it follows that $\mathcal{T}(Q, \mathcal{S})$ is a triangulation. □

Another way of stating Claim 3.5 is that $f_{\mathcal{P}}$ is one-to-one if and only if $\mathcal{T}(Q, \mathcal{S})$ is a valid embedding of $\mathcal{S}$. Section 2.4 describes how to validate an embedding of an abstract simplicial complex.

This dissertation is predominantly interested in the case where the domain and codomain have the same dimension[6], $c = d$. A PL function having the same dimension for the domain and codomain is referred to as a *d-dimensional PL function*. The *inverse* of a PL function in this case means the inverse over the range of a one-to-one PL function. If $f_{\mathcal{P}}$ is one-to-one, then the inverse (over the range) of $f_{\mathcal{P}}$ is given by the piecewise linear function $f_{\mathcal{P}^{-1}}$ parameterized by $\mathcal{P} = (Q, P, \mathcal{S})$. This fact is proven in the following claim.

Claim 3.6. *Let $f_{\mathcal{P}}$ be a one-to-one d-dimensional PL function parameterized by $\mathcal{P} = (P, Q, \mathcal{S})$. Then ${f_{\mathcal{P}}}^{-1} = f_{\mathcal{P}^{-1}}$ where $f_{\mathcal{P}^{-1}}$ is a PL function parametrized by $\mathcal{P}^{-1} = (Q, P, \mathcal{S})$.*

Proof. Since $f_{\mathcal{P}}$ is one-to-one, $\mathcal{T}(Q, \mathcal{S})$ is a triangulation by Claim 3.5. Thus, $\mathcal{P}^{-1} = (Q, P, \mathcal{S})$ is a valid parameterization of a PL function $f_{\mathcal{P}^{-1}}$. Let $\bar{s}_i \in \mathcal{T}(P, \mathcal{S})$ be a d-simplex with $\text{vert}(\bar{s}_{p,i}) = \{\mathbf{p}_{i_1}, \mathbf{p}_{i_2}, \ldots, \mathbf{p}_{i_{d+1}}\}$. Let $\bar{s}_{q,i} \in \mathcal{T}(Q, \mathcal{S})$ with $\text{vert}(\bar{s}_{q,i}) = \{\mathbf{q}_{i_1}, \mathbf{q}_{i_2}, \ldots, \mathbf{q}_{i_{d+1}}\}$. Then $f_{\mathcal{P}}(\mathbf{p}_{i_j}) = \mathbf{q}_{i_j}$, but $f_{\mathcal{P}^{-1}}(\mathbf{q}_{i_j}) = \mathbf{p}_{i_j}$. Moreover $f_{\mathcal{P}}$ affinely maps $\bar{s}_{p,i}$ into $\bar{s}_{q,i}$ and $f_{\mathcal{P}^{-1}}$ affinely maps $\bar{s}_{q,i}$ into $\bar{s}_{p,i}$. It follows that $f_{\mathcal{P}^{-1}}$ is the inverse of $f_{\mathcal{P}}$. □

Thus, given the parameterization of an invertible d-dimensional PL function, one can immediately write down the parameterization of the inverse of that function. Closed form invertibility is one of the primary motivating benefits in the present study of PL approximations.

[6] If a continuous one-to-one function $f : A \to \mathbb{R}^c$, $A \subset \mathbb{R}^d$ with nonempty interior, has a codomain of higher dimension than the domain, i.e. $c > d$, then $f(A)$ will be a thin set in the codomain. In this case a pseudo-inverse of f could be constructed by extending f^{-1} to the whole codomain, and then $f_B^{-1}(f(\mathbf{x})) = \mathbf{x}$ but $f(f_B^{-1}(\mathbf{y})) \neq \mathbf{y}$ unless $\mathbf{x} \in f(A)$.

CHAPTER 4

The MINVAR Algorithm

Chapter 3 presented a useful parameterization of PL functions, a triplet $(P, Q, \mathcal{S})$ consisting of a set of domain vertices P, corresponding set of codomain vertices Q, and an abstract simplicial complex $\mathcal{S}$, such that $\mathcal{T}(P, \mathcal{S})$ is a triangulation with $|\mathcal{T}(P, \mathcal{S})|$ being the domain of the PL function. In this chapter we consider the problem of constructing a PL approximation to a set of data, and present MINVAR, a novel algorithm to accomplish this task. The chapter concludes with a numerical example.

Consider $\mathcal{Z} = (\mathbf{x}_i, \mathbf{y}_i)_{i=1}^{N_d}$, $\mathbf{x} \in \mathbb{R}^d$ and $\mathbf{y} \in \mathbb{R}^c$, a data set of N_d input-output pairs. In function approximation, a function from a parameterized family of functions is chosen in order to minimize some error criterion. For the approximation problem, in contrast to the interpolation problem, there must be much larger quantity of data than the number of parameters in the approximation. As a rule of thumb, there should be an order of magnitude more input-output pairs than vertices in the PL approximation. This dissertation concerns approximations that minimize the mean squared error (MSE),

$$MSE = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{y}_i - f_{\mathcal{P}}(\mathbf{x}_i)\|^2 , \tag{4.1}$$

called least squares approximation. Under the assumption of independent identically distributed additive Gaussian noise on the output data, the parameters of the least squares approximation are the maximum likelihood estimate of the parameters given the data [Poo94]. This statistical property promotes interest in the least squares approximation[1].

Section 3.3 showed that if the only free parameters of a PL function are the codomain vertices Q (i.e. the domain vertices P and the abstract simplicial complex $\mathcal{S}$ are fixed), then the least squares PL approximation to a data set is linear in the codomain vertices and can thus be solved using standard linear least squares methods. This is the typical approach taken by the spline literature, such as [Jul99, dB01, Chu88]. Fixing the domain

[1] MSE is not, however, the only reasonable error criterion. Two other popular choices are *(i)* to replace the 2-norm in Equation 4.1 with the 1-norm or *(ii)* to replace the summation in Equation 4.1 with maximum. The former handles outliers better, while the latter guarantees that the approximation is always close to the data.

triangulation $\mathcal{T}(P, \mathcal{S})$ is equivalent to selecting a family of PL basis functions over $|\mathcal{T}(P, \mathcal{S})|$. The codomain vertices are coefficients of the basis functions. This is a fine approach so long as one is willing to commit to a fixed set of basis functions.

What if the data generating function (that is, the "true" source of the data set that is being approximated) has high curvature in some regions but is nearly flat in other regions? If the basis functions, or equivalently the domain d-simplices, are distributed uniformly, then it may well require a huge number of vertices in order to approximate the data generating function sufficiently, since unnecessary vertices will be placed in the region of low curvature in order to yield a good approximation in the areas of high curvature. In order to approximate such a function with a PL function, the domain simplices should be large in the flat regions and small in the regions of high curvature parsimoniously. Ideally, the basis functions would adapt themselves to the data set, placing more vertices in areas where the curvature of the function is higher. Gridding methods in finite element analysis and nonparametric statistical methods build up the domain triangulation incrementally, adapting the triangulation to the data. The approach here is different. Given a fixed number of vertices, both the domain and codomain vertices are considered as free parameters. One difficulty that arises is that there is no explicit representation for the space of valid PL parameters, though given a specific parameterization one can algorithmically check that it defines a good PL function (if $\mathcal{T}(P, \mathcal{S})$ is a triangulation, plus $\mathcal{T}(Q, \mathcal{S})$ for a PL homeomorphism). Moving the domain vertices makes the problem nonlinear- and moreover nonconvex-in-parameters, so one is faced with local minima and iterative optimization procedures.

One option would be to apply gradient descent directly to the MSE. This approach proves to be computationally expensive and slow, as illustrated by the numerical studies presented in Section 6.3. Gradient descent guarantees a reduction in the MSE from the point in the initial point parameter space, but the descent can often get stuck in a local minimum of the MSE, which could be much worse than the global minimum.

The remainder of this chapter presents the MINVAR algorithm, a novel non-gradient method for computing a good d-dimensional piecewise linear approximation to a set of data.

4.1 The MINVAR Algorithm

The MINVAR algorithm is an iterative scheme to generate a locally good PL approximation to data. Similar to algorithms proposed in the piecewise polynomial approximation literature [Bai94, Kio80, Kio81, TB97], MINVAR takes advantage of the piecewise structure of PL functions, using easy-to-compute local least squares affine approximations to determine how to modify the PL approximation.

Let $\mathcal{Z} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N_s}$, where $\mathbf{x}_i \in D \subseteq \mathbb{R}^d$ and $\mathbf{y}_i \in \mathbb{R}^c$, be the set of input-output

data to be approximated. The MINVAR algorithm iteratively improves a piecewise linear approximation to the data, $f_{\mathcal{P}}^{(k)}$, parameterized by $\mathcal{P}^{(k)} = (P^{(k)}, Q^{(k)}, \mathcal{S})$, such that $\left|\mathcal{T}(P^{(k)}, \mathcal{S})\right| = D$. A superscript in parentheses denotes iteration number. The algorithm consists of two stages. The first stage partitions the data into subsets according to the d-simplices of $\mathcal{T}(P^{(k)}, \mathcal{S})$ and computes the least squares affine approximation for each subset of the data. The second stage chooses $P^{(k+1)}$ and $Q^{(k+1)}$ to make $f_{\mathcal{P}}^{(k+1)}$, a continuous PL function, be "close" to the discontinuous approximation from the first stage. The stages are then iterated.

A vertex $\mathbf{p}_i \in \partial\left|\mathcal{T}(P, \mathcal{S})\right|$ is called a domain *boundary vertex*. Recall from Chapter 3 that the domain of $f_{\mathcal{P}}$ is $|\mathcal{T}(P, \mathcal{S})|$, so moving a boundary vertex will in general change the domain of the PL function. Since we desire a fixed domain, the present exposition of the MINVAR algorithm holds fixed all domain boundary vertices. In Section 4.2 this is relaxed to allow certain domain boundary vertices to move in appropriately chosen affine subspaces such that the domain is unaffected.

From an initial parameterization $\mathcal{P}^{(0)} = (P^{(0)}, Q^{(0)}, \mathcal{S})$, the MINVAR algorithm generates a sequence of parameterizations, $\mathcal{P}^{(k)} = (P^{(k)}, Q^{(k)}, \mathcal{S})$. Really only $P^{(0)}$ and $\mathcal{S}$ need to be specified initially, since the algorithm does not use the $Q^{(k)}$ to determine the succeeding approximation. The algorithm generates the sequence of approximations as follows:

MINVAR Algorithm

1. Partition the data set $\mathcal{Z}$ into subsets $\mathcal{Z}_j$ corresponding to the d-simplices, $\bar{s}_1, \ldots, \bar{s}_N$ of $\mathcal{T}(P^{(k)}, \mathcal{S})$

2. Compute the least squares affine approximation, $L_j(\mathbf{x}) = \hat{\mathbf{A}}_j\mathbf{x} + \hat{\mathbf{b}}_j$, for each data subset $\mathcal{Z}_j$

3. Update the vertex locations,

$$\mathbf{p}_i^{(k+1)} = \underset{x\in\mathbb{R}^d}{\arg\min}\ \operatorname{var} L^i(\mathbf{x}) + \lambda\left\|\mathbf{x} - \mathbf{p}_i^{(k)}\right\|^2 \qquad \forall\, \mathbf{p}_i \notin \partial\left|\mathcal{T}(P, \mathcal{S})\right| \tag{4.2}$$

$$\mathbf{p}_i^{(k+1)} = \mathbf{p}_i^{(k)} \qquad \text{otherwise} \tag{4.3}$$

where

$$\operatorname{var} L^i(\mathbf{x}) = \sum_{\bar{s}_j \in \mathrm{St}\{\mathbf{p}_i\}} \left\| L_j(\mathbf{x}) - \frac{1}{N_i} \sum_{\bar{s}_k \in \mathrm{St}\{\mathbf{p}_i\}} L_k(\mathbf{x}) \right\|^2 \tag{4.4}$$

and

$$\mathbf{q}_i^{(k+1)} = \frac{1}{N_i} \sum_{\bar{s}_j \in \mathrm{St}\{\mathbf{p}_i\}} L_j\left(\mathbf{p}_i^{(k+1)}\right) \qquad \text{for } i = 1, \ldots, n \tag{4.5}$$

4. If the vertices are not converged, then $k \leftarrow k + 1$, go to 1

In Equations 4.4 and 4.5 the summations are over $\bar{s}_j \in \mathrm{St}\{\mathbf{p}_i\}$, the set of d-simplices from $\mathcal{T}(P, \mathcal{S})$ that have $\mathbf{p}_i$ as a vertex[2], and $L_j(\mathbf{x})$ is then the least squares affine fit corresponding to $\bar{s}_j \in \mathrm{St}\{\mathbf{p}_i\}$. L^i is the collection of least squares affine fits $L_j(\mathbf{x})$ corresponding to $\bar{s}_j \in \mathrm{St}\{\mathbf{p}_i\}$. From Chapter 2, N_i is defined to be the number of d-simplices in $\mathrm{St}\{\mathbf{p}_i\}$, which is given by

$$N_i = \sum_{\bar{s}_k \in \mathrm{St}\{\mathbf{p}_i\}} 1.$$

The remainder of this section provides an intuitive description of the MINVAR algorithm, while details on computing the least squares affine approximations of Step 2 and the quadratic minimization of Equation 4.2 will be presented in the following subsection.

Consider the function that on each d-simplex of $\mathcal{T}(P^{(k)}, \mathcal{S})$ takes its corresponding least squares affine approximation as computed in Step 2 of the algorithm. This function is the optimal discontinuous piecewise linear approximation for the triangulation $\mathcal{T}(P^{(k)}, \mathcal{S})$. If the triangulation were adjusted so that the optimal discontinuous piecewise linear approximation was everywhere continuous, then we would have a locally good PL approximation to the data (recall that PL is used to indicate *continuous* piecewise linear functions), since any other PL approximation over the same triangulation would have higher mean squared error and any nearby triangulation (with respect to the domain vertices) will generally have an optimal discontinuous piecewise linear approximation that is not continuous. The computation in Step 3 may be interpreted as adjusting the domain vertex in order to make the optimal discontinuous piecewise linear approximation closer to continuous. The following paragraphs will illustrate how the quadratic minimization of Equation 4.2 achieves this.

For $d = 1$, a vertex $\mathbf{p}_i$ is shared by two 1-simplices (unless $\mathbf{p}_i \in \partial\,|\mathcal{T}(P, \mathcal{S})|$, in which case it is in a single 1-simplex). If the least squares approximations are not parallel, then $\operatorname{var} L^i(\mathbf{x}) = 0$ at the domain location where the least squares approximations intersect, illustrated in Figure 4.1. If $\lambda = 0$, MINVAR moves the domain and codomain vertices to the intersection point of the least squares approximations, called the "graph intersection point." The precursor to the MINVAR algorithm, presented in Chapter 6, is called the "Graph Intersection" for this reason. If $\lambda > 0$, MINVAR moves the domain vertex toward the graph intersection point, but is penalized from moving too far away from the previous domain vertex. If the algorithm reaches a fixed point, then over each domain interval the MINVAR approximation is the same as the least squares affine approximation to the data in that interval. The graph intersection point does not generalize to higher dimensions. Consider the case where $c = d$, $d > 1$. Generically, each pair of least squares approximations will have a unique intersection point, but each domain vertex will generically be a member of at

[2]The time index k on $\mathbf{p}_i$ is suppressed in $\bar{s}_j \in \mathrm{St}\{\mathbf{p}_i\}$. In the present exposition $\mathcal{S}$ does not changing, so $\mathbf{p}_i$ will be a vertex of the same set of d-simplices at each time step. Modification to MINVAR described in later sections update $\mathcal{S}$ periodically, in which case the set of d-simplices containing $\mathbf{p}_i$ will clearly depend on the time index. In an abuse of notation, we drop the index in this case as well in order to avoid overly cumbersome notation.

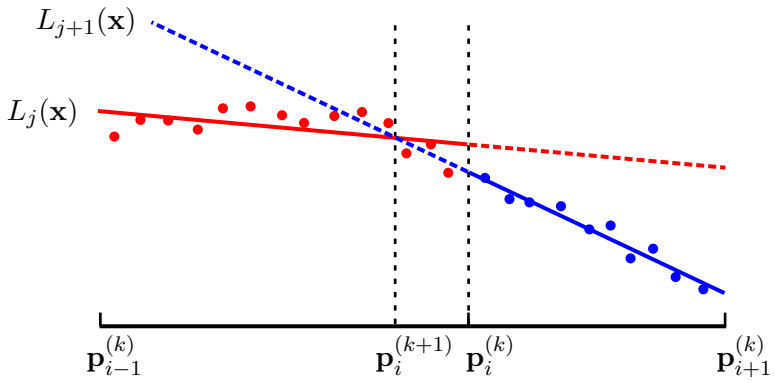

Figure 4.1: Moving a vertex with the scalar MINVAR algorithm. When $\lambda = 0$, the new location of the domain vertex $\mathbf{p}_i^{(k+1)}$ is the intersection point of the least squares affine approximations L_j and L_{j+1} corresponding to the 1-simplices (line segments) that neighbor $\mathbf{p}_i$.

least three d-simplices. Thus, for higher dimensions the least squares approximations will generally not all intersect at the same point.

Consider a vertex $\mathbf{p}_i$ and L^i, the collection of least squares affine approximations $L_j(\mathbf{x})$ corresponding to the d-simplices incident to $\mathbf{p}_i$, $\bar{s}_j \in \mathrm{St}\{\mathbf{p}_i\}$. Evaluating this set of N_i approximations at a point $\mathbf{x}$ in the domain yields a cluster of N_i points in the codomain, as illustrated in Figure 4.2. The nonnegative quantity $\operatorname{var} L^i(\mathbf{x})$ measures how tightly these points cluster about their mean. If $\operatorname{var} L^i(\mathbf{x}) = 0$ for some $\mathbf{x}$ then all of the approximations in L^i intersect at $\mathbf{x}$. If the domain vertices could be moved such that $\operatorname{var} L^i(\mathbf{p}_i) = 0$ for each $\mathbf{p}_i$, then the optimal discontinuous piecewise linear function over the triangulation $\mathcal{T}(P, \mathcal{S})$ would be continuous. In general, there will be no point in the domain the makes $\operatorname{var} L^i(\mathbf{x}) = 0$, so instead the algorithm moves the domain vertex to the location that minimizes $\operatorname{var} L^i(\mathbf{x})$ (or rather, the vertex is so moved if $\lambda = 0$). The λ regularization term is described below.

The location that minimizes $\operatorname{var} L^i(\mathbf{x})$ may be far away from the current location of the vertex, for example when the least squares approximations are nearly parallel in the scalar case. Moving the domain vertices like this may "tangle" the domain triangulation of the approximation. That is, it could prevent $\mathcal{T}(P, \mathcal{S})$ from being a triangulation due to overlapping simplices. The λ term in Equation 4.2 is a regularization that prevents a vertex from moving long distances. Tuning of λ is used to avoid tangles. Currently, choosing an good value for λ is an art. Nonzero λ also guarantees that Equation 4.2 will have a unique minimum, even if all the least squared approximations are parallel. This generally occurs when a domain vertex moves through one of its opposing faces.

The following subsection describes in further detail the calculations performed in MINVAR, specifically the computation of the least squares affine approximations and the quadratic

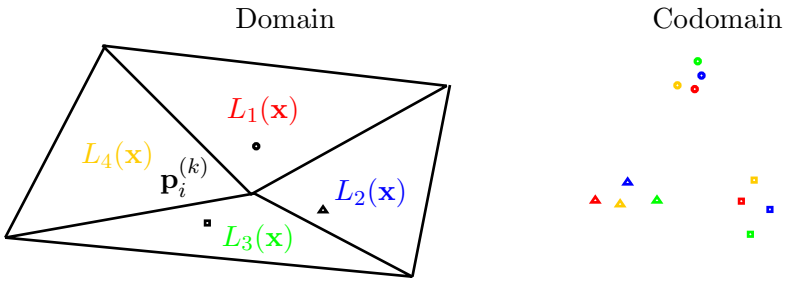

Figure 4.2: Moving a vertex in the d-dimensional MINVAR algorithm. In the illustration above, L^i, the collection of least squares affine approximations around $\mathbf{p}_i^{(k)}$, includes L_1, L_2, L_3, and L_4. Evaluating these approximations at a point in the domain gives a cluster of points in the codomain. Three different domain points and their corresponding clusters are illustrated above with circles, triangles, and squares. The MINVAR algorithm with $\lambda = 0$ moves $\mathbf{p}_i^{(k+1)}$ to the point in the domain that yields the tightest cluster of points in the codomain. Above, the circles illustrate the tightest cluster.

minimization in the "min var" step.

4.1.1 Calculations for MINVAR

Least Squares Affine Approximations

After the data has been sorted into subsets $\mathcal{Z}_j$, the least squares affine approximation for each data set must be computed. One method for computing these approximations is presented here. For convenience the subscript from the data subset is suppressed. Let $\mathcal{Z} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$. Let

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1{}^{\mathrm{T}} & 1 \\ \mathbf{x}_2{}^{\mathrm{T}} & 1 \\ \vdots & \\ \mathbf{x}_N{}^{\mathrm{T}} & 1 \end{bmatrix}, \qquad \mathbf{Y} = \begin{bmatrix} \mathbf{y}_1{}^{\mathrm{T}} \\ \mathbf{y}_2{}^{\mathrm{T}} \\ \vdots \\ \mathbf{y}_N{}^{\mathrm{T}} \end{bmatrix}, \qquad \mathbf{\Theta} = \begin{bmatrix} \hat{\mathbf{A}}^{\mathrm{T}} \\ \hat{\mathbf{b}}^{\mathrm{T}} \end{bmatrix}$$

Minimizing $\sum_{i=1}^N \|\mathbf{A}\mathbf{x}_i + \mathbf{b} - \mathbf{y}_i\|^2$ is equivalent to minimizing $\|\mathbf{X}\mathbf{\Theta} - \mathbf{Y}\|_F^2$, where F indicates the Frobenius norm. This is a standard least squares problem [Str88]. Usually the standard least squares problem has a vector of parameters rather than a matrix $\mathbf{\Theta}$, but we may consider the problem to be c independent standard least squares problems corresponding to the columns of $\mathbf{\Theta}$ and $\mathbf{Y}$. The least squares solution is given by solving the linear system $\mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{\Theta} = \mathbf{X}^{\mathrm{T}}\mathbf{Y}$, called the normal equations. The matrix $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ will be invertible so long as the dimension of the affine hull of the $\mathbf{x}_i$'s is d, a condition on the richness of the data set. If this condition is satisfied, then the least squares affine approximation is given by $\mathbf{\Theta} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{Y}$. Examining the structure of the matrices $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ and $\mathbf{X}^{\mathrm{T}}\mathbf{Y}$, we see

that

$$\begin{aligned}\mathbf{X}^{\mathrm{T}}\mathbf{X} &= \sum_{i=1}^{N} \begin{bmatrix}\mathbf{x}_i \\ 1\end{bmatrix} \begin{bmatrix}\mathbf{x}_i{}^{\mathrm{T}} & 1\end{bmatrix}, \\ \mathbf{X}^{\mathrm{T}}\mathbf{Y} &= \sum_{i=1}^{N} \begin{bmatrix}\mathbf{x}_i \\ 1\end{bmatrix} \mathbf{y}_i{}^{\mathrm{T}},\end{aligned}$$

which are scaled submatrices of the data set's empirical covariance matrix. These covariance submatrices can be constructed directly rather than forming $\mathbf{X}$ and $\mathbf{Y}$ and then computing $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ and $\mathbf{X}^{\mathrm{T}}\mathbf{Y}$.

Solving the least squares problem by solving the normal equations is often discouraged for numerical reasons. The condition number of $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ is the square of the condition number of $\mathbf{X}$. If the data in the subset is poorly distributed, for example if the data generating function has been regularly sampled in the domain and the subset corresponds to a skinny simplex such that most but not all of the data lies in a hyperplane, then significant error can arise in solving the normal equations. In this case, it is recommended to use a QR or SVD decomposition of $\mathbf{X}$ to minimize $\|\mathbf{X}\boldsymbol{\Theta} - \mathbf{Y}\|_F^2$. Details on this procedure may be found, for example, in [GV96]. Increased accuracy does not come cost free. If the number of data points N is much larger than d, then using the QR decomposition to solve the least squares problem uses more memory and about twice as many calculations.

Performing the "min var" Calculation

The function minimized in the "min var" calculation of Step 3 (Equation 4.2),

$$\operatorname{var} L^i(\mathbf{x}) + \lambda \left\| \mathbf{x} - \mathbf{p}_i^{(k)} \right\|^2$$

is a quadratic function of $\mathbf{x}$ and moreover is nonnegative, and under mild assumptions positive, definite. Such a function can be minimized in closed form by "completing the square," an algebraic manipulation such that $\mathbf{x}$ appears only in a quadratic form of a positive definite matrix and not in any linear terms. Claim B.1 proves that the solution to Equation 4.2 is given by

$$\mathbf{p}_i^{(k+1)} = -\mathbf{H}_i{}^{-1}\mathbf{h}_i, \tag{4.6}$$

where

$$\mathbf{H}_i = \left[\frac{1}{N_i} \sum_{\bar{s}_j \in \mathrm{St}\{\mathbf{p}_i\}} \hat{\mathbf{A}}_j{}^{\mathrm{T}} \hat{\mathbf{A}}_j \right] - \overline{\hat{\mathbf{A}}_i}^{\mathrm{T}} \overline{\hat{\mathbf{A}}_i} + \lambda \mathbf{I}, \tag{4.7}$$

$$\mathbf{h}_i = \left[\frac{1}{N_i} \sum_{\bar{s}_j \in \mathrm{St}\{\mathbf{p}_i\}} \hat{\mathbf{A}}_j{}^{\mathrm{T}} \hat{\mathbf{b}}_j \right] - \overline{\hat{\mathbf{A}}_i}^{\mathrm{T}} \overline{\hat{\mathbf{b}}_i} - \lambda \mathbf{p}_i^{(k)}, \tag{4.8}$$

$$\overline{\hat{\mathbf{A}}_i} = \frac{1}{N_i} \sum_{\bar{s}_j \in \mathrm{St}\{\mathbf{p}_i\}} \hat{\mathbf{A}}_j, \qquad \overline{\hat{\mathbf{b}}_i} = \frac{1}{N_i} \sum_{\bar{s}_j \in \mathrm{St}\{\mathbf{p}_i\}} \hat{\mathbf{b}}_j. \tag{4.9}$$

Thus the minimization of Equation 4.2 is computed by solving a d dimensional linear system of equations, which is easy and fast. Naively choosing a criterion for moving the vertices could have easily led to a nonlinear minimization that could not be solved quickly or reliably. Keeping Step 3 efficient is important to the overall speed of the algorithm.

Computing the Codomain Vertices

Step 3 of the MINVAR algorithm computes an updated codomain vertex as the average of the least squares affine approximations evaluated at the corresponding updated domain vertex. This average can be replaced with a weighted average,

$$\mathbf{q}_i^{(k+1)} = \frac{1}{t_i} \sum_{\bar{s}_j \in \mathrm{St}\{\mathbf{p}_i\}} w_j L_j \left(\mathbf{p}_i^{(k+1)}\right), \qquad t_i = \sum_{\bar{s}_j \in \mathrm{St}\{\mathbf{p}_i\}} w_j \tag{4.10}$$

The weights can be chosen in a variety of ways to affect the resulting approximation. For example, using $w_j = \mathrm{card}(\mathcal{Z}_j)$ will bias the resulting codomain vertex biased toward the least squares approximations corresponding to d-simplices that use more data. Other options include weighting by d-simplex volume (given by Claim 2.10) or by the inverse of the MSE of the least squares approximations. These weighting schemes have the greatest effect when the data is far from piecewise linear and so the final approximation fits the data set poorly in some locations.

4.2 Affinely Constrained Vertices

In the statement of MINVAR algorithm in Section 4.1, all vertices on the boundary of the domain $\mathbf{p}_i^{(k)} \in \partial \left|\mathcal{T}(P^{(k)}, \mathcal{S})\right|$ are held fixed through the iterations so that the domain of the PL function does not change from iteration to iteration. Requiring all boundary vertices to remain fixed is a stricter requirement than necessary, since a boundary vertex that lies on part of the boundary that is locally an affine subspace can move in that affine subspace without changing the domain. Such a vertex is called a *movable boundary vertex*. Only vertices that are "corners," i.e. extreme points, of the domain must remain fixed. For example, consider a PL function for which the domain is a cube, an apt example since experimental data is often taken over a hypercubic region. A boundary vertex that lies in a face of the cube can move around in that face, a boundary vertex that lies on an edge may move along the edge, but the vertices at the corners of the cube must stay fixed.

Experimentation with an early version of MINVAR in which all boundary vertices remained fixed led to several interesting observations. The first was that data points with the greatest absolute error generally occurred on the boundary of the domain. The second was that hand-tuning the location of the movable boundary vertices led to appreciable drops in the mean squared error of the resulting approximation. These observations motivated an extension of MINVAR to allow a vertex to move in an appropriately chosen affine subspace using a constrained version of Equation 4.2.

Let $\mathbf{p}_i$ be a domain vertex to be constrained to an l-dimensional affine subspace, and let $\bar{\mathbf{p}}_1, \ldots, \bar{\mathbf{p}}_{l+1} \in \mathbb{R}^d$ be affinely independent points whose affine hull is that desired affine subspace. Any point $\mathbf{x} \in \text{aff}\{\bar{\mathbf{p}}_1, \ldots, \bar{\mathbf{p}}_{l+1}\}$ may be represented by a point $\hat{\mathbf{x}} \in \mathbb{R}^l$ by the transformation

$$\mathbf{x} = \mathbf{C}\hat{\mathbf{x}} + \mathbf{d}$$

where

$$\begin{aligned} \mathbf{C} &= \begin{bmatrix} \bar{\mathbf{p}}_1 - \bar{\mathbf{p}}_{l+1} & \bar{\mathbf{p}}_2 - \bar{\mathbf{p}}_{l+1} & \cdots & \bar{\mathbf{p}}_l - \bar{\mathbf{p}}_{l+1} \end{bmatrix} \\ \mathbf{d} &= \bar{\mathbf{p}}_{l+1} \end{aligned}$$

This is a coordinatization of the affine subspace. Applying this coordinatization, the "min var" equation can be minimized over the affine subspace rather than all of $\mathbb{R}^d$. Claim B.2 proves that the solution to this constrained minimization is given by

$$\mathbf{p}_i^{(k)} = -\mathbf{C}{\mathbf{H}_{c,i}}^{-1}\mathbf{h}_{c,i} + \mathbf{d} \tag{4.11}$$

where

$$\mathbf{H}_{c,i} = \mathbf{C}^{\mathrm{T}}\mathbf{H}_i\mathbf{C}, \qquad \mathbf{h}_{c,i} = \mathbf{C}^{\mathrm{T}}\left(\mathbf{H}_i\mathbf{d} + \mathbf{h}_i\right), \tag{4.12}$$

and $\mathbf{H}_i$ and $\mathbf{h}_i$ are given by Equations 4.7 and 4.8. Note that this constrained minimization only guarantees that the vertex will remain on the chosen affine subspace, and does not guarantee that the vertex will stay on the boundary of the domain. Once again, the λ regularization term is used to prevent tangles.

4.3 Heuristically Guided Retriangulation

PL functions are parameterized by a triplet $(P, Q, \mathcal{S})$. Thus far in the presentation of the MINVAR algorithm, the combinatorial parameters $\mathcal{S}$ have remained fixed, while only the continuous parameters, the domain vertices P and the codomain vertices Q, have been used to approximate the function. The combinatorial parameters play an important role in the approximation. Figure 4.3 illustrates how two PL functions with the same continuous parameters but different combinatorial parameters can differ locally quite a bit. This sort of observation motivated the work by Dyn et al. [DLR90b, DLR90a], discussed in Section 3.2.1, on data dependent triangulations for interpolation.

In [TB97], Tourigny et al. propose an algorithm for computing discontinuous piecewise linear approximations to a data generating function (not just to data from that function). As an addition to their algorithm, they intermittently adapt the combinatorial structure of a triangulation using a data dependent criterion influenced by Dyn et al. The data dependent criterion in this case is a the error between the approximation and the data generating function, which in Tourigny's case is available directly to the algorithm, rather than through data. Like Dyn et al., Tourigny's work is specialized for functions from $\mathbb{R}^2$ to $\mathbb{R}$.

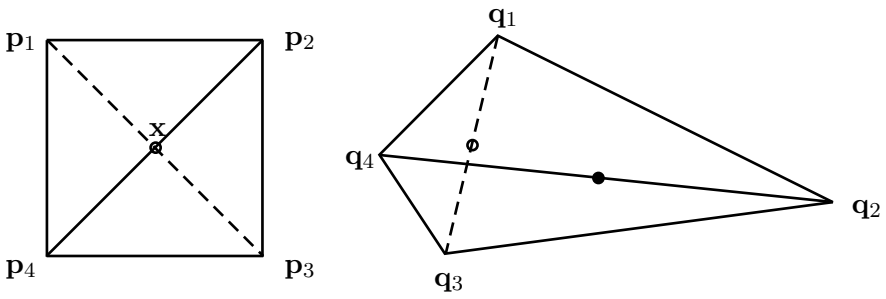

Figure 4.3: Effect of different triangulations on a PL function. Consider the two 2-dimensional PL functions f_1 and f_2 shown above (domain on the left, codomain on the right) parameterized by $(P, Q, \mathcal{S}_1)$ and $(P, Q, \mathcal{S}_2)$ respectively, where $P = \{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4\}$, $Q = \{\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3, \mathbf{q}_4\}$, and the d-simplices of $\mathcal{S}_1$ and $\mathcal{S}_2$ are given by $\{\{1, 2, 4\}, \{2, 3, 4\}\}$ and $\{\{1, 2, 3\}, \{1, 3, 4\}\}$, respectively. In the codomain, $f_1(\mathbf{x})$ is marked by the closed dot and $f_2(\mathbf{x})$ is marked by the open dot, illustrating that PL functions with the same continuous parameters can differ quite a bit due to their combinatorial structure.

Choosing an appropriate time to change the combinatorial structure for an algorithm such as MINVAR that iteratively changes the continuous parameters is tricky, due to the interaction between the combinatorial and continuous components. Consider two different abstract simplicial complexes $\mathcal{S}_1$ and $\mathcal{S}_2$ that each forms a valid triangulation of the domain with P_0, $\mathcal{T}(P_0, \mathcal{S}_1)$ and $\mathcal{T}(P_0, \mathcal{S}_2)$. If MINVAR is run with the initial conditions $\mathcal{P}_1 = (P_0, Q_0, \mathcal{S}_1)$ and $\mathcal{P}_2 = (P_0, Q_0, \mathcal{S}_2)$, then in general the converged continuous parameters obtained by iterating MINVAR, (P_1, Q_1) and (P_2, Q_2) respectively, will be different from each other, as verified in numerical experience. The combinatorial structure determines where the continuous parameters converge. Additionally, even if $\mathcal{T}(P_1, \mathcal{S}_1)$ and $\mathcal{T}(P_2, \mathcal{S}_2)$ are valid triangulations, it may be that $\mathcal{T}(P_1, \mathcal{S}_2)$ and $\mathcal{T}(P_2, \mathcal{S}_1)$ are not. Due to the interactions between the combinatorial and continuous parameters, it seems unwise to change the combinatorial component at each iteration, because these discrete (and hence discontinuous) changes can lead to poor overall behavior. Instead the combinatorial structure is modified periodically, with the period determined as an input parameter to the algorithm.

A significant difference between [DLR90b, DLR90a] and the present work is that MINVAR is an approximation problem rather than an interpolation problem, and thus more data is available to inform the choice of combinatorial parameters. This data could be used directly to select how to make local changes to the triangulation. For example, consider a flippable subcomplex. The least squares affine approximation for a fixed triangulation (as described in Section 3.3) could be computed for the original subcomplex and the flipped subcomplex, and the subcomplex with the lowest residual error would be chosen. This would be an interesting approach, but remains as future work. Rather the present heuristic method for retriangulating creates a data dependent triangulation based on the current domain and

codomain vertices using a generalization of the "jump in normal derivative" criterion. This criterion is computationally easy to compute, but a method that is better informed by the data will probably prove superior in the long run.

Several difficulties arise in generalizing data dependent triangulations to higher dimensions. For $d > 2$, the number of d-simplices in a triangulation is not constant. Flips, as described in Section 2.3.4, are performed between one subcomplex and another, and a subcomplex involves several d-simplices and facets as compared to two 2-simplices and one facet in the planar case. For example, a generic flip in $\mathbb{R}^3$ converts between two tetrahedra and three tetrahedra. The data dependent criterion must take this into account.

An analogue to the jump in normal derivative (JND) criterion proposed by Dyn et al. can be derived for a single facet in higher dimensions. Consider a PL function parameterized by $(P, Q, \mathcal{S})$ with d-simplices $\bar{s}_i$ and $\bar{s}_j$ that share a facet, and let $f_{\mathcal{P}}\big|_{\bar{s}_i}(\mathbf{x}) = \mathbf{A}_i\mathbf{x} + \mathbf{b}_i$ and $f_{\mathcal{P}}\big|_{\bar{s}_j}(\mathbf{x}) = \mathbf{A}_j\mathbf{x} + \mathbf{b}_j$. From Claim 3.1, $\mathbf{A}_i - \mathbf{A}_j$ is rank 1, and its rows will be scalar multiples of each other and normal to the shared facet. Thus, $\|\mathbf{A}_i - \mathbf{A}_j\|_F$, where $\|\cdot\|_F$ is the Frobenius norm, is an analogue to the JND criterion for a single facet in higher dimensions. In higher dimensions, however, multiple facets are simultaneously modified in a subcomplex flip, so an entire subcomplex, rather than just a single facet, must be evaluated. The generalization for the jump in normal derivative criterion for a subcomplex T is

$$\text{JND} = \max\left\{\|\mathbf{A}_i - \mathbf{A}_j\|_F \;\middle|\; \bar{s}_i, \bar{s}_j \text{ are } d\text{-simplices that share a facet in } T\right\}. \tag{4.13}$$

Variations on this generalization include taking an average rather than a maximum of the facet JNDs, and considering all facets of T, even those shared by d-simplices not in T. The author suspects that the latter criterion will eventually lead to a data dependent flipping algorithm that provably terminates, based on the proof techniques of Lawson [Law77].

The steps of MINVAR's heuristic retriangulation are as follows. First the algorithm proceeds facet by facet through the triangulation, looking for flippable subcomplexes as defined in Section 2.3.4. A subcomplex is flipped if it leads to a lower generalized JND as defined above. Checking all the facets in the triangulation completes one cycle of the retriangulation. Retriangulation is limited to a maximum number of cycles, since there is currently no proof that this process will terminate rather than continue indefinitely. In practice, this process generally terminates within ten cycles. The sequence in which facets are checked alters the resulting triangulation.

In [TB97], the authors suggest that "mesh tangles," i.e. when their algorithm moves the domain vertices such that the triangulation is no longer valid, be interpreted as a sign that the combinatorial structure of the PL function is not appropriate for approximation of the data generating function in this region. This is a nice intuitive argument, but unfortunately, very little is known for certain in this area. Periodic local retriangulation and also global retriangulation (when mesh tangles occur) are used in MINVAR to deal with these problems.

Vertices	uniform LS	MINVAR	MINVAR LS
2^2	1.79801e-01	1.79818e-01	1.79801e-01
3^2	9.85617e-02	4.56123e-02	4.48297e-02
4^2	4.45294e-02	1.92465e-02	1.89605e-02
5^2	2.47517e-02	1.38427e-02	1.36523e-02
6^2	1.55282e-02	7.35190e-03	7.25304e-03
7^2	1.05933e-02	5.40420e-03	5.34596e-03
8^2	7.63439e-03	4.63040e-03	4.56706e-03
9^2	5.84815e-03	3.38636e-03	3.34218e-03
10^2	4.53419e-03	2.96688e-03	2.92793e-03
11^2	3.71421e-03	2.50778e-03	2.47792e-03
12^2	2.99647e-03	2.34302e-03	2.32279e-03
13^2	2.49846e-03	1.86386e-03	1.84316e-03

Table 4.1: RMSE of approximations by uniform LS, MINVAR, and MINVAR LS.

There are many challenging open questions relating to the interaction of the continuous and discrete parameters in PL approximations.

4.4 A Numerical Example

This section presents an example of the MINVAR algorithm's performance on a "test function," $f : [0,1]^2 \to \mathbb{R}^2$, given by

$$f(x) = \begin{bmatrix} \tanh\frac{5}{8}\left(2x_1 - 4{x_2}^4 + 3{x_2}^2 - 1\right) \\ \tanh\frac{5}{8}\left(2{x_1}^2 - \ {x_1}^4 + 2x_2 - 1\right) \end{bmatrix}, \tag{4.14}$$

which is invertible over the domain $D = [0,1]^2$. The implementation of MINVAR constructs an approximation to a discrete set of data, and includes constrained motion of boundary vertices as well as data dependent retriangulation. Since the test function is neither piecewise linear nor directly available, Theorem 5.1 provides no performance guarantees, but good performance under these circumstances suggests MINVAR's broader applicability. Two sets of numerical studies are presented. The first examines the effects of varying the number of vertices in the PL approximation, and the second examines how data set size affects approximations with a fixed number of vertices.

The first set of experiments fits PL approximations of differing sizes to a single data set. The data set was generated by sampling the test function on an 80×80 uniform grid over the domain. For each $n = 2, \ldots, 13$, three different PL approximations with n^2 vertices were computed: i) the least squares continuous PL approximation on a fixed uniform triangulation of the domain (referred to as "uniform LS") ii) MINVAR, initialized

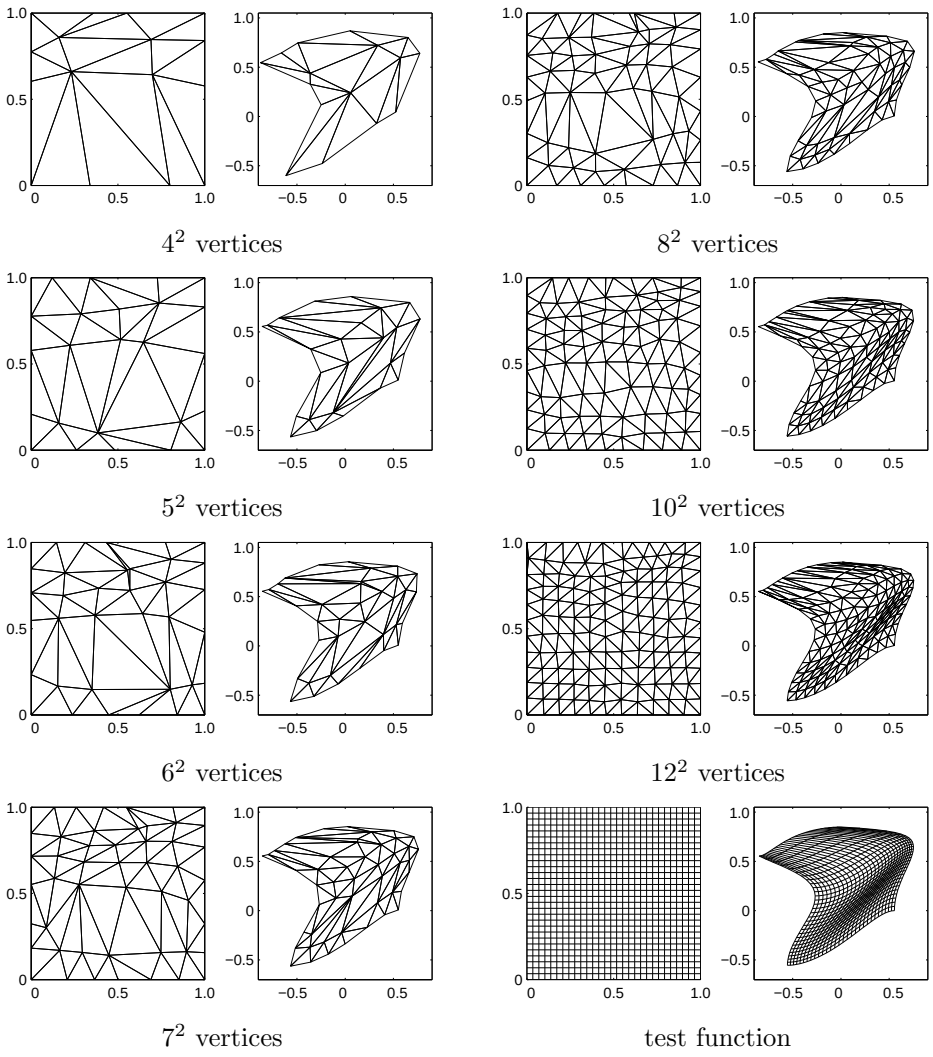

Figure 4.4: Visualization on MINVAR approximations to a test function. Visualizations of the test function (lower-right) and a series of PL approximations to the test function. In each subfigure, the domain is displayed on the left and the range on the right. Note that the approximations are invertible, since there are no tangles in the range triangulations.

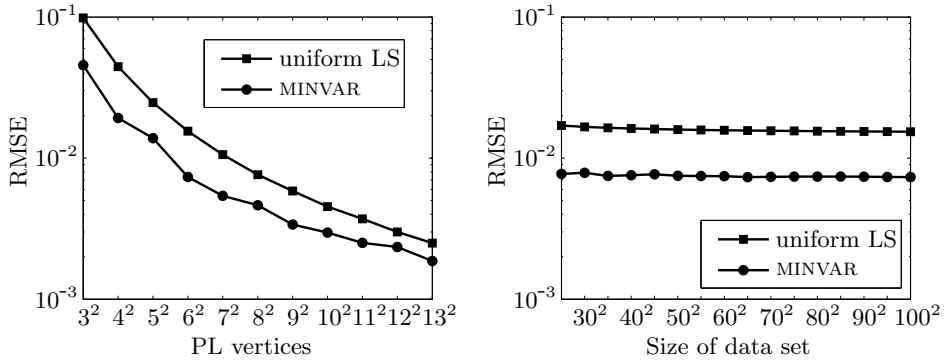

Figure 4.5: Comparison of MINVAR with least squares PL approximation on a uniform triangulation. (a) vertices in approximation vs. RMSE for approximations to the 80×80 data set. (b) density of training data set vs. RMSE on validation data for approximations with 6^2 vertices.

on a uniform triangulation of the domain (referred to as "MINVAR"), and iii) the least squares continuous PL approximation on the final triangulation from MINVAR (referred to as "MINVAR LS"). Recall that when the domain triangulation is fixed, the least squares continuous PL approximation problem becomes linear-in-parameters and the solution can be computed directly [Chu88, SP95]. Figure 4.4 shows the test function and several exemplars of the MINVAR approximations. Table 4.1 and Figure 4.5(a) show the root mean square error (RMSE) of the approximations as a function of the number of vertices. For 2^2 domain vertices, all of them are on the corners of the domain and must remain fixed, so MINVAR can only change the range vertices. Since MINVAR is not guaranteed to give the least squares continuous PL approximation for a given triangulation, it is not surprising that the uniform LS approximation's RMSE is slightly lower than MINVAR's in this case. The RMSE difference between MINVAR and MINVAR LS approximations is less than 2%. Least Squares could be applied as a post processing step to MINVAR, but since the differences are relatively small, this might not be necessary in application settings. From the triangulations of the MINVAR approximations in Figure 4.4, the domain triangulations move farther for lower numbers of vertices. As the number of vertices increases, the triangulations visually seem to deviate less from the initial uniform triangulation. The RMSE performance of MINVAR reflects this, giving the biggest reductions in RMSE as compared to uniform LS for triangulations with 3^2 to 6^2 vertices. Since this study uses initial conditions in which the vertices are on a uniform grid, approximations with large numbers of vertices may be getting caught in local minima near their initial conditions. In this case, performance could be improved by refining converged less dense approximations to create initial conditions for the dense approximations [TB97].

In the second set of experiments, PL approximations with 6^2 vertices were trained using

data sets of varying size. The PL approximations were chosen to have 6^2 vertices because, as mentioned above, this is in the region of sizes where the performance gains from MINVAR are greatest. The data sets were generated by sampling the test function on a uniform $n \times n$ grid, $n = 25, 30, 35, \ldots, 100$. The approximations were compared using a validation data set generated by evaluating the test function at 1000 points sampled from a uniform probability distribution over the domain. Figure 4.5(b) shows the validation set RMSE for MINVAR and uniform LS. With a 20×20 data set, MINVAR fails to run with a 6^2 vertex approximation, because as the vertices move, several simplices shrink to the point that they do not contain enough data to make the linear least squares approximation unique. Data sparsity is a serious issue in this type of local approximation.

From this example, MINVAR shows marked benefit when the approximation has relatively few vertices relative to the complexity of the test function. We expect that MINVAR's performance on higher order (more vertices) approximations could be improved by seeding initial conditions based on lower order approximations. The algorithm produces a consistent approximation to variously sized data sets, so long as there is enough data for it to run.

CHAPTER 5

A Local Convergence Proof for the MINVAR Algorithm

We now proceed to the central analytical result: a local convergence proof for the MINVAR algorithm[1]. The result is for the "approximation" version of the algorithm, as opposed to the "estimation" version presented in Chapter 4. The distinction is that in the "approximation" algorithm the data generating function is considered to be directly available in closed form, rather than through a set of discrete data. In this case, the least squares affine approximations from Step 2 of MINVAR become the best L_2 affine approximations. The best L_2 affine approximation is the orthogonal projection of the data generating function onto the subspace of affine functions. Since only Step 2 involves the data or data generating function, the approximation version may be viewed as the limit behavior of the estimation version when provided with an unbounded quantity of uniformly distributed data. Section B.2 discusses the relationship between computing the least squares and the best L_2 affine approximations.

Theorem 5.1 shows that, if the data generating function is a nondegenerate (to be defined below) PL function and the approximation is initialized "close enough" to the data generating function, then the MINVAR algorithm with $\lambda = 0$ will cause the vertices of the approximation to converge to the vertices of the data generating function. In this case, "close enough" means that the initial approximation shares the same combinatorial structure as the data generating function, and the domain vertices of the approximation start close to the corresponding vertices of the data generating function. As a corollary, the approximation will converge to the data generating function in the L_∞ sense, due to the continuity of a PL function in its continuous parameters, Claim 3.3. Examining MINVAR when $\lambda = 0$ admits a simpler proof while capturing the essence of the algorithm. Similar results could be obtained for $\lambda > 0$, though the convergence rate would be slower. An additional technical condition, that the data generating function be nondegenerate, is required when $\lambda = 0$ in order to guarantee existence of a unique solution to Equation 4.2, whereas for $\lambda > 0$ the regularized variance minimization in Equation 4.2 is guaranteed to have a unique solution.

A piecewise linear function $f_{\mathcal{P}}$ parameterized by $\mathcal{P} = (P, Q, \mathcal{S})$ is said to be *nondegen-*

[1] A version of this proof appears in [GKK03]

erate if for all $\mathbf{p}_i \in P$ such that $\mathbf{p}_i \notin \partial\,|\mathcal{T}(P,\mathcal{S})|$, the matrix $\mathbf{H}_i$ is full rank, where

$$\mathbf{H}_i = \left(\frac{1}{N_i} \sum_{\bar{s}_j \in \mathrm{St}\{\mathbf{p}_i\}} {\mathbf{A}_j}^{\mathrm{T}} \mathbf{A}_j \right) - \overline{\mathbf{A}_i}^{\mathrm{T}} \overline{\mathbf{A}_i}, \qquad \text{where } \overline{\mathbf{A}_i} = \frac{1}{N_i} \sum_{\bar{s}_j \in \mathrm{St}\{\mathbf{p}_i\}} \mathbf{A}_j.$$

Intuitively, nondegeneracy of $f_{\mathcal{P}}$ means that for any $\mathbf{p}_i \notin \partial\,|\mathcal{T}(P,\mathcal{S})|$, the affine functions that $f_{\mathcal{P}}$ takes in the d-simplices surrounding $\mathbf{p}_i$ are sufficiently "different" from one another. e.g. they are not all parallel.

In this chapter, as throughout this dissertation, vector norms are the standard Euclidean norm and matrix norms are the induced two norm, unless otherwise noted. With these facts and definitions in place, the result more formally may be stated more formally.

5.1 Statement of Local Convergence Theorem

Theorem 5.1. *Let $f^*_{\mathcal{P}}$ be a nondegenerate piecewise linear data generating function parameterized by $(P^*, Q^*, \mathcal{S}^*)$. Let*

$$\epsilon_0 = \min \left\{ \epsilon_d,\ \frac{1}{c_4} \right\}, \tag{5.1}$$

where ϵ_d and c_4 are given in Proposition 5.1. If for some $0 < \epsilon < \epsilon_0$ the initial piecewise linear approximation $f^{(0)}_P$ with parameterization $(P^{(0)}, Q^{(0)}, \mathcal{S}^*)$ satisfies $\left\|\mathbf{p}^{(0)}_i - \mathbf{p}^*_i\right\| < \epsilon$ for all i, then iteration of the MINVAR *algorithm with $\lambda = 0$ gives a sequence of approximations $f^{(k)}_P$ satisfying*

$$\begin{aligned} \lim_{k\to\infty} \left\|\mathbf{p}^{(k)}_i - \mathbf{p}^*_i\right\| x &= 0 \\ \lim_{k\to\infty} \left\|\mathbf{q}^{(k)}_i - \mathbf{q}^*_i\right\| &= 0 \end{aligned}$$

for all i.

Theorem 5.1 provides an immediate corollary.

Corollary 5.1. *Let $f^*_{\mathcal{P}}$ be a nondegenerate piecewise linear data generating function parameterized by $\mathcal{P}^* = (P^*, Q^*, \mathcal{S}^*)$. Let $\epsilon_0 = \epsilon_0(\mathcal{P}^*)$ be given by Equation 5.1. Let the initial approximation $f^{(0)}_{\mathcal{P}}$ be parameterized by $(P^{(0)}, Q^{(0)}, \mathcal{S}^*)$, satisfying for some $\epsilon < \epsilon_0$, $\left\|\mathbf{p}^{(0)}_i - \mathbf{p}^*_i\right\| < \epsilon$ for all i. Then application of the* MINVAR *algorithm with $\lambda = 0$ yields a sequence of approximations satisfying*

$$\lim_{j\to\infty} \left\| f^{(j)}_{\mathcal{P}} - f^*_{\mathcal{P}} \right\|_\infty = 0.$$

Proof. Theorem 5.1 shows that iteration of the MINVAR algorithm causes the vertices of the approximation to converge to the vertices of the data generating function. By Claim 3.3, a PL function is continuous in its vertices. The corollary follows directly. □

Theorem 5.1 follows readily Proposition 5.1. The statements and proofs of the propositions and lemmas follow in the next subsections, but first a short sketch of the structure of the proof is provided. The essence of Proposition 5.1 is that when the distances between the vertices of the approximation and the corresponding vertices of the data generating function are bounded by ϵ, then after one iteration of the MINVAR algorithm the distances will be bounded by a constant times ϵ^2. This result is established by applying two lemmas corresponding to the two stages of the algorithm. Lemma 5.1 proves that if the distances between corresponding vertices are bounded by ϵ, then the perturbation of the least squares affine map over a given simplex of the approximation from the affine map in the corresponding simplex of the data generating function is bounded by a constant times ϵ^2. Lemma 5.2 proves that if the perturbation of the least squares affine map over a simplex of the approximation from the affine map in the corresponding simplex of the data generating function is bounded by Δ, then the variance minimization will place the new vertices of the approximation such that the distance between them and the corresponding vertices of the data generating function are bounded by a constant times Δ. The combination of Lemma 5.1 and 5.2 provides Proposition 5.1.

The quadratic rate of convergence in Proposition 5.1 arises from the hypothesis that the data generating function is piecewise linear and close to the initial approximation. Without this assumption, Lemma 5.1 would fail to provide an ϵ^2 perturbation in the least squares affine approximations. In this case, the author suspects the convergence rate of the algorithm to be linear. Also, in the case where $\lambda > 0$, the convergence rate is suspected to slow to linear. Convergence may be slower on fine triangulations, but since this algorithm is intended primarily for use with a discrete set of data, the fineness of the triangulation is inherently limited by the amount of data provided. In applications, MINVAR can run triangulations of practical size in a few minutes.

5.2 Lemmas and Propositions

This section states the lemmas and propositions, while the proofs are provided in the following section.

Several recurring constants related to the geometry of the domain triangulation of the data generating function appear in the proof. Let $r_1 = r_1(P^*, \mathcal{S}^*)$ be the maximum inter-vertex distance, defined in Equation 2.19. Let $r_2 = r_2(P^*, \mathcal{S}^*)$ be the minimum orthogonal distance, defined in Equation 2.20. Let $r_3 = r_3(P^*, \mathcal{S}^*)$ be the dilation radius, defined in Equation 2.21.

The first lemma shows that if the domain vertices of the approximation are ϵ close to the domain vertices of the data generating function, then the least squares affine fit over a simplex $\bar{s}_i$ is a perturbation away from the affine function that the data generating function takes in $\bar{s}_i^*$. Moreover, the perturbation is quadratic in ϵ. We write $\Pi(f)$ to denote the L_2

orthogonal projection of the function f onto the space of affine functions, that is, the best L_2 affine approximation to f. The computation is shown in Section B.2.

Lemma 5.1. *Let $f^*_{\mathcal{P}}$ be a piecewise linear data generating function parameterized by $\mathcal{P}^* = (P^*, Q^*, \mathcal{S}^*)$. Consider a piecewise linear approximation $f_{\mathcal{P}}$ parameterized by $\mathcal{P} = (P, Q, \mathcal{S}^*)$. Let $\epsilon < \epsilon_c$, where*

$$\epsilon_c := \min\left\{\tfrac{1}{2(d+1)} r_2, \quad r_3, \quad 1\right\}. \tag{5.2}$$

Consider the simplices $\bar{s}^*_i$ and $\bar{s}_i$. Let $\mathbf{x}_c \in \bar{s}^*_i$. Let $f^*_{\mathcal{P}}|_{\bar{s}^*_i}(\mathbf{x}) = \mathbf{A}^*_i(\mathbf{x} - \mathbf{x}_c) + \mathbf{b}^*_i$. If $\left\|\mathbf{p}_j - \mathbf{p}^*_j\right\| < \epsilon$ for all $\mathbf{p}^*_j \in \bar{s}^*_i$, then the least squares approximation to $f^*_{\mathcal{P}}$ on $\bar{s}_i$, $\Pi(f^*_{\mathcal{P}}|_{\bar{s}_i})(\mathbf{x}) = \hat{\mathbf{A}}_i(\mathbf{x} - \mathbf{x}_c) + \hat{\mathbf{b}}_i$, satisfies the property

$$\left\| \begin{bmatrix} \hat{\mathbf{A}}_i{}^{\mathrm{T}} - \mathbf{A}^{*\mathrm{T}}_i \\ \hat{\mathbf{b}}_i{}^{\mathrm{T}} - \mathbf{b}^{*\mathrm{T}}_i \end{bmatrix} \right\|_2 < c_{1,i}\epsilon^2, \tag{5.3}$$

where $c_{1,i} = c_{1,i}(\mathcal{P}^*)$ is given by Equation 5.15.

The second lemma considers one set of affine functions that all intersect at a common point and another set of affine functions which are perturbations of the first set of functions. It is shown that performing the variance minimization, equivalent to Equation 4.2 with $\lambda = 0$, on the second set of functions generates a point whose distance from the intersection point is linear in the norm of the perturbations.

Lemma 5.2. *Let L^* be a set of N affine maps, $L^*_1, \ldots L^*_N$, such that all intersect at $(\mathbf{p}^*, \mathbf{q}^*)$ and are written as $L^*_i(x) = \mathbf{A}^*_i(\mathbf{x} - \mathbf{p}^*) + \mathbf{q}^*$, and such that $\mathbf{H}^*$, given by Equation 5.17, is full rank. Let L be a set of perturbed affine maps, $L_1, \ldots L_N$, expressed as $L_i(\mathbf{x}) = \hat{\mathbf{A}}_i(\mathbf{x} - \mathbf{p}^*) + \hat{\mathbf{q}}_i$, which satisfy the property*

$$\left\| \begin{bmatrix} \hat{\mathbf{A}}_i{}^{\mathrm{T}} - \mathbf{A}^{*\mathrm{T}}_i \\ \hat{\mathbf{q}}_i{}^{\mathrm{T}} - \mathbf{q}^{*\mathrm{T}} \end{bmatrix} \right\| < \Delta \tag{5.4}$$

for $\Delta < \Delta_0$, where $\Delta_0 = \Delta_0(\mathbf{A}^*_i, \mathbf{p}^*, \mathbf{q}^*)$ is given by Equation 5.16. Let $\mathbf{p}'$ and $\mathbf{q}'$ be given by

$$\begin{aligned} \mathbf{p}' &= \arg\min_x \quad \mathrm{var}\ L(x) \\ \mathbf{q}' &= \frac{1}{N}\sum_{i=1}^{N} L(\mathbf{p}'). \end{aligned}$$

Then $\mathbf{p}'$ and $\mathbf{q}'$ satisfy

$$\begin{aligned} \left\|\mathbf{p}' - \mathbf{p}^*\right\| &< c_2\Delta \\ \left\|\mathbf{q}' - \mathbf{q}^*\right\| &< c_3\Delta \end{aligned}$$

where $c_2 = c_2(\mathbf{A}^*_i, \mathbf{p}^*, \mathbf{q}^*)$ and $c_3 = c_3(\mathbf{A}^*_i, \mathbf{p}^*, \mathbf{q}^*)$ are given by Equation 5.20 and Equation 5.21.

The proposition brings the two lemmas together to show that a single step of the MINVAR algorithm induces a quadratic change in the distance of the approximation vertices to the data generating function vertices.

Proposition 5.1. *Let $f^*_{\mathcal{P}}$ be a nondegenerate piecewise linear data generating function parameterized by $\mathcal{P}^* = (P^*, Q^*, \mathcal{S}^*)$. Let $\epsilon < \epsilon_d$,*

$$\epsilon_d := \min\left\{\epsilon_c, \quad \sqrt{\frac{\Delta_0^m}{c_1}}\right\},$$

where $\epsilon_c = \epsilon_c(\mathcal{P}^*)$ is given by Equation 5.2, $\Delta_0^m = \Delta_0^m(\mathcal{P}^*)$ by Equation 5.22, and $c_1 = c_1(\mathcal{P}^*)$ by Equation 5.24.

If the piecewise linear approximation $f_{\mathcal{P}}$ parameterized by $(P, Q, \mathcal{S}^*)$ satisfies $\|\mathbf{p}_i - \mathbf{p}_i^*\| < \epsilon$ for all i, then one iteration of the MINVAR algorithm with $\lambda = 0$ gives the new approximation f'_P parameterized by $(P', Q', \mathcal{S}^*)$, which satisfies

$$\begin{aligned} \left\|\mathbf{p}_i' - \mathbf{p}_i^*\right\| &< c_4\epsilon^2 \\ \left\|\mathbf{q}_i' - \mathbf{q}_i^*\right\| &< c_5\epsilon^2 \end{aligned}$$

for all i, where $c_4 = c_4(\mathcal{P}^*)$ and $c_5 = c_5(\mathcal{P}^*)$ are given by Equation 5.28 and Equation 5.29.

The proof of Theorem 5.1 applies Proposition 5.1 to show that if the initial condition is close enough then iteration of the MINVAR algorithm causes convergence of the vertices of the approximation to the vertices of the data generating function.

5.3 Proofs of Lemmas, Proposition, and Theorem

This section presents proofs for the lemmas, proposition, and theorem introduced in the previous section.

Proof of Lemma 5.1.

Let $\varphi_i(\mathbf{x}) := \mathbf{A}_i^*(\mathbf{x} - \mathbf{x}_c) + \mathbf{b}_i^*$, the extension of $f^*_{\mathcal{P}}|_{\bar{s}_i^*}$ to $\mathbb{R}^d$. Let $\psi_i(\mathbf{x}) := f^*_{\mathcal{P}}(\mathbf{x}) - \varphi_i(\mathbf{x})$.

The orthogonal projection Π is a linear operator, and moreover for g affine, $\Pi(g) = g$. It follows that

$$\begin{aligned} \Pi(f^*_{\mathcal{P}}|_{\bar{s}_i}) &= \Pi(\varphi_i|_{\bar{s}_i}) + \Pi(\psi_i|_{\bar{s}_i}) \\ &= \varphi_i + \Pi(\psi_i|_{\bar{s}_i}). \end{aligned} \tag{5.5}$$

Let $\hat{\mathbf{A}}_i$ and $\hat{\mathbf{b}}_i$ be such that $\Pi(f^*_{\mathcal{P}}|_{\bar{s}_i})(\mathbf{x}) = \hat{\mathbf{A}}_i(\mathbf{x}-\mathbf{x}_c)+\hat{\mathbf{b}}_i$. Then from Equation 5.5 it follows that $\Pi(\psi_i|_{\bar{s}_i})(\mathbf{x}) = (\hat{\mathbf{A}}_i-\mathbf{A}_i^*)(\mathbf{x}-\mathbf{x}_c)+(\hat{\mathbf{b}}_i-\mathbf{b}_i^*)$. Moreover, since $\Pi(\psi_i|_{\bar{s}_i}) = \Pi(f^*_{\mathcal{P}}|_{\bar{s}_i}-\varphi_i|_{\bar{s}_i})$, thus $(\hat{\mathbf{A}}_i-\mathbf{A}_i^*)$ and $(\hat{\mathbf{b}}_i-\mathbf{b}_i^*)$ can be computed using the formula for L_2 orthogonal projection of $f^*_{\mathcal{P}}|_{\bar{s}_i} - \varphi_i|_{\bar{s}_i}$, (See Appendix B.2)

$$\begin{bmatrix} (\hat{\mathbf{A}}_i - \mathbf{A}_i^*)^{\mathrm{T}} \\ (\hat{\mathbf{b}}_i - \mathbf{b}_i^*)^{\mathrm{T}} \end{bmatrix} = {\mathbf{S}_{xx,i}}^{-1}\mathbf{S}_{xy,i},$$

$$\mathbf{S}_{xx,i} = \int_{\bar{s}_i} \begin{bmatrix} \mathbf{x} - \mathbf{x}_c \\ 1 \end{bmatrix} \begin{bmatrix} \mathbf{x}^{\mathrm{T}} - \mathbf{x}_c{}^{\mathrm{T}} & 1 \end{bmatrix} d\mathbf{x}, \qquad \mathbf{S}_{xy,i} = \int_{\bar{s}_i} \begin{bmatrix} \mathbf{x} - \mathbf{x}_c \\ 1 \end{bmatrix} \left(f^*_{\mathcal{P}}(\mathbf{x}) - \varphi_i(\mathbf{x})\right)^{\mathrm{T}} d\mathbf{x}.$$

The submultiplicative property holds for the induced two norm,

$$\left\| \begin{bmatrix} (\hat{\mathbf{A}}_i - \mathbf{A}_i^*)^{\mathrm{T}} \\ (\hat{\mathbf{b}}_i - \mathbf{b}_i^*)^{\mathrm{T}} \end{bmatrix} \right\| \leq \left\| \mathbf{S}_{xx,i}{}^{-1} \right\| \left\| \mathbf{S}_{xy,i} \right\|,$$

so on $\left\|\mathbf{S}_{xx,i}{}^{-1}\right\|$ and $\|\mathbf{S}_{xy,i}\|$ may be independently established. We will proceed to bound $\left\|\mathbf{S}_{xx,i}{}^{-1}\right\|$. Since $\bar{s}_i$ is a d-simplex, it follows from calculus and the definition of $\mathbf{S}_{xx,i}$ that $\mathbf{S}_{xx,i}$ is a positive definite matrix. Let $\mathbf{M}_i^*$ be given by

$$\mathbf{M}_i^* := \int_{\mathcal{D}(\bar{s}_i^*, -\epsilon_c)} \begin{bmatrix} \mathbf{x} - \mathbf{x}_c \\ 1 \end{bmatrix} \begin{bmatrix} \mathbf{x}^{\mathrm{T}} - \mathbf{x}_c{}^{\mathrm{T}} & 1 \end{bmatrix} d\mathbf{x}$$

Since $0 < \epsilon < \epsilon_c$ by hypothesis and $\epsilon_c \leq \frac{1}{2(d+1)} r_2$ by definition, it follows from Claim 2.11 that $\mathcal{D}(\bar{s}_i^*, -\epsilon_c)$ is a d-simplex and thus has non-empty interior. It follows that $\mathbf{M}_i^*$ is positive definite, and hence invertible. Since $\epsilon < \epsilon_c$, and each vertex of $\bar{s}_i$ is less than ϵ away from the corresponding vertex of $\bar{s}_i^*$, it follows that $\mathcal{D}(\bar{s}_i^*, -\epsilon_c) \subseteq \bar{s}_i$. Thus, $\mathbf{x}^{\mathrm{T}} \mathbf{M}_i^* \mathbf{x} < \mathbf{x}^{\mathrm{T}} \mathbf{S}_{xx,i} \mathbf{x}$ for all $\mathbf{x} \in \mathbb{R}^d$, which implies that $\lambda_{\min}(\mathbf{M}_i^*) < \lambda_{\min}(\mathbf{S}_{xx,i})$. This provides the bound on $\left\|\mathbf{S}_{xx,i}{}^{-1}\right\|$,

$$\left\| \mathbf{S}_{xx,i}{}^{-1} \right\| < \left\| \mathbf{M}_i^{*-1} \right\|.$$

Now we proceed to $\|\mathbf{S}_{xy,i}\|$. By the properties of norms,

$$\|\mathbf{S}_{xy,i}\| \quad \leq \quad \int_{\bar{s}_i} \left\| \begin{bmatrix} \mathbf{x} - \mathbf{x}_c \\ 1 \end{bmatrix} \right\| \|\psi_i(\mathbf{x})\| \, d\mathbf{x} \tag{5.6}$$

Let $\mathcal{C}(\bar{s}_i, \epsilon) = \left\{ \mathbf{x} \in \mathbb{R}^d \middle| \delta(\mathbf{x}, \bar{s}_i) \leq \epsilon \right\}$. (Recall that $\delta(\cdot, \cdot)$ is the distance between a point and a set defined in Chapter 2.) By hypothesis, the vertices of $\bar{s}_i$ are less than ϵ away from the vertices of $\bar{s}_i^*$, so $\mathrm{vert}(\bar{s}_i) \subseteq \mathcal{C}(\bar{s}_i^*, \epsilon)$. Moreover, since $\mathcal{C}(\bar{s}_i^*, \epsilon)$ is convex and contains the extreme points of $\bar{s}_i$, $\bar{s}_i \subseteq \mathcal{C}(\bar{s}_i^*, \epsilon)$. The integrand in Equation 5.6 is nonnegative definite, so Equation 5.6 is bounded by

$$\leq \quad \int_{\mathcal{C}(\bar{s}_i^*, \epsilon)} \left\| \begin{bmatrix} \mathbf{x} - \mathbf{x}_c \\ 1 \end{bmatrix} \right\| \|(\psi_i(\mathbf{x}))\| \, d\mathbf{x}$$

By hypothesis $\mathbf{x}_c \in \bar{s}_i^*$. By the definition of r_1 and since $\epsilon < \epsilon_c$, it follows that $\forall \mathbf{x} \in \mathcal{C}(\bar{s}_i^*, \epsilon)$, $\|\mathbf{x} - \mathbf{x}_c\| \leq \bar{r} := r_1 + 2\epsilon_c$. Thus, the integral above is further bounded by

$$\leq \sqrt{1 + \bar{r}^2} \int_{\mathcal{C}(\bar{s}_i^*, \epsilon)} \|(\psi_i(x))\| \, dx \tag{5.7}$$

Once again the integrand is nonnegative definite, so Equation 5.7 can be bounded by integrating over $\mathcal{D}(\bar{s}_i^*, \epsilon)$, since $\mathcal{C}(\bar{s}_i^*, \epsilon) \subseteq \mathcal{D}(\bar{s}_i^*, \epsilon)$,

$$\leq \quad \sqrt{1 + \bar{r}^2} \textstyle\int_{\mathcal{D}(\bar{s}_i^*, \epsilon)} \|(\psi_i(\mathbf{x}))\| \, d\mathbf{x} \tag{5.8}$$

$$= \quad \sqrt{1 + \bar{r}^2} \textstyle\sum_{j=1}^{N} \int_{U_j} \|(\psi_i(\mathbf{x}))\| \, d\mathbf{x} \tag{5.9}$$

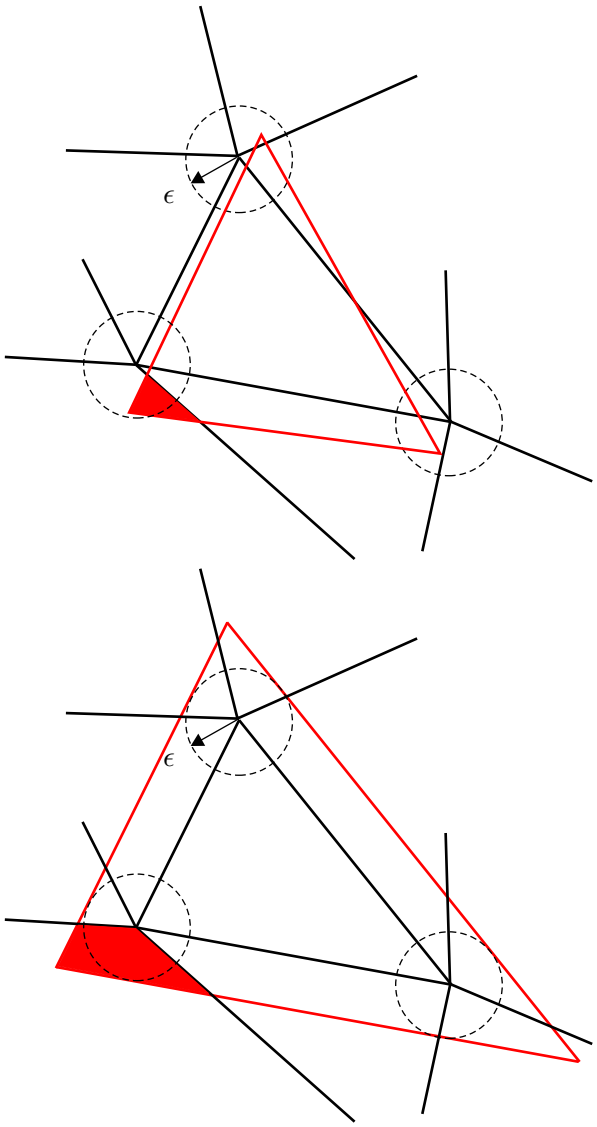

Figure 5.1: In these two figures illustrating Lemma 5.1, the domain triangulation of the data generating function is indicated with thick, dark lines. *(Top)* The d-simplex of the approximation, $\bar{s}_i$, (light colored line) does not match up exactly with the corresponding d-simplex of the data generating function, $\bar{s}_i^*$, but all the vertices of the approximation are within ϵ of the vertices of the data generating function. Any intersection of $\bar{s}_i$ with a simplex of the data generating function other than $\bar{s}_i^*$ perturbs the least squares affine approximation over $\bar{s}_i$ away from the data generating function's affine map over $\bar{s}_i^*$. The shaded patch shows one such perturbation causing region. *(Bottom)* To bound the perturbation, the mismatch is overestimated using the dilation, $\mathcal{D}(\bar{s}_i^*, \epsilon)$, (light colored line). The dilation is represented in terms of barycentric coordinates, whose interpretation as distances allows the appropriate bounds to be constructed. A difficult part of the bounding argument deals with the differences caused by the dimension of $\dim(\bar{s}_i^* \cap \bar{s}_j^*)$, where $\bar{s}_j^* \cap \bar{s}_i^* \neq \emptyset$.

where $U_j = \mathcal{D}(\bar{s}_i^*, \epsilon) \cap \bar{s}_j^*$ and N is the total number of d-simplices in the domain triangulation. Since $\psi_i(\mathbf{x}) = 0$ on $\bar{s}_i^*$, the term corresponding to $j = i$ in Equation 5.9 is zero. By hypothesis $\epsilon < \epsilon_c \leq r_3$, so by Claim 2.15, $\bar{s}_j^* \cap \mathcal{D}(\bar{s}_i^*, \epsilon) \neq \emptyset$ if and only if $\bar{s}_j^* \in \operatorname{St} \bar{s}_i^*$. Thus the terms in Equation 5.9 are only nonzero for j such that $\bar{s}_j^*$ is incident to $\bar{s}_i^*$. Consider such a term,

$$\int_{U_j} \|\psi_i(\mathbf{x})\| \, d\mathbf{x} = \int_{U_j} \left\| \left(\mathbf{A}_j^* - \mathbf{A}_i^*\right)(\mathbf{x} - \mathbf{x}_c) + \mathbf{b}_j^* - \mathbf{b}_i^* \right\| d\mathbf{x},$$

where $f_{\mathcal{P}}^*|_{\bar{s}_j^*}(\mathbf{x}) = \mathbf{A}_j^*(\mathbf{x} - \mathbf{x}_c) + \mathbf{b}_j^*$. By Claim 3.1, there exists $\mathbf{x}_O \in \bar{s}_j^* \cap \bar{s}_i^* \subseteq U_j$ such that $\left(\mathbf{A}_j^* - \mathbf{A}_i^*\right)(\mathbf{x}_O - \mathbf{x}_c) + \mathbf{b}_j^* - \mathbf{b}_i^* = 0$. Applying the change of coordinates $\mathbf{y} = \mathbf{x} - \mathbf{x}_O$ gives

$$\int_{U_j} \|\psi_i(\mathbf{x})\| \, d\mathbf{x} = \int_{U_j - \mathbf{x}_O} \left\| \left(\mathbf{A}_j^* - \mathbf{A}_i^*\right)\mathbf{y} \right\| \, d\mathbf{y} \tag{5.10}$$

Let L be the linear subspace parallel to $\operatorname{aff}(\bar{s}_i^* \cap \bar{s}_j^*)$. Recall that $\dim L = \dim \bar{s}_i^* \cap \bar{s}_j^* := d_{i,j}$ (see Section 2.2.3). By Claim 3.1, $L \subseteq \mathcal{N}(\left(\mathbf{A}_j^* - \mathbf{A}_i^*\right))$. Let $\mathbf{v}_1, \ldots, \mathbf{v}_{d_{i,j}}$ be an orthonormal basis for L. Let $\mathbf{v}_{d_{i,j}+1}, \ldots, \mathbf{v}_d$ be an orthonormal basis for $L^\perp$. Then $\mathbf{P} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_d \end{bmatrix}$ is an orthogonal matrix. Rewrite Equation 5.10 under the change of coordinates $\mathbf{z} = \mathbf{P}^{\mathrm{T}}\mathbf{y}$,

$$= \int_{\mathbf{P}^{\mathrm{T}}(U_j - \mathbf{x}_O)} \left\| \left(\mathbf{A}_j^* - \mathbf{A}_i^*\right)\mathbf{P}\mathbf{z} \right\| \, d\mathbf{z} \tag{5.11}$$

Since the integrand is a nonnegative definite function, we may bound Equation 5.11 by increasing the volume over which the integrand is integrated. By Claim 2.12, there exists $\kappa_{i,j}$ such that for all $\mathbf{x} \in U_j$, $\delta\Big(\mathbf{x}, \operatorname{aff}(\bar{s}_i^* \cap \bar{s}_j^*)\Big) < \kappa_{i,j}\epsilon$. Equivalently $\delta(\mathbf{y}, L) < \kappa_{i,j}\epsilon$ for any $\mathbf{y} \in U_j - \mathbf{x}_O$, from which it follows that the projection of $\mathbf{y}$ onto $L^\perp$ must have magnitude less than $\kappa_{i,j}\epsilon$. Moreover, by the definition of $\bar{r}$, the projection of y onto L must have magnitude less than $\bar{r}$. It follows that $\mathbf{P}^{\mathrm{T}}(U_j - \mathbf{x}_O) \subseteq [-\bar{r}, \bar{r}]^{d_{i,j}} \times B_{\bar{d}_{i,j}}(\kappa_{i,j}\epsilon)$, where $\bar{d}_{i,j} = d - d_{i,j}$, $d_{i,j}$ is defined in Equation 2.17, and $B_{\bar{d}_{i,j}}(\kappa_{i,j}\epsilon)$ is the $\bar{d}_{i,j}$-dimensional ball of radius $\kappa_{i,j}\epsilon$. Then Equation 5.11 is bounded by

$$\leq \int_{[-\bar{r},\bar{r}]^{d_{i,j}} \times B_{\bar{d}_{i,j}}(\kappa_{i,j}\epsilon)} \left\| \left(\mathbf{A}_j^* - \mathbf{A}_i^*\right)\mathbf{P}\mathbf{z} \right\| \, d\mathbf{z} \tag{5.12}$$

The first $d_{i,j}$ columns of $\left(\mathbf{A}_j^* - \mathbf{A}_i^*\right)\mathbf{P}$ are zero, since the first $d_{i,j}$ columns of $\mathbf{P}$ are in the nullspace of $\left(\mathbf{A}_j^* - \mathbf{A}_i^*\right)$. Thus, the integrand in Equation 5.12 has no dependence on $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_{d_{i,j}}$, so we can integrate through for $\mathbf{z}_1, \ldots, \mathbf{z}_{d_{i,j}}$, giving

$$= (2\bar{r})^{d_{i,j}} \int_{B_{\bar{d}_{i,j}}(\kappa_{i,j}\epsilon)} \left\| \left(\mathbf{A}_j^* - \mathbf{A}_i^*\right)\mathbf{P} \begin{bmatrix} \mathbf{I} \\ \mathbf{0} \end{bmatrix} \bar{\mathbf{z}}_1 \right\| \, dz_{d_{i,i}+1} \ldots dz_d \tag{5.13}$$

where ${\bar{\mathbf{z}}_1}^{\mathrm{T}} = \begin{bmatrix} z_{d_{i,j}+1} & \cdots & z_d \end{bmatrix}^{\mathrm{T}}$. Since the first $d_{i,j}$ columns of $\left(\mathbf{A}_j^* - \mathbf{A}_i^*\right)\mathbf{P}$ are zero and $\mathbf{P}$ is orthogonal, it follows that $\left\| \left(\mathbf{A}_j^* - \mathbf{A}_i^*\right)\mathbf{P} \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix}^{\mathrm{T}} \right\| \leq \left\| \mathbf{A}_j^* - \mathbf{A}_i^* \right\|$. Thus, Equation

5.13 can be further bound by

$$\leq (2\bar{r})^{d_{i,j}} \left\|\mathbf{A}_j^* - \mathbf{A}_i^*\right\| \int_{B_{\bar{d}_{i,j}}(\kappa_{i,j}\epsilon)} \|\bar{\mathbf{z}}_1\| \; d\bar{\mathbf{z}}_1 \tag{5.14}$$

From calculus (see for example [Spi65]) it can be shown that[2]

$$\int_{B_k(\epsilon)} \|\mathbf{w}\| \; d\mathbf{w} = \frac{k}{k+1} \frac{\pi^{k/2}}{\Gamma(k/2)} \epsilon^{k+1}$$

Applying this with Equation 5.14 to Equation 5.9, and then simplifying using the fact that $\epsilon^m \leq \epsilon^2$ for $m \geq 2$ since $\epsilon < \epsilon_c \leq 1$, gives

$$\|\mathbf{S}_{xy,i}\| \leq \sqrt{1+\bar{r}^2}\; l_i \max_{\substack{j \text{ s.t. } j \neq i, \\ \bar{s}_j^* \in \mathrm{St}\bar{s}_i^*}} \left((2\bar{r})^{d_{i,j}} \left\|\mathbf{A}_j^* - \mathbf{A}_i^*\right\| {\kappa_{i,j}}^{1+\bar{d}_{i,j}} \frac{\bar{d}_{i,j}}{\bar{d}_{i,j}+1} \frac{\pi^{(\bar{d}_{i,j})/2}}{\Gamma((\bar{d}_{i,j})/2)} \right) \epsilon^2$$

where $l_i = \sum_{\bar{s}_j^* \in \mathrm{St}\bar{s}_i^*} 1$. Then

$$\begin{aligned} \left\| \begin{bmatrix} {\hat{\mathbf{A}}_i}^{\mathrm{T}} - {\mathbf{A}_i^*}^{\mathrm{T}} \\ {\hat{\mathbf{b}}_i}^{\mathrm{T}} - {\mathbf{b}_i^*}^{\mathrm{T}} \end{bmatrix} \right\| &\leq \left\|{\mathbf{S}_{xx,i}}^{-1}\right\| \|\mathbf{S}_{xy,i}\| \\ &< c_{1,i}\epsilon^2 \end{aligned}$$

where

$$c_{1,i} := \left\|\mathbf{M}_i^{*-1}\right\| \sqrt{1+\bar{r}^2}\; l_i \max_{\substack{j \text{ s.t. } j \neq i, \\ \bar{s}_j^* \in \mathrm{St}\bar{s}_i^*}} \left((2\bar{r})^{d_{i,j}} \left\|\mathbf{A}_j^* - \mathbf{A}_i^*\right\| {\kappa_{i,j}}^{1+\bar{d}_{i,j}} \frac{\bar{d}_{i,j}}{\bar{d}_{i,j}+1} \frac{\pi^{(\bar{d}_{i,j})/2}}{\Gamma((\bar{d}_{i,j})/2)} \right) \tag{5.15}$$

and $\bar{d}_{i,j} = d - d_{i,j}$, $l_i = \sum_{\bar{s}_j^* \in \mathrm{St}\bar{s}_i^*} 1$, and $\bar{r} = r_1 + 2\epsilon_c$. □

Proof of Lemma 5.2.

Let the constant Δ_0 from the statement of the lemma be given by

$$\Delta_0 := \min \left\{ \frac{1}{N} \sum_{j=1}^{N} \left\|\mathbf{A}_j^*\right\| \quad , \quad \left(\frac{12}{N} \left\|\mathbf{H}^{*-1}\right\| \sum_{j=1}^{N} \left\|\mathbf{A}_j^*\right\| \right)^{-1} \right\}, \tag{5.16}$$

where $\mathbf{H}^*$ is given by Equation 5.17.

Solving $\mathbf{p}' = \arg\min_{\mathbf{x}} \text{var}\; L(\mathbf{x})$ is equivalent to solving Equation 4.2 with $\lambda = 0$. As with Equation 4.2, a closed form expression for p' can be found by "completing the square." Specifically, $p' = \mathbf{H}^{-1}h$,

$$\begin{aligned} \mathbf{H} &:= \left(\frac{1}{N} \sum_{j=1}^{N} {\hat{\mathbf{A}}_j}^{\mathrm{T}} \hat{\mathbf{A}}_j \right) - \overline{\hat{\mathbf{A}}}^{\mathrm{T}} \overline{\hat{\mathbf{A}}} \\ \mathbf{h} &:= \left(\frac{1}{N} \sum_{j=1}^{N} {\hat{\mathbf{A}}_j}^{\mathrm{T}} (\hat{\mathbf{q}}_j - \hat{\mathbf{A}}_j \mathbf{p}^*) \right) - \overline{\hat{\mathbf{A}}}^{\mathrm{T}} \overline{\hat{\mathbf{b}}} \end{aligned}$$

[2]For even k, $\Gamma(k/2) = (k/2)!$. For odd k, let $k' = \frac{1}{2}(k-1)$, then $\Gamma(k/2) = \Gamma(\frac{1}{2}+k') = \sqrt{\pi}\frac{(2k'+2)!}{(k+1)!4^{k'+1}}$.

where $\overline{\hat{\mathbf{A}}} = \frac{1}{N}\sum_{j=1}^{N}\hat{\mathbf{A}}_j$ and $\overline{\hat{\mathbf{b}}} = \frac{1}{N}\sum_{j=1}^{N}(\hat{\mathbf{q}}_j - \hat{\mathbf{A}}_j\mathbf{p}^*)$. Let $\tilde{\mathbf{A}}_j = \hat{\mathbf{A}}_j - \mathbf{A}_j^*$ and $\tilde{\mathbf{q}}_j = \hat{\mathbf{q}}_j - \mathbf{q}^*$. Then $\mathbf{H} = \mathbf{H}^* + \tilde{\mathbf{H}}$ and $\mathbf{h} = \mathbf{h}^* + \tilde{\mathbf{h}}$, with

$$\begin{aligned}\mathbf{H}^* &:= \left(\frac{1}{N}\sum_{j=1}^{N}\mathbf{A}_j^{*\mathrm{T}}\mathbf{A}_j^*\right) - \overline{\mathbf{A}^*}^{\mathrm{T}}\overline{\mathbf{A}^*}, \\ \mathbf{h}^* &:= \left(\frac{1}{N}\sum_{j=1}^{N}\mathbf{A}_j^{*\mathrm{T}}(\mathbf{q}^* - \mathbf{A}_j^*\mathbf{p}^*)\right) - \overline{\mathbf{A}^*}^{\mathrm{T}}.\end{aligned} \tag{5.17}$$

$$\begin{aligned}\tilde{\mathbf{H}} &= \frac{1}{N}\left[\sum_{j=1}^{N}(\mathbf{A}_j^* + \tilde{\mathbf{A}}_j)^{\mathrm{T}}\tilde{\mathbf{A}}_j + \sum_{j=1}^{N}\tilde{\mathbf{A}}_j{}^{\mathrm{T}}\mathbf{A}_j^*\right] - \overline{\mathbf{A}^*}^{\mathrm{T}}\overline{\tilde{\mathbf{A}}} - \overline{\tilde{\mathbf{A}}}^{\mathrm{T}}\overline{\mathbf{A}^*} - \overline{\tilde{\mathbf{A}}}^{\mathrm{T}}\overline{\tilde{\mathbf{A}}}, \\ \tilde{\mathbf{h}} &= \frac{1}{N}\left[\sum_{j=1}^{N}(\mathbf{A}_j^* + \tilde{\mathbf{A}}_j)^{\mathrm{T}}(\tilde{\mathbf{q}}_j - \tilde{\mathbf{A}}_j\mathbf{p}^*) + \sum_{j=1}^{N}\tilde{\mathbf{A}}_j{}^{\mathrm{T}}(\mathbf{q}^* - \mathbf{A}_j^*\mathbf{p}^*)\right] - \overline{\mathbf{A}^*}^{\mathrm{T}}\overline{\tilde{\mathbf{b}}} - \overline{\tilde{\mathbf{A}}}^{\mathrm{T}}\overline{\mathbf{b}^*} - \overline{\tilde{\mathbf{A}}}^{\mathrm{T}}\overline{\tilde{\mathbf{b}}}.\end{aligned}$$

where

$$\overline{\mathbf{A}^*} = \frac{1}{N}\sum_{j=1}^{N}\mathbf{A}_j^* \qquad \overline{\mathbf{b}^*} = \frac{1}{N}\sum_{j=1}^{N}(\mathbf{q}^* - \mathbf{A}_j^*\mathbf{p}^*)$$
$$\overline{\tilde{\mathbf{A}}} = \frac{1}{N}\sum_{j=1}^{N}\tilde{\mathbf{A}}_j \qquad \overline{\tilde{\mathbf{b}}} = \frac{1}{N}\sum_{j=1}^{N}(\tilde{\mathbf{q}}_j - \tilde{\mathbf{A}}_j\mathbf{p}^*)$$

Notice that $\mathbf{H}^*$ and $\mathbf{h}^*$ depend only on $\mathbf{A}_j^*$, $\mathbf{p}^*$ and $\mathbf{q}^*$. Moreover, since all functions in L^* go through $(\mathbf{p}^*, \mathbf{q}^*)$, it must be that $\mathbf{p}^* = \arg\min_{\mathbf{x}} \mathrm{var}\ L^*(\mathbf{x})$, and thus $\mathbf{p}^* = \mathbf{H}^{*-1}\mathbf{h}^*$. Rewriting $\mathbf{p}' = \mathbf{H}^{-1}\mathbf{h}$, gives

$$\left(\mathbf{H}^* + \tilde{\mathbf{H}}\right)\left(\mathbf{p}^* + (\mathbf{p}' - \mathbf{p}^*)\right) = \mathbf{h}^* + \tilde{\mathbf{h}}.$$

Applying $\mathbf{H}^*\mathbf{p}^* = \mathbf{h}^*$ and solving for $\mathbf{p}' - \mathbf{p}^*$ yields

$$\mathbf{p}' - \mathbf{p}^* = \left(\mathbf{H}^* + \tilde{\mathbf{H}}\right)^{-1}\left(\tilde{\mathbf{h}} - \tilde{\mathbf{H}}\mathbf{p}^*\right)$$

From the hypothesis it follows that $\left\|\tilde{\mathbf{A}}_j\right\| < \Delta$ and $\|\tilde{\mathbf{q}}_j\| < \Delta$. Applying these bounds and the properties of norms, it follows after some computation that

$$\begin{aligned}\left\|\tilde{\mathbf{H}}\right\| &\leq 2\Delta^2 + \frac{4\Delta}{N}\sum_{j=1}^{N}\left\|\mathbf{A}_j^*\right\|, \\ \left\|\tilde{\mathbf{h}}\right\| &\leq 2\left(1 + \|\mathbf{p}^*\|\right)\Delta^2 + \frac{2\Delta}{N}\sum_{j=1}^{N}\left[\left\|\mathbf{A}_j^*\right\|\left(1 + \|\mathbf{p}^*\|\right) + \|\mathbf{q}^*\| + \left\|\mathbf{A}_j^*\right\|\|\mathbf{p}^*\|\right].\end{aligned}$$

Since $\Delta < \Delta_0$ and by definition $\Delta_0 \leq \frac{1}{N}\sum_{j=1}^{N}\left\|\mathbf{A}_j^*\right\|$, the above bounds may be further simplified to

$$\left\|\tilde{\mathbf{H}}\right\| < \left(\frac{6}{N}\sum_{j=1}^{N}\left\|\mathbf{A}_j^*\right\|\right)\Delta \tag{5.18}$$

$$\left\|\tilde{\mathbf{h}}\right\| < \left(\frac{2}{N}\sum_{j=1}^{N}\left[2\left\|\mathbf{A}_j^*\right\|\left(1+\left\|\mathbf{p}^*\right\|\right)+\left\|\mathbf{q}^*\right\|+\left\|\mathbf{A}_j^*\right\|\left\|\mathbf{p}^*\right\|\right]\right)\Delta. \tag{5.19}$$

Also by definition $\Delta_0 \leq \left(\frac{12}{N}\left\|\mathbf{H}^{*-1}\right\|\sum_{j=1}^{N}\left\|\mathbf{A}_j^*\right\|\right)^{-1}$, so the bound in Equation 5.18 can be simplified to $\left\|\tilde{\mathbf{H}}\right\| < \frac{1}{2\|\mathbf{H}^{*-1}\|}$, and thus $\left\|\mathbf{H}^{*-1}\tilde{\mathbf{H}}\right\| < \frac{1}{2}$. From [GV96, Lemma 2.3.3], if $\mathbf{M}\in\mathbb{R}^{n\times n}$ and $\|\mathbf{M}\|<1$, then $\mathbf{I}-\mathbf{M}$ is nonsingular and $\left\|(\mathbf{I}-\mathbf{M})^{-1}\right\| \leq \frac{1}{1-\|\mathbf{M}\|}$. Applying this fact and the bound on $\left\|\mathbf{H}^{*-1}\tilde{\mathbf{H}}\right\|$, we derive after some computation

$$\left\|(\mathbf{H}^*+\tilde{\mathbf{H}})^{-1}\right\| < 2\left\|\mathbf{H}^{*-1}\right\|.$$

So then

$$\begin{aligned}\left\|\mathbf{p}'-\mathbf{p}^*\right\| &\leq \left\|\left(\mathbf{H}^*+\tilde{\mathbf{H}}\right)^{-1}\right\|\left\|\tilde{\mathbf{h}}-\tilde{\mathbf{H}}\mathbf{p}^*\right\| \\ &\leq 2\left\|\mathbf{H}^{*-1}\right\|\left(\left\|\tilde{\mathbf{h}}\right\|+\left\|\tilde{\mathbf{H}}\right\|\left\|\mathbf{p}^*\right\|\right) \\ &< c_2\Delta\end{aligned}$$

$$\text{where} \qquad c_2 := \frac{4\left\|\mathbf{H}^{*-1}\right\|}{N}\sum_{j=1}^{N}\left(6\left\|\mathbf{A}_j^*\right\|\left\|\mathbf{p}^*\right\|+2\left\|\mathbf{A}_j^*\right\|+\left\|\mathbf{q}^*\right\|\right), \tag{5.20}$$

which is the first part of the desired result. Applying this bound and the definition of $\mathbf{q}'$, we find after some computation that

$$\left\|\mathbf{q}'-\mathbf{q}^*\right\| < c_3\Delta$$

$$\text{where} \quad c_3 := 1+\frac{2c_2}{N}\sum_{j=1}^{N}\left\|\mathbf{A}_j^*\right\| \tag{5.21}$$

which completes the desired result. □

Proof of Proposition 5.1.
Let

$$\Delta_0^m = \min_{\substack{i \text{ s.t.} \\ \mathbf{p}_i^*\in P^*}}\left\{\frac{1}{N_i}\sum_{j=1}^{N_i}\left\|\mathbf{A}_{i_j}^*\right\| \quad , \quad \left(\frac{12}{N_i}\left\|\mathbf{H}_i^{*-1}\right\|\sum_{j=1}^{N_i}\left\|\mathbf{A}_{i_j}^*\right\|\right)^{-1}\right\}, \tag{5.22}$$

where $\bar{s}_{i_1}^*,\ldots,\bar{s}_{i_{N_i}}^*$ are the N_i d-simplices in $\mathrm{St}\{\mathbf{p}_i^*\}$, and

$$\mathbf{H}_i^* = \left(\frac{1}{N_i}\sum_{j=1}^{N_i}\mathbf{A}_{i_j}^{*\,\mathrm{T}}\mathbf{A}_{i_j}^*\right)-\left(\frac{1}{N_i}\sum_{j=1}^{N_i}\mathbf{A}_{i_j}^*\right)^{\mathrm{T}}\left(\frac{1}{N_i}\sum_{j=1}^{N_i}\mathbf{A}_{i_j}^*\right).$$

We will examine the effect of a single iteration of the MINVAR algorithm on $\mathbf{p}_i \in P$, $\mathbf{q}_i \in Q$. The results will hold independently of i, giving the desired result.

The first stage of the MINVAR algorithm calculates the least squares projection $\Pi(f^*_{\mathcal{P}}|_{\bar{s}_i})$ in each d-simplex $\bar{s}_i$ of the approximation. (Step 2 of the algorithm. Since this proposition addresses the approximation version of the problem, there is no partitioning of data to be performed in Step 1.) Let $\bar{s}_{i_1}, \ldots, \bar{s}_{i_{N_i}}$ be the d-simplices in $\mathrm{St}\{\mathbf{p}_i\}$. Since $f^*_{\mathcal{P}}$ and $f_{\mathcal{P}}$ are parameterized with the same abstract simplicial complex $\mathcal{S}^*$, $\bar{s}^*_{i_1}, \ldots, \bar{s}^*_{i_{N_i}}$ are the corresponding d-simplices in $\mathrm{St}\{\mathbf{p}^*_i\}$. Let $f^*_{\mathcal{P}}|_{\bar{s}^*_{i_j}}(\mathbf{x}) = \mathbf{A}^*_{i_j}(\mathbf{x} - \mathbf{p}^*_i) + \mathbf{q}^*_i$ and $\Pi(f^*_{\mathcal{P}}|_{\bar{s}_i}) = \hat{\mathbf{A}}_{i_j}(\mathbf{x} - \mathbf{p}^*_i) + \hat{\mathbf{q}}_{i_j}$. Since $\epsilon < \epsilon_d \leq \epsilon_c$, it follows from Lemma 5.1 that for each $j = 1, \ldots, N_i$,

$$\left\| \begin{bmatrix} \hat{\mathbf{A}}_{i_j}{}^{\mathrm{T}} - \mathbf{A}^{*\,\mathrm{T}}_{i_j} \\ \hat{\mathbf{q}}_{i_j}{}^{\mathrm{T}} - \mathbf{q}^{*\mathrm{T}}_i \end{bmatrix} \right\| < c_{1,i_j}\epsilon^2, \tag{5.23}$$

where c_{1,i_j} is given by Equation 5.15. Let

$$c_1 = \max_{i=1,..,N} c_{1,i}, \tag{5.24}$$

where N is the total number of d-simplices in $\mathcal{T}(P, \mathcal{S}^*)$. Then for $j = 1 \ldots, N_i$,

$$\left\| \begin{bmatrix} \hat{\mathbf{A}}_{i_j}{}^{\mathrm{T}} - \mathbf{A}^{*\,\mathrm{T}}_{i_j} \\ \hat{\mathbf{b}}_{i_j}{}^{\mathrm{T}} - \mathbf{b}^{*\,\mathrm{T}}_{i_j} \end{bmatrix} \right\| < c_1\epsilon^2. \tag{5.25}$$

Step 3 of MINVAR moves the vertices, taking $(\mathbf{p}_i, \mathbf{q}_i) \to (\mathbf{p}'_i, \mathbf{q}'_i)$. Since $\epsilon < \epsilon_d \leq \sqrt{\Delta_0^m / c_1}$ by hypothesis, it follows that $c_1\epsilon^2 < \Delta_0^m$. Thus, Equation 5.25 implies that the bound in Equation 5.4 is satisfied, permitting application of Lemma 5.2, which gives

$$\begin{aligned} \left\|\mathbf{p}'_i - \mathbf{p}^*_i\right\| &< c_{2,i}c_1\epsilon^2, \\ \left\|\mathbf{q}'_i - \mathbf{q}^*_i\right\| &< c_{3,i}c_1\epsilon^2, \end{aligned}$$

where

$$c_{2,i} := \frac{4\left\|\mathbf{H}^{*-1}_i\right\|}{N_i} \sum_{j=1}^{N_i} \left(6\left\|\mathbf{A}^*_{i_j}\right\| \|\mathbf{p}^*_i\| + 2\left\|\mathbf{A}^*_{i_j}\right\| + \|\mathbf{q}^*_i\|\right), \tag{5.26}$$

$$c_{3,i} := 1 + \frac{2c_{2,i}}{N_i} \sum_{j=1}^{N_i} \left\|\mathbf{A}^*_{i_j}\right\|. \tag{5.27}$$

For each $\mathbf{p}_i$, $\mathbf{q}_i$ we can compute such bounds. Let

$$c_4 := c_1 \max_i c_{2,i} \tag{5.28}$$

$$c_5 := c_1 \max_i c_{3,i} \tag{5.29}$$

Then, for all i, $\mathbf{p}'_i$ and $\mathbf{q}'_i$ satisfy

$$\left\|\mathbf{p}'_i - \mathbf{p}^*_i\right\| < c_4\epsilon^2 \tag{5.30}$$

$$\left\|\mathbf{q}'_i - \mathbf{q}^*_i\right\| < c_5\epsilon^2, \tag{5.31}$$

which is the desired result. □

Proof of Theorem 5.1.
First we establish by induction that for $k \geq 1$,

$$\frac{1}{c_4}\left(c_4\epsilon\right)^{2^k} < \epsilon_d, \tag{5.32}$$

$$\left\|\mathbf{p}_i^{(k)} - \mathbf{p}_i^*\right\| < \frac{1}{c_4}\left(c_4\epsilon\right)^{2^k}, \tag{5.33}$$

$$\left\|\mathbf{q}_i^{(k)} - \mathbf{q}_i^*\right\| < \frac{c_5}{c_4}\left(c_4\epsilon\right)^{2^k}. \tag{5.34}$$

For $k = 1$, $c_4\epsilon^2 < \epsilon_d$ since $\epsilon < \epsilon_0 \leq \sqrt{\epsilon_d/c_4}$. For $k > 1$, $\frac{1}{c_4}\left(c_4\epsilon\right)^{2^k} < \epsilon_d$ by the induction hypothesis. Moreover, $c_4\epsilon < 1$ since $\epsilon < \epsilon_0 \leq \frac{1}{c_4}$. Then $\frac{1}{c_4}\left(c_4\epsilon\right)^{2^{(k+1)}} = \left(\frac{1}{c_4}\left(c_4\epsilon\right)^{2^k}\right)\left(c_4\epsilon\right)^{2^k} < \epsilon_d$. This establishes Equation 5.32. Since $\epsilon < \epsilon_0 \leq \epsilon_d$, it follows that for $k = 1$, after a single iteration of the MINVAR algorithm, Equation 5.33 and Equation 5.34 will hold by Proposition 5.1. For $k > 1$, for all i, $\left\|\mathbf{p}_i^{(k)} - \mathbf{p}_i^*\right\| < \frac{1}{c_4}\left(c_4\epsilon\right)^{2^k}$ by the induction hypothesis. From Equation 5.32, proven above, $\frac{1}{c_4}\left(c_4\epsilon\right)^{2^k} < \epsilon_d$. Since for all i, $\left\|\mathbf{p}_i^{(k)} - \mathbf{p}_i^*\right\| < \epsilon_d$, it follows from Proposition 5.1 that

$$\begin{aligned}
\left\|\mathbf{p}_i^{(k+1)} - \mathbf{p}_i^*\right\| &< c_4\left(\tfrac{1}{c_4}\left(c_4\epsilon\right)^{2^k}\right)^2 = \frac{1}{c_4}\left(c_4\epsilon\right)^{2^{(k+1)}} \\
\left\|\mathbf{q}_i^{(k+1)} - \mathbf{q}_i^*\right\| &< c_5\left(\tfrac{1}{c_4}\left(c_4\epsilon\right)^{2^k}\right)^2 = \frac{c_5}{c_4}\left(c_4\epsilon\right)^{2^{(k+1)}},
\end{aligned}$$

which establishes Equation 5.33 and Equation 5.34. Since $c_4\epsilon < 1$, as argued above, it follows that Equation 5.33 and Equation 5.34 go to 0 as k goes to infinity. □

CHAPTER 6

The Graph Intersection Algorithm: a Special Case of MINVAR

Exploration of piecewise linear homeomorphisms for approximation began with an investigation of the scalar case, i.e. where the domain and codomain are scalar. The Graph Intersection (GI) algorithm, predecessor of the MINVAR algorithm, computes scalar piecewise linear approximations to data. This chapter presents the GI algorithm, which differs slightly from the scalar version of MINVAR. A numerical study is presented, comparing the GI algorithm with a gradient descent technique for computing PL approximations as well as two other approximation techniques, neural networks and Taylor polynomials. The previously presented local convergence result for GI [GKK00] has been subsumed by the local convergence result for the MINVAR algorithm, presented in Chapter 5.

6.1 The Graph Intersection Algorithm

The Graph Intersection (GI) algorithm is the scalar version of the MINVAR algorithm. In the scalar case, a number of things are simplified in comparison to general dimension. For example, the triangulation of a set of points in $\mathbb{R}$ is unique. The success of the GI algorithm motivated the generalization to higher dimensions.

The parameterization of a scalar PL function is $\mathcal{P} = (P, Q)$, where $P = \{p_1, p_2, \ldots, p_n\}$, $p_i \in \mathbb{R}$ with $p_1 < p_2 < \cdots < p_n$, and $Q = \{q_1, q_2, \ldots, q_n\}$ with $q_i \in \mathbb{R}$. An abstract simplicial complex does not need to be specified for a scalar PL function since a set of points in $\mathbb{R}$ has a unique triangulation, specifically $\mathcal{S} = \{\{1,2\},\{2,3\},\ldots,\{n-1,n\}\}$. The triplet $(P, Q, \mathcal{S})$ is then a PL parameterization as defined in Chapter 3, and the scalar PL function could be evaluated using Equation 3.1. In the scalar case, Equation 3.1 reduces to

$$f_{\mathcal{P}}(x) = \left(\frac{q_{i+1} - q_i}{p_{i+1} - p_i}\right)(x - p_i) + q_i \quad \text{for } p_i \le x \le p_{i+1} \tag{6.1}$$

Clearly $f_{\mathcal{P}}$ is invertible, equivalent to strict monotonicity in the scalar case, if the codomain vertices are ordered, i.e. either $q_1 < q_2 < \cdots < q_n$ or $q_n < q_{n-1} < \cdots < q_1$. This condition corresponds to the range forming a valid triangulation, corresponding to Claim 3.5.

Let a set of input-output data be given, $\mathcal{Z} = \{(x_i, y_i)\}_{i=1}^{N_d}$. Similar to the MINVAR algo-

rithm, the GI algorithm takes an initial parameterization $\mathcal{P}^{(0)} = (P^{(0)}, Q^{(0)})$ and generates a sequence of parameterizations, $\mathcal{P}^{(k)} = (P^{(k)}, Q^{(k)})$, as follows:

Graph Intersection Algorithm

1. Partition data set $\mathcal{Z}$ into subsets $\mathcal{Z}_i^{(k)}$ according to the intervals of $P^{(k)}$,
   $\mathcal{Z}_i^{(k)} = \{(x_i, y_i) \in \mathcal{Z} \mid \}$
2. For $i = 1, ..., n-1$,
   Compute the least squares affine approximation $L_i(x) = m_i x + b_i$
   to the data subset $\mathcal{Z}_i$.
3. For $i = 1, ..., n-2$,
   if L_i, L_{i+1} are not parallel,
   let (p'_{i+1}, q'_{i+1}) be the graph intersection point of L_i, L_{i+1}
   $p'_{i+1} = \frac{b_{i+1} - b_i}{m_i - m_{i+1}}, \quad q'_{i+1} = m_i p'_{i+1} + b_i.$
   if (p'_{i+1}, q'_{i+1}) meets the Acceptance Condition (see page 83)
   $(p_{i+1}^{(k+1)}, q_{i+1}^{(k+1)}) = (p'_{i+1}, q'_{i+1})$.
   else
   choose $(p_{i+1}^{(k+1)}, q_{i+1}^{(k+1)})$ using "Special Rule" (see page 84)
   else
   let $(p_{i+1}^{(k+1)}, q_{i+1}^{(k+1)}) = (p_{i+1}^{(k)}, q_{i+1}^{(k)})$
4. $(p_{i+1}^{(k+1)}, q_{i+1}^{(k+1)}) = (p_{i+1}^{(k)}, L_1(p_{i+1}^{(k)}))$.
   $(p_n^{(k+1)}, q_n^{(k+1)}) = (p_n^{(k)}, L_{n-1}(p_n^{(k)}))$.
5. If $\mathcal{P}^{(k)}$ has converged
   Stop.
   Else $k = k + 1$. Goto 1.

The first and second steps of the algorithm correspond directly to the MINVAR algorithm. The differences appear in step 3. In the MINVAR algorithm, the new location of the domain vertex is given as the solution of a regularized quadratic minimization. In the scalar case, the unregularized minimization reduces to finding the intersection point of the least squares affine approximations on either side of the previous vertex, the "graph intersection point." Step 3 of the GI algorithm directly computes the GI point, using no regularization. Instead, the domain vertex is checked to see if it meets the "acceptance condition," which guarantees that a tangle of the domain vertices will not occur.

In some circumstances, the best fit lines of neighboring cells have the same slope, precluding any intersection point (in the case of parallel lines) or yielding a continuum of intersection points (in the case of coincident lines). An equally important, but less obvious problem occurs when neighboring cells have best fit lines which are nearly parallel. This may cause the resulting intersection point to be far away from the previous vertex, potentially even outside of the domain of approximation, destroying the domain triangulation. The Acceptance Condition and "Special Rule" ensure that the proper domain vertex ordering, and hence a well-defined approximation, is maintained. The Acceptance Condition

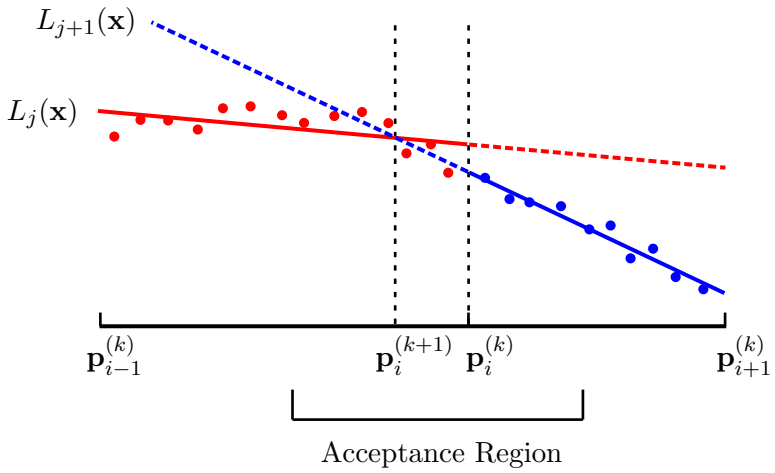

Figure 6.1: Visualization of the Graph Intersection algorithm. Since the intersection point of L_j and L_{j+1} lie within the acceptance region, $\mathbf{p}_i^{(k+1)}$ and $\mathbf{q}_i^{(k+1)}$ are placed at the intersection point. If the intersection point did not lie within the acceptance region, then a special rule would be used.

checks if the intersection point is between the "midpoints" of the vertex's two cells. If not, the new vertex is selected using a "Special Rule" rather than the intersection point. The "Special Rules" are also discussed below.

The least squares affine fits in step 2 of the algorithm can be computed using the standard least squares methodology. Explicitly, for the subset of the data $\mathcal{Z}_i$, let

$$N_i = \sum_{(x_j,y_j)\in\mathcal{Z}_i} 1.$$

$$s_{x,i} = \frac{1}{N_i}\sum_{(x_j,y_j)\in\mathcal{Z}_i} x_j, \qquad s_{xx,i} = \frac{1}{N_i}\sum_{(x_j,y_j)\in\mathcal{Z}_i} {x_j}^2,$$

$$s_{y,i} = \frac{1}{N_i}\sum_{(x_j,y_j)\in\mathcal{Z}_i} y_j, \qquad s_{xy,i} = \frac{1}{N_i}\sum_{(x_j,y_j)\in\mathcal{Z}_i} x_j y_j,$$

Then the parameters of the least squares affine fit $L_i(x) = m_i x + b_i$ to $\mathcal{Z}_i$ are given by

$$\begin{aligned} m_i &= \frac{s_{xy,i} - s_{x,i}\, s_{y,i}}{s_{xx,i} - s_{x,i}\, s_{x,i}}, \\ b_i &= \frac{s_{xx,i}\, s_{y,i} - s_{x,i}\, s_{xy,i}}{s_{xx,i} - s_{x,i}\, s_{x,i}}. \end{aligned}$$

In Step 3 of the algorithm, the graph intersection point is checked to see if it is "acceptable." The Acceptance Condition is,

Acceptance Condition: A graph intersection point (p_i', q_i') is considered acceptable if $s_{x,i-1} < p_i' < s_{x,i}$

In other words, the intersection point is acceptable if it lies between the means of the data in the left- and right-hand neighboring cells. If all intersection points are acceptable, then clearly the order of the domain vertices is preserved and the triangulation does not tangle. If an intersection point is not acceptable, then a new acceptable domain vertex is chosen via a "special rule" described below. Forcing all domain vertices to be acceptable is sufficient but not necessary for preventing tangles of the domain triangulation. Though conservative, it provides a convenient method for diagnosing and fixing potential problems.

Several different "special rules" have been tested for selecting domain and codomain vertices when the graph intersection point fails to be acceptable. Surprisingly the numerical performance of the algorithm appears almost entirely independent of the rule used. Observing the algorithm's convergence on a wide variety of datasets suggests a qualitative reason. In general the special rules are needed only during the first few iterations of the algorithm, after which time all intersection points are all acceptable as the algorithm converges. The numerical results presented in the next section use the pointwise decision rule.

The Pointwise Decision rule, inspired by the Kioustelidis algorithm [Kio81, Kio80], directly examines the error modulus. The rule starts at the location of the previous domain vertex and looks at the nearest data point. It determines which least squares affine map, from the left or right interval, has smaller error at that data point. The domain vertex is moved in the direction that reduces the error (e.g. if the least squares fit to the left yields a smaller error on the data point, then move the domain vertex to the right). Then pointwise-decision continues to step through data in this direction until the least squares fit that was better begins to yield worse errors compared to the other least squares fit. For the following discussion, assume the data set $\mathcal{Z}$ is ordered according to the domain value of the data points. Let the error modulus be $e_i(x_j, y_j) = |y_j - L_{i-1}(x_j)| - |y_j - L_i(x_j)|$. The Pointwise Decision rule is given by:

1. Find the data point x_j nearest to $p_i^{(k)}$
2. If $e_i(x_j, y_j) > 0$
   $d = 1$
   Else
   $d = -1$
3. While $d \cdot e_i(x_j, y_j) > 0$
   $j = j + d$
4. $p_i^{(k)} = x_j, \quad q_i^{(k)} = \frac{1}{2}(L_{i-1}(x_j) + L_i(x_j))$

This rule extends a domain interval data point by data point at the cost of its neighbor, explicitly looking for a lower error modulus. In general, the Pointwise Decision rule will stop where the two neighboring maps are discontinuous. Since we want a continuous approximation, we pick the knot's range value as the average of the two neighboring maps at that point. The maps in the two neighboring cells are no longer the least squares maps

which were used while determining the new knot, thus the error modulus reduction is not guaranteed. Nevertheless, this rule works quite well in practice.

The second "special" rule is cross-validation. The underlying principle is to expand the interval whose least squares fit is performing better in terms of mean square error over data points. The mean square errors are computed and are used to form a convex combination of the "midpoints" of the two intervals. Let MSE_1 and MSE_2 be the mean squared error of $L_{i-1}(x)$ and $L_i(x)$ respectively on the data set $\{(x_j, y_j) \in \mathcal{Z} | \, s_{x,i-1} < x_j < s_{x,i}\}$. Let $\alpha = MSE_1/(MSE_1 + MSE_2)$. Then the cross-validation rule gives

$$p_i^{(k)} = \alpha s_{x,i-1} + (1-\alpha) s_{x,i} \tag{6.2}$$

$$q_i^{(k)} = \alpha s_{y,i-1} + (1-\alpha) s_{y,i} \tag{6.3}$$

The new vertex locations are a convex combination of the mean of the data on the left and right side of the knot, with the coefficients of the combination based on the mean square error of the least squares affine approximations on the data lying in the acceptable region. The side which has the larger mean square error will shrink, while the side with the lower mean square error will expand.

A third special rule, "Error Balance," was also inspired by Kioustelidas's algorithm, and was a precursor to the Pointwise Decision rule. The error balance rule is no longer used.

6.1.1 Relationship of Graph Intersection and MINVAR

An iteration of the GI algorithm in which all the intersection points are acceptable is exactly the same as an iteration of the MINVAR algorithm with $\lambda = 0$. The local convergence result presented in Chapter 5 considers the MINVAR with $\lambda = 0$, and shows that the vertices of the approximation get closer and closer to the vertices of the data generating function, if the initial approximation starts close enough. If the GI algorithm, rather than MINVAR, is started close enough to the vertices of the data generating function, then the acceptance condition is never violated, and the GI algorithm is identical to the MINVAR algorithm with $\lambda = 0$. Thus the result from Chapter 5 applies to the GI algorithm as well, superseding a specialized scalar result that appeared in [GKK00].

6.2 Implementation of Graph Intersection Algorithm

The GI algorithm was implemented as a set of Matlab scripts, which are available at `http://www.eecs.umich.edu/~regroff/research/software.html`. The benefit of Matlab is the speed with which code can be prototyped. Evaluating a scalar piecewise linear function requires searching for the domain interval in which the point to be evaluated lies. This is a particularly slow task in Matlab, requiring either `for` loops or a matrix-ized calculation that computes much more than necessary, but is nonetheless faster due to the speed of Matlab's matrix routines. In the scalar case, the overhead of Matlab is tolerable.

For higher dimensions, the overhead becomes too weighty, and it is best to write code for evaluating and approximating PL functions in a compiled language, as was done for the MINVAR implementation.

The GI Matlab scripts can produce PL approximations using both the GI algorithm as well as the scalar MINVAR algorithm. (Recall that these algorithms differ in that MINVAR uses regularization to prevent domain triangulation tangles while GI uses the special rules to avoid tangles.) The following section presents results from the GI algorithm exclusively, while the application section on camera calibration at the end of this chapter uses both MINVAR and GI.

6.3 Numerical Results for the Graph Intersection Algorithm

We studied the approximation of homeomorphisms of the interval $[0, 1]$ into itself from noiseless data using piecewise linear approximants (PL) as well as neural networks (NN) and Taylor polynomials (TP) representing truncated Taylor series. Each approximation was given the same number of parameters. For example, the data shown here is for 16 parameters, corresponding to a NN with 5 neurons, a degree 15 TP and a PL approximant with 9 segments (the endpoints are fixed). All code was implemented in MATLAB. TP solves the standard linear least squares problem in closed form to find the coefficients. The NN has a single hidden layer with hyperbolic tangent neurons and trains using Levenberg-Marquardt descent on the squared error. Results for two different forms of PL training are shown. The first (constr PL) uses MATLAB's constrained optimization algorithm directly on the squared error. The Graph Intersection (GI PL) algorithm is the second method for computing PL approximations.

The choice of homeomorphisms, functions which are invertible, continuous, and have a continuous inverse, is motivated by the proposed applications, where a "change of coordinates" is learned between an *a priori* topological model with invariant properties and the world as represented by data. Six families of homeomorphisms were chosen for study. Families `tansig1` and `tansig2` are invertible superpositions of hyperbolic tangents, and families `plfun1` and `plfun2` are invertible piecewise linear. Families `tansig1` and `plfun1` lie within the representational power of the neural network and the piecewise linear approximant, respectively, whereas the approximations are "underparameterized" on `tansig2` and `plfun2`. The fifth family is high degree polynomials (`rndply`). These functions are generated by taking the composition of seven invertible quadratic functions. The last family is time one maps from differential equations (`diffeq`). Recall that the time one map of a differential equation with unique solutions is not only a homeomorphism but also a diffeomorphism. One hundred functions were randomly sampled from each of the six families.

Figure 6.2 summarizes the approximation error for the study. As expected, NN outperforms the other three techniques on the class `tansig1`, where the functions lie with

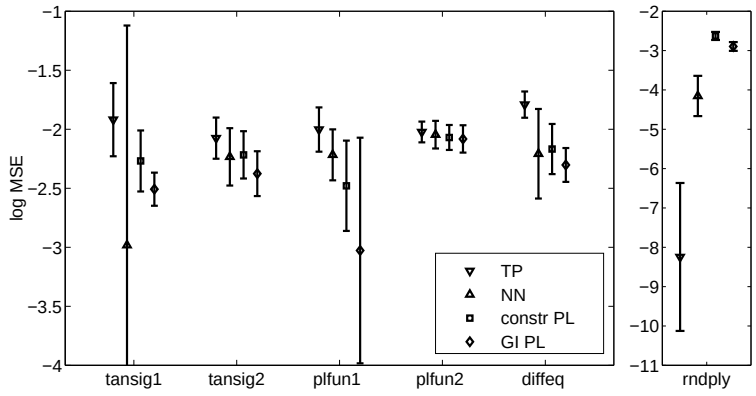

Figure 6.2: Comparison of approximation techniques on six function families. The approximation techniques are Taylor Polynomial(TP), Neural Network(NN), PL using constrained optimization (constr PL) and PL using GI (GI PL). For each function family, 100 functions were randomly sampled. Mean and standard deviation bars of the Log Mean Squared Error are shown for each approximation technique on each function family.

Table 6.1: Mean computational cost (in megaflops) of the approximation techniques on the six function families.

	TP	NN	PL con	PL GI
`tansig1`	0.187	10.2	1.89	0.168
`tansig2`	0.187	9.38	1.68	0.240
`plfun1`	0.187	9.71	1.93	0.185
`plfun2`	0.187	9.74	2.12	0.538
`rndply`	0.187	10.7	1.02	0.157
`diffeq`	0.187	9.71	2.06	0.195

the parameter space of the NN. Similarly both PL techniques outperform the other two techniques on `plfun1`, where the functions lie within the parameter space of PL. GI PL consistently produces better results than constr PL. On most families the PL techniques perform better than the other two techniques. An exception is the `rndply` family, on which NN did notably better than PL, but otherwise PL is competitive in terms of error, and also has the benefit of being closed form invertible.

Table 6.1 shows training cost in terms of floating point operations (flops) for the algorithms on the various function families. By far the highest computational cost belongs to NN, which is almost two orders of magnitude more intensive than TP. Note that constr PL is also computationally expensive. GI PL is close to TP in computational cost and on some families actually uses less flops than TP, which is a linear-in-parameters approximation.

Overall, on the classes of homeomorphisms studied, GI PL is competitive in terms of squared error with both TP and NN, and it is surprisingly computationally efficient. Moreover it is possible to easily check the invertibility of the resulting function, as well as invert it in closed form.

CHAPTER 7

Color Systems Management

Interest in invertible approximations was initiated through an NSF GOALI[1] sponsored collaboration between Xerox Corp. and the University of Michigan on control problems in printing. Through this collaboration, color systems management emerged as a compelling problem for which which progress could be made. The color systems management problem requires a computationally effective representation for changes of coordinates.

If one repeatedly prints a color image on a laser printer over a period of weeks and examines the resulting prints, one will notice variation in color tone from print to print, though each may look fine individually. These variations are undesirable, and in some situations unacceptable, for instance when reproducing trademarked colors or photographic quality images. Offset lithography, the technology currently used for the majority of color printing, reproduces color with very high consistency, but has higher fixed costs per document run[2] [Bla83]. In comparison, electrophotography, the technology in laser printers and photocopiers, has very low fixed costs per document run, making personally customized documents feasible, but produces colors less consistently. With improved color consistency, electrophotography would emerge as the preferred option in terms of overall quality and economy for small color printing jobs, i.e. less than 2000 copies. Toward this goal, Xerox has implemented low level signal controls inside laser printers and copiers for twenty years. The next step is to wrap a higher level feedback around the color commands to the print engine.

This chapter provides background on the electrophotographic process[3] and an overview of control of electrophotography. The color systems management problem is introduced, and it is shown that this problem can be reduced to the approximation of a change of

[1]This work is a collaboration between the University of Michigan and Xerox Corp., funded in part by NSF Award ECS-96322801 under the Grant Opportunities for Academic Liaison with Industry program.

[2]Offset lithography requires a set of etched metal plates, one plate for each color ink used, typically cyan, magenta, yellow, and black. When printing a new document, the fixed costs associated with the initial production of the plates is relatively high, but the marginal cost per printed page is low. In comparison, electrophotography (used in laser printers and photocopiers) has essentially no fixed cost for printing a new document, but the marginal costs are higher.

[3]A version of this background material appears in [GT03], cowritten with Tracy Thieret of Xerox Corp.

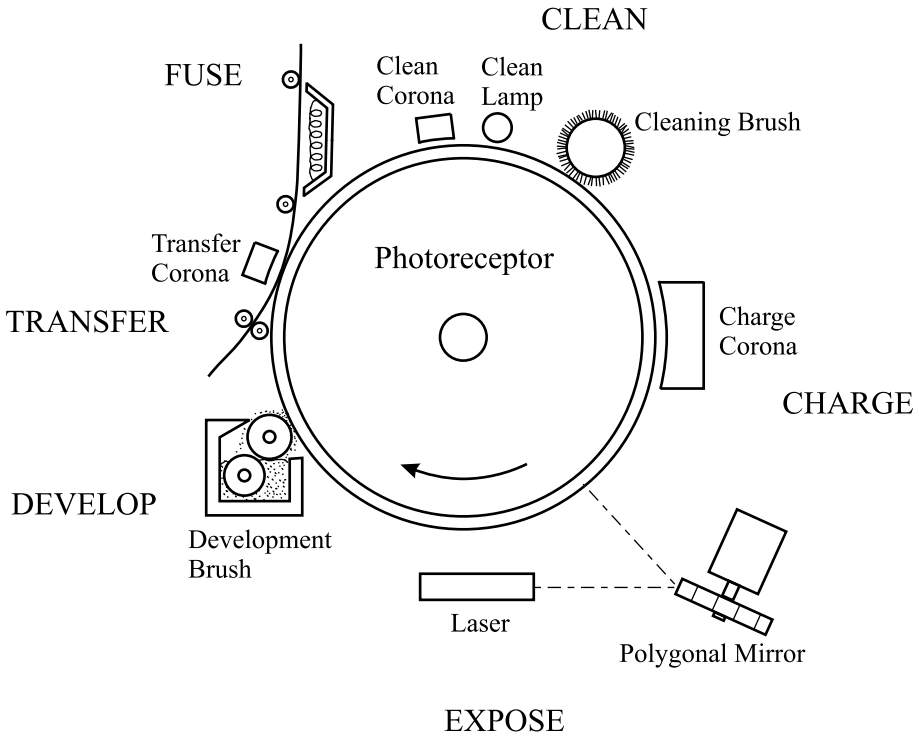

Figure 7.1: The print engine of a laser printer with the steps of the electrophotographic process labeled.

coordinates. A numerical study is presented, comparing MINVAR PLH approximations and lookup tables, the industry standard, for a set of color data supplied by Xerox Corp.[4]

7.1 Electrophotography

Laser printers are based on electrophotography, the same marking technology used in a photocopier. Electrophotographic reproduction centers around the photoreceptor, a belt or drum consisting of at least two layers, a photoconductive layer and a conductive substrate. In darkness, the photoreceptor can hold a static charge, but when exposed to light it discharges. The desired image is "painted" in static electricity and then developed with toner, small charged plastic particles. The toner is transferred to the print medium and then fused. The electrophotographic process consists of six steps: Charge, Expose, Develop, Transfer, Fuse, and Clean. The photoreceptor transports the image, in its various forms, between the subsystems. This section will discuss the steps of the electrophotographic process for monochrome printing, illustrated in Figure 7.1. Further exposition is provided in [DNT02, PS93, Sch88].

The Charge step deposits a uniform static charge on the photoreceptor. Typically, this

[4]Some of these results have been published in [GKK+99, GKKT00].

is performed by a corona discharge, produced by a corotron or scorotron. A corotron is a thin wire to which a high ac and dc voltage is applied. The voltage creates a corona, the breakdown of the surrounding air, which transfers charge to the photoreceptor. A scorotron is a corotron with the addition of a control grid between the wire and the photoreceptor. Voltage is applied to the grid to limit and to improve the uniformity of the charge on the photoreceptor. Consistent, uniform charging of the photoreceptor is necessary for accurate image reproduction.

The Expose step produces a latent image, a pattern of charged and discharged areas, of the desired output on the photoreceptor. In a traditional light lens photocopier, the photoreceptor is discharged in the areas that are not to receive toner by bright light reflected off the original document. In this case a process called Charged Area Development (CAD) is used to develop the latent image, covering the remaining charged areas with toner. In a printer or digital photocopier, the latent image is produced by an addressable light source, a laser or light emitting diode (LED) array. For most text images, the total toner area coverage is between 5 and 10%. For this reason, printers and digital copiers use the addressable light source to discharge areas of the image that are to receive toner, reducing the duty factor of the light source, the percentage of time the light source is on. In this case, a process called Discharged Area Development (DAD) is used to develop the latent image, covering the discharged areas with toner.

When the light source is a laser, the output image is rasterized, broken up into lines from top to bottom, in a similar way as a video raster is painted on the screen of a monitor by the electron beam. The light source, typically a diode laser, remains fixed in place, while the laser beam, reflected off a rotating polygonal mirror with constant angular velocity, sweeps across the photoreceptor. Each face of the mirror causes the laser to sweep out one line across the photoreceptor. The laser is modulated on and off by a bit stream, producing regions on the photoreceptor which are uncharged or charged, respectively. The combination of the laser and the rasterizing optics is collectively referred to as a Raster Output Scanner or ROS. The resulting pattern of charges on the photoreceptor is called the latent image.

Another popular addressable light source is the LED bar. Light Emitting Diodes may be constructed into silicon chip arrays and then assembled to produce an exposure system covering the full width of the print medium called an image bar. Each of the individual LEDs may be modulated directly by addressing logic contained in the carrier for the image bar. The drive electronics may also contain compensating resistors that trim the intensities of the individual LEDs so that the illumination from each is uniform across the bar. The bar is placed in the appropriate exposure location and the LEDs are turned on and off by a bit stream, similar to the laser imaging case. LED bars avoid the architectural (they are smaller than the laser and the optical system) and control (no rapidly moving parts) constraints that govern the use of laser diodes. However, the loss of a single LED shows up

readily as an image quality defect that requires the purchase and installation of an expensive new image bar.

In both cases the imaging system imposes a two dimensional grid of dots on the photoreceptor. Each of these dots is called a pixel (from PICture ELement), analogous to the well-known pixel of video display technology with the exception that most electrophotographic imaging technologies are capable of producing only binary (two level - on/off) pixels. One dimension of the two-dimensional grid is achieved by moving the photoreceptor. This dimension is called the "process direction," because the media moves through the system in this direction, or the "slow scan direction," and corresponds to the vertical dimension in video rasters. The spatial frequency of the lines taken in the process direction is a function of the photoreceptor speed and the scan speed of the laser or the strobing frequency of the LED bar. The direction perpendicular to the slow scan direction is called the "fast scan direction" and corresponds to the horizontal sweep in the video raster. The spatial frequency of the pixels in this direction is governed by the frequency of modulation provided to the laser for ROS systems or by the LED spacing in LED bars.

When the two dimensional grid of pixels is designed, the designer specifies a certain addressability. This quantity indicates how many dots/inch (dpi) may be written to the photoreceptor and is, for historical reasons, often specified in multiples of 300. Thus, when a printing system is advertised as being 600x1200, the raster lines are placed 1/600in. (42.3μm) apart and the modulation of the imaging system is 1200 dpi in the fast scan direction. "Addressability" is often confused with "resolution." Addressability is associated with the imaging system's ability to space dots closer or farther from one another. Resolution is the ability of an optical system to discriminate fine detail, referring in this case to the imaging system's ability to reproduce fine structure in an image. The difference between these two terms arises as a function of the size of the dot produced by the imaging system. Smaller dots will preserve image detail better than larger dots at the same addressability. The imaging system does not use exactly rectangular dots, but usually elliptical, and thus it is impossible to fill a pixel exactly. The dot size is often made larger than a pixel in order to avoid holes at the corners of the pixels that would receive no exposure. Overfilled dots reduce the resolution of the printer at constant addressability. Marketing statements tend to focus on the easier to quantify number of addressability. The issues of addressability versus resolution arise in the other printing technologies as well.

The Development step uses toner to develop the latent image. Toner consists of pigmented, electrostatically charged plastic particles, 5-25μm in diameter. In the developer housing, the toner is mixed with larger carrier particles or beads, 80-700μm in diameter, which serve two purposes. First, extremely fine powders such as toner are difficult to transport, and can produce dirt inside the machine when it escapes the housing or spots on portions of the document that were supposed to be white. The carrier beads may carry up to 1000 toner particles, preventing powder contamination of other system components

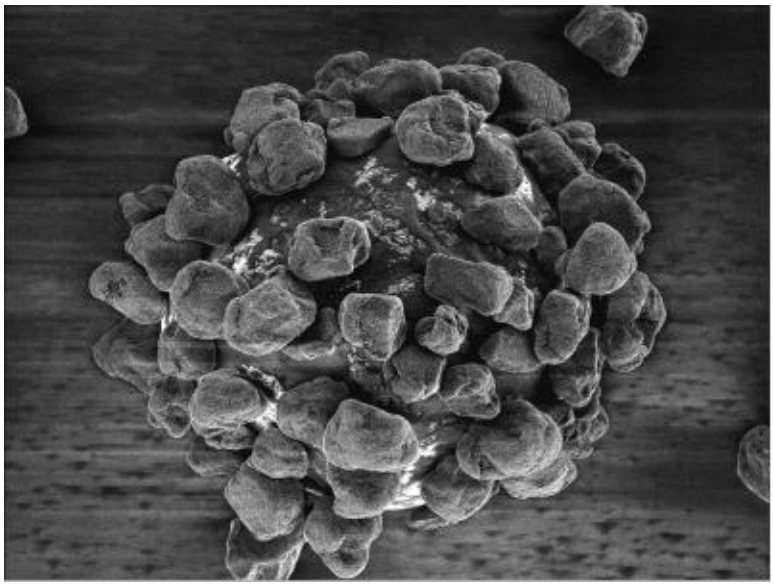

Figure 7.2: A carrier bead with attached toner. This bead is approximately 60 μm across.

or the image. Second, the carrier beads charge the toner particles triboelectrically, i.e. by friction. A photomicrograph of a carrier bead and its attached toner is shown in Figure 7.2. The magnetic brush development system is the most widespread. In this system, the carrier beads are also magnetic. The toner-covered carrier beads form brush like chains on a revolving shell, bringing the beads into contact with the photoreceptor. The resulting physical agitation in the development nip serves to break the adhesive and electrostatic forces binding the toner to the carrier and frees the toner to move under the influence of the photoreceptor latent image.

The developer housing is biased at a voltage between the photoreceptor's charge and discharge voltages. This dc bias produces two polarities of field with the photoreceptor. In Discharged Area Development (DAD), used in digital printing, the areas that were exposed by the imaging system, the "development" field, points toward the photoreceptor, attracting the tribocharged toner. Meanwhile, in the unexposed photoreceptor regions (the areas intended to be white in the final image), the electric field ("cleaning" field) points toward the development roll, causing toner to remain on the roll. Thus, the charged toner can discriminate between the image and background regions of the image.

The Transfer step moves the developed image to the print medium, generally paper. The medium is brought in contact with the photoreceptor. A transfer corona, with polarity opposite the toner, pulls the toner from the photoreceptor to the paper. Large particles tend to be transferred more efficiently than small particles, placing a limit on the size reduction of toner particles. In a typical, well-functioning system, between 90-100% of the toner is transferred from the photoreceptor to the print medium.

The Fusing step permanently fixes the toner to the print medium, typically by applying heat and pressure by passing the medium between a pair of heated rollers. The rollers heat the toner sufficiently above the plastic's glass transition temperature to allow it to melt and fuse with the print medium. The pressure forces the melted toner into intimate contact with the paper fibers. When the toner cools, it undergoes thermal contraction. For images that cover a large percentage of the paper, the thermal contraction can cause the paper to curl, necessitating a de-curling step to obtain flat sheets.

The Cleaning step prepares the photoreceptor for the next image by removing any remaining toner left from the transfer step. This is typically performed by a third corona, which discharges the toner left on the photoreceptor, coupled with a bright light that discharges the photoreceptor. A brush or elastomer blade, similar to that used in the development stage, wipes the toner from the photoreceptor. Finally, an erase lamp removes any remaining charge from the photoreceptor.

Laser printers are very quiet and fast. The printers range from desktop models that print 2-4 pages per minute at an addressability of 300x300 dpi, to commercial printers at up to 2400 dpi. The fastest of these devices can print and bind a 250-page book with covers, inserted tabs, and binding in less than two minutes.

7.2 Control of Color Electrophotography

Color printing requires a number of other technologies layered on top of the monochrome print engine described in the previous section. To print a color image, for each pixel the print engine receives a command specifying a continuous tone value for each of the toner colors, specifying how "dark" that toner color should be. Toners are typically cyan, magenta, and yellow (CMY), though black (K) is often added to extend the color gamut, the range of reproducible colors [Kan97]. Electrophotography, like many other print engine technologies including ink jets, can only produce binary colored dots; toner is either on the page or not. Continuous tones are approximated by halftoning, a technique that uses patterns of binary pixels to trick the eye into "seeing" intermediate tones [Uli87, Kan02]. The separate halftoned images corresponding to each color of toner are printed one on top of the other, producing the desired colors. Misalignment of the separated images, called misregistration, leads to poor color reproduction. Monochrome electrophotography is itself a complicated process involving electrostatics and the sources for process variation are many. Color electrophotography is still more complicated, and color consistency is difficult to achieve.

Historically, stabilization of color reproduction has focused on stabilization of the electrophotographic process itself. It is common practice to calibrate the electrophotography during setup or warm-up and run the system largely open loop during printing. High quality, high speed color printing at high throughput rates requires active feedback and feedforward

controls of the electrophotographic process in order to maintain stable, predictable color performance. Xerox has practiced closed-loop control in copiers and printers for nearly 20 years, and for monochrome and early color printing, stabilizing the process alone was sufficient. However, the number of process actuators available is very limited and the number of outputs that require stabilization are many. The need for improved consistency in color printing is driving interest in control of electrophotography [MWD+96, GKK+99, THB95].

The tone reproduction curve (TRC) plots the printed (monochrome) tone against the requested tone. A color printer has one TRC for each color of toner. Drift in the TRC indicates drift in the image reproduction. In 1995, a three level control architecture for stabilizing the TRCs of a color printer was devised and patented [THB95]. Level 1 wraps tight feedback loops around sensor-actuator pairs in various subsystems. Sensors monitor system variables such as toner concentration[5], temperature, relative humidity, and photoreceptor voltages. Actuators in the system include the grid biases, corotron voltage, and toner dispense[6]. Level 2 uses subsystem performance information to update parameters, such as set points, in Level 1 control algorithms. Level 3 uses system wide information to maintain consistent TRCs. Recent innovations have been productized first in the DocuColor 40 printer and more recently in the DocuColor 2060. These innovations involve color sensing at the output and feedback to the TRCs in the imaging system. Many short term variations in color reproduction are compensated for in this way. However, these three control processes only stabilize the single separation TRCs and do not address the disturbances that affect the color mixing recipes. Materials and process variability still require that these disturbances be addressed. The customer's frequent need for printer recalibration is testimony to this fact.

7.2.1 Control Via the Forward and Inverse Device Characterization

An additional loop at the level of the preprocessor and print engine would further improve color consistency. Figure 7.3 shows the decomposition of a printer into the preprocessor and print engine. In order to understand how color images are specified to a printer and how this outer loop would work, it is first necessary to introduce some concepts from color science: color spaces and color space transformations.

The human eye has three different types of cones, the structures responsible for perceiving color, that correspond roughly to red, green, and blue, though the response spectra overlap considerably. If two light sources have different spectra but produce the same response in the cones, then they will be perceived as the same color. A *color space* is a coordinate space used to represent colors [Kan97]. Due to the nature of the human visual sensor, a color space may have as few as three coordinates to describe all perceivable colors, even though visible light spectra are, for all practical purposes, infinite dimensional. More-

[5]Toner concentration (TC) is the ratio of toner to carrier particles in the development housing.

[6]Toner dispense is the process of moving toner from the reservoir to the development housing.

over, most perceivable colors can be reproduced by mixing just three appropriately chosen colors. In printers, cyan, magenta and yellow (CMY) pigments are typically used. Three pigments are not sufficient to produce all colors, because the spectra of the pigments do not exactly match the response spectra of the cones, so reproducing some colors would require "negative" pigment. A previously mentioned, black (K, giving CMYK) is often added in order to extend the color gamut. Printer manufacturers are interested in moving eventually to six or even more colors to further extend the gamut.

Color spaces are divided into two categories, device independent and device dependent. A device independent color space is based upon some absolute standard for color. For example, the L*a*b* coordinate space is a device independent space based on the psychophysics of the human eye [Kan97]. It is constructed so that a "just noticeable difference," the smallest perceivable difference in color, corresponds to a ball of approximately radius 1 in L*a*b*, though the true size of the just noticeable difference sphere varies with position in L*a*b*. In contrast, a device dependent coordinate system is related to how a specific device produces colors. For example, in a color printer the device dependent color space coordinates give a recipe for creating the color, specifying how much toner of each type to use. In a computer monitor, the device dependent color space is RGB (red, green, blue) and indicates the intensities of the electron guns or LEDs. Note that a color represented by the device dependent coordinates depends on the specific device and on the settings and configuration of that device.

A *color space transformation* is a transformation from one color space to another such that a point in the first color space is perceptually identical to the transformed point in the second color space [Kan97]. In other words, a color space transformation is a change of coordinates. The print engine of the printer accepts a color command in CMY coordinates and prints a color measured in L*a*b* coordinates. Thus, the print engine can be modeled as the embodiment of a color space transformation $T : \text{CMY} \to \text{L*a*b*}$ (See Figure 7.3). The transformation T, known as the *forward device characterization*, indicates how a CMY color command to the engine results in a printed color measured in L*a*b* coordinates.

The user's objective in color printing is to reproduce specific psychophysically perceived colors accurately. Ideally, the reproduced images should appear the same independent of the printer or time. Images are specified to the printer in terms of a device independent color space, say L*a*b*, so that the combinations of pigments used to reproduce the desired colors on a particular print engine are transparent to the user. In order to print an image specified in L*a*b*, the printer must perform a color space transformation to its device dependent coordinate system, CMY.[7] This transformation is in fact T^{-1}, the inverse of the forward device characterization, known appropriately as the *inverse device characterization*.

[7]Printers often use four or even more pigments, but the present discussion addresses only CMY. When the dimension of the device dependent color space is greater than three, numerous combinations of toner produce the same observed color in L*a*b*. Thus, there is no single well defined inverse of T, but rather only a pseudoinverse $T^{\#}$, such that $T \circ T^{\#} = i_d$, where i_d is the identity transform. The present discussion is restricted to the case where the device dependent coordinate system is CMY.

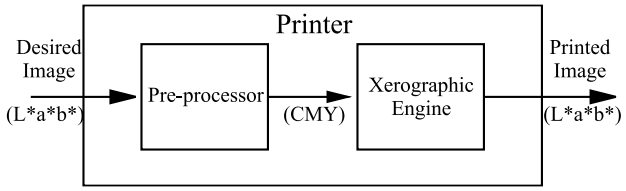

Figure 7.3: Decomposition of an electrophotographic (laser) printer into the print engine and preprocessor. The engine embodies the transformation T, while the preprocessor performs an approximation of the inverse $\tilde{T}^{-1}$, such that $T \circ \tilde{T}^{-1} \approx i_d$.

In the absence of sufficiently accurate first principles models for T or T^{-1}, an approximation to T^{-1} must be computed, written as $\tilde{T}^{-1}$. An overview of this type of inverse problem in color science is provided in [Bal03].

Industry practice has gradually settled on that least parsimonious of nonparametric representations — the look-up table (LUT) [Kan97] — for the inverse device characterization. These large LUTs, known as color mixing tables, are constructed at printer calibration and interpolated to yield the halftone densities of the individual toners layered to construct the desired color. The data used to construct these tables is typically 1000 or more color patches, each printed using a specified recipe (CMY) and then colorimetrically measured to determine the resulting L*a*b* coordinates. Color patch experiments correspond to evaluating T rather than T^{-1}. In order to generate a uniform grid in L*a*b* space for the LUT, either the results of the color patch experiments must be interpolated or the experiments must be cleverly iterated. Since color patch experiments are time consuming, interpolation is generally used [KNPH95, KK92], increasing the potential error.

Generating an approximation to the inverse device characteristic is part of the printer calibration procedure. For low end printers, a single calibration may be performed in the factory for an entire product line, while high end printers are individually calibrated in the factory and can be recalibrated in the field by a technician. The forward device characterization of the print engine drifts due to environmental factors such as temperature and humidity as well as a variety of disturbances that remain unaccounted for in the control loops of Levels 1-3. The resulting mismatch between the current forward device characterization and the formerly calibrated approximation to the inverse device characterization causes undesirable variation in the reproduced image.

Rather than periodically calling a technician to recalibrate the printer, real-time or automatic periodic updates to the inverse device characterization approximation could be used to stabilize color reproduction. For this purpose, it is useful to have access to an effectively computable approximation to both the forward device characterization and its inverse[8]. The forward device characterization is useful for performing feedback on the print

[8]It is of considerable importance in the application that the functional representation itself be invertible. The alternative of separately fitting to data an independent forward and inverse model is undesirable.

engine, while the inverse device characterization is required for generating color commands for the print engine. The term "effectively computable" means that the approximation should be both numerically simple and parsimoniously parameterized, which would ensure that the control action could be imposed in a timely manner using a small amount of data.

Two difficulties are associated with transferring this process to real time in the machine. The first is the number of patches required and the resources needed to print them. Making paper prints reduces the productivity of the device and also requires that the patches be read. Input scanners would be useful for this purpose but there are large regions of color space where their sensitivity is inadequate. The second is that if the patches are made slowly and read by machine sensors, then the number of patches required would consume a time comparable to the disturbances, making the exercise pointless. This situation further argues for a technique that will yield sufficient accuracy but require many fewer patches. Fewer patches implies a more parsimonious representation of the inverse device characteristic. If such a technique were realized, the goal of real-time control of color mixing might be accomplished.

The degree of parsimony of an approximant reduces to the question of how approximation errors decrease as the dimension of the defining parameter space is allowed to increase. In the case of a PL, the dimension of the parameter space is a linear function of the number of vertices in the domain triangulation. It is standard practice in the color systems management industry [Kan97] to use piecewise linear interpolation, often called tetrahedral interpolation, on calibrated LUTs[9]. Since the grid density of an LUT is typically quite high (e.g. entailing thousands of color patches) the resulting approximation cannot be considered parsimonious. Moreover, the resulting PL is not generally invertible, at least not in the form of a LUT. Given an LUT data set, it seems natural to inquire whether some better means of locating the vertices might yield an invertible and far more parsimonious PL representation.

This thesis proposes piecewise linear homeomorphisms (PLH) as an effectively computable approximation of the device characteristic and its inverse. An approximation of the forward device characteristic could be computed from color patch experiments. This approximation can be inverted in closed form to give an approximation to the inverse device characteristic. A PLH computed with MINVAR can be considerably more parsimonious than a LUT since the domain triangulation is adjusted to place more parameters in areas of the domain where they are needed. MINVAR, a batch algorithm, could periodically recalibrate the device characteristic approximation. A yet-to-be-developed online version of the MINVAR algorithm could eventually allow real-time control at the level of the print engine and

Calibration experiments and control measures are typically affected with respect to the forward map since that is how real user inputs are presented to a physical device. If the inverse cannot be computationally derived from the forward representation, then a new inverse must be fit independently at each re-calibration, and, worse, at each new control step.

[9]Section 3.2 discusses data interpolation. In tetrahedral interpolation, the data forms a regular grid of the space and each cube of data uses the same local triangulation.

preprocessor using the device characterizations.

7.3 Numerical Performance of the MINVAR Algorithm on Color Data

Xerox provided a set of data from CMY to L*a*b* generated from a color model of a commercial printer [Bal96]. This data set is used here to perform a comparison between MINVAR PL approximations and uniform lookup table (LUT) approximations. LUTs are the industry standard for representing color space transformations in printers. Note that the approximations in this section are from CMY to L*a*b*, whereas a printer preprocessor requires the inverse device characteristic, from L*a*b* to CMY. As discussed in the previous section, an LUT approximation of L*a*b* to CMY is more difficult to construct. A comparison of approximations from CMY to L*a*b* is more straightforward to conduct using the data set, and the results illustrate the comparative performances of MINVAR PLs and LUTs.

The Xerox data set is a 21x21x21 uniform grid in CMY color space. Three LUTs were subsampled from the data, with sizes 3x3x3, 6x6x6, and 11x11x11. A validation set, one tenth of the total amount of data, was selected from the points remaining after the subsampling of the 11x11x11 table. All data except the validation data was used for computing the PL approximation using the MINVAR algorithm.

Tetrahedral interpolation is used on the lookup tables [Kan97]. This interpolation scheme breaks up each cube of data in the LUT into six tetrahedra and performs piecewise linear interpolation on these tetrahedra. In this case the LUT is itself a PL function, but with very rigidly located domain vertices and a fixed combinatorial structure. The potential benefit of the MINVAR computed PL approximation derives from the flexibility of moving the vertices to apply more approximation effort in areas that need it.

The MINVAR algorithm is implemented in C++ and includes affinely constrained domain vertices and heuristically guided retriangulation. Documentation on the MINVAR code will appear as a University of Michigan technical report [Gro03]. The MINVAR PL approximations are labeled as PL(i, j, k, l), where i is the number of fixed domain vertices, j is the number of domain vertices constrained to a 1-dimensional affine subspace, k is the number of domain vertices constrained to a 2-dimensional affine subspace, and l is the number of unconstrained domain vertices. The total number of domain (or codomain) vertices is $i + j + k + l$. All codomain vertices are unconstrained. As mentioned in Chapter 4, once MINVAR has converged, the least squares continuous PL approximation can be computed for the final MINVAR triangulation as a post processing step. This appears as the column MV LS in the Table 7.1.

Three different error statistics are presented for each approximation. The first is the

root mean squared error (RMSE) given by

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \|\mathbf{y}_i - f_{\mathcal{P}}(\mathbf{x}_i)\|^2}. \tag{7.1}$$

The second error measure is called $\overline{\Delta \mathrm{E}}$ (read "average ΔE"). ΔE is the Euclidean distance in L*a*b* space, and $\overline{\Delta \mathrm{E}}$ is the average of the ΔE errors, given by

$$\overline{\Delta \mathrm{E}} = \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{y}_i - f_{\mathcal{P}}(\mathbf{x}_i)\|. \tag{7.2}$$

$\overline{\Delta \mathrm{E}}$ is a standard metric used in color science. Note that this quantity will be less than or equal to the RMSE[10]. A ΔE of 1 corresponds roughly to a just noticeable color difference, the smallest difference in color that can be distinguished by an average human observer. The radius of the just noticeable difference sphere actually varies a bit depending on the location in L*a*b* space. The third error measure is E_∞, or the "max" error, given by

$$\mathrm{E}_\infty = \max_{i=1,\ldots N} \|\mathbf{y}_i - f_{\mathcal{P}}(x_i)\|. \tag{7.3}$$

Th E_∞ error measure is important, because sometimes, as in the case of trademark colors, it is necessary to print specific colors very accurately. A small E_∞ error ensures that the color errors are small over the entire space, whereas the other two measures can trade off large errors in one region with very small errors in another.

The results for the numerical experiments are provided in Table 7.1, showing for each approximation the number of vertices, the number of parameters, and the three error statistics of the approximation on the validation data. For MINVAR approximations, the RMSE of the postprocessed approximation, MV LS, is also provided. Visualizations of two MINVAR PL approximations are provided in Figures 7.4 and 7.5. It is instructive to compare the approximations in terms of the number of parameters each approximation has at its disposal and the resulting error statistics on the validation data. For an LUT, the domain vertices are rigidly fixed. The only parameters are the codomain values in the table, and since the codomain is three dimensional, there are three parameters per point. Thus, a $k \times k \times k$ LUT has $3k^3$ parameters. For a MINVAR PL approximation, each codomain vertex has three parameters, but the number of parameters for a domain vertex depends on the dimension of the space in which it is constrained to move: 0 parameters for a fixed vertex (0-dimensional affine subspace), 1 parameter for a 1-dimensional affine subspace, 2 parameters for a a 2-dimensional affine subspace, and 3 parameters for an unconstrained vertex. Thus, the approximation $\mathrm{PL}(i, j, k, l)$ has $j + 2k + 3l + 3(i + j + k + l)$ parameters.

[10]In most fields, RMSE is the preferred over ΔE, since the parameters of the approximation that minimizes RMSE, or equivalently MSE, corresponds to maximum likelihood estimate of those parameters under the assumption of independent identically distributed Gaussian additive noise. Nevertheless $\overline{\Delta \mathrm{E}}$ is the metric of choice in the the color industry.

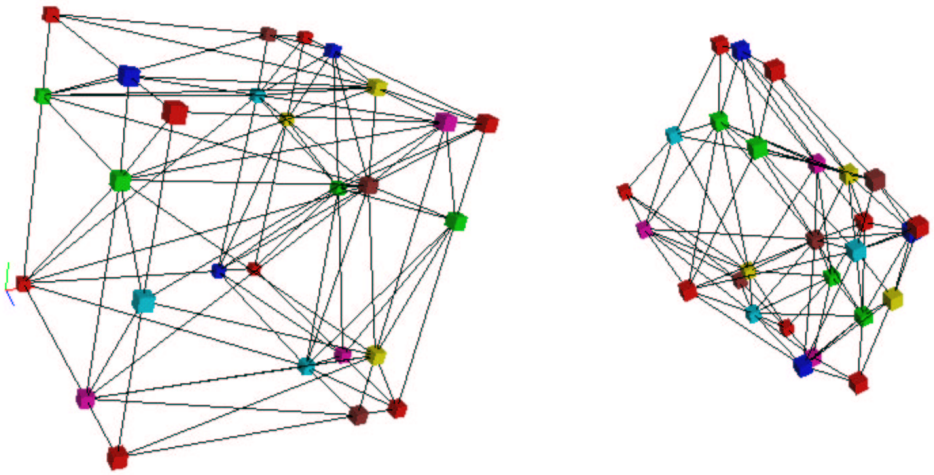

Figure 7.4: PL(8,12,6,1) MINVAR approximation to the color data, with the domain (CMY) on the left and codomain (L*a*b*) on the right. See Table 7.1 for approximation performance statistics. Anecdotally, the color transformation becomes very nonlinear as cyan saturates. This is reflected in the converged positions of the domain vertices being toward the right of the domain, corresponding to high cyan values.

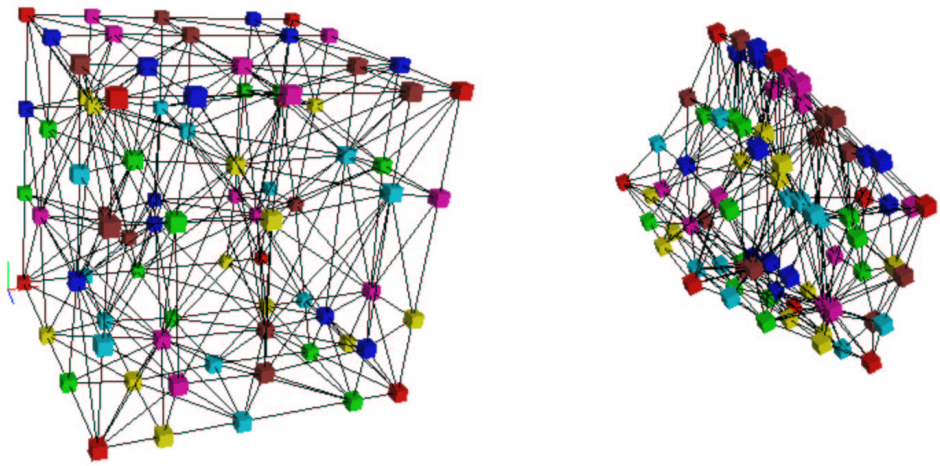

Figure 7.5: PL(8,28,32,12) MINVAR approximation to the color data, with the domain (CMY) on the left and codomain (L*a*b*) on the right. See Table 7.1 for approximation performance statistics. Notice again that the domain vertices have moved toward the right, corresponding to high values of cyan.

Table 7.1: Comparison of various error measures for tetrahedral interpolation LUTs and PL approximations computed by the MINVAR algorithm. PL(i, j, k, l) is a PL function with i vertices fixed in the domain, j vertices constrained to a 1D affine subspace in the domain, k vertices constrained to a 2D affine subspace in the domain, and l freely movable vertices. MV LS RMSE is the RMSE of the least squares approximation on the domain triangulation provided by MINVAR. For the PLs with all domain vertices fixed, errors are reported for the least squares approximation on the fixed domain triangulation.

	Vertices	Parameters	RMSE	MV LS RMSE	$\overline{\Delta E}$	E_∞
PL (27,0,0,0)	27	81	N/A	3.37	2.94	12.4
PL (26,0,0,1)	27	84	3.29	3.25	2.91	10.8
PL (8, 12,6,1)	27	114	2.89	2.84	2.59	9.29
PL (64,0,0,0)	64	192	N/A	2.31	2.05	7.65
PL (56,0,0,8)	64	216	2.06	2.04	1.82	6.39
PL (8,24,24,8)	64	288	1.67	1.63	1.48	6.55
PL (125,0,0,0)	125	375	N/A	1.56	1.38	5.77
PL (98,0,0,27)	125	456	1.50	1.44	1.36	4.23
PL (80,0,0,0)	80	240	N/A	1.88	1.67	6.82
PL (8,28,32,12)	80	364	1.55	1.51	1.38	6.27
PL (8,0,0,1)	9	30	5.01		4.19	16.98
PL (26,0,0,8)	34	126	2.95		2.46	10.59
PL (28,0,0,8)	36	132	2.72		2.31	8.42
PL (28,0,0,9)	37	138	2.73		2.32	8.87
PL (23,4,2,8)	37	143	2.36		2.07	6.89
PL (10,11,11,9)	41	186	1.83		2.07	5.29
LUT 3x3x3	27	81	6.94		6.17	16.92
LUT 6x6x6	216	648	1.31		1.10	3.66
LUT 11x11x11	1331	3993	0.62		0.40	2.35

The benefit of moving the domain vertices is illustrated by comparing the approximations in the top half Table 7.1. There are several groups of PL approximations that use the same number of vertices but allow different numbers of domain vertices to move, such as PL(27,0,0,0), PL(26,0,0,1), and PL(8,12,6,1). These results show that allowing the domain vertices on the convex hull to move reduces the error, at the cost of added parameters. The difference between PL(56,0,0,8) and PL(8,24,24,8) is perhaps the most striking. It is also interesting to compare the PL approximations with 27 vertices with the 3x3x3 LUT. These use the same number of vertices and display a significant differences due to domain vertex movement.

The 6x6x6 and 11x11x11 LUTs have huge numbers of parameters and very good error statistics. The PL approximations approach the 6x6x6 LUT in terms of error while using significantly fewer parameters. No PL approximations, however, have even close to the the number of parameters as the 11x11x11 LUT. This is because MINVAR runs into difficulties when it does not have enough data in each simplex. Consider a PL approximation with $6^3 = 216$ vertices that are initially placed on a uniform grid. The resulting triangulation will have about $6 \cdot 5^3 = 750$ tetrahedra (3-simplices). Since the entire data set, including validation data, has $21^3 = 9261$ points, there will be about 12.3 points in each simplex on average when the algorithm is initialized. As the domain vertices move, some tetrahedra gain data, at the expense of other tetrahedra losing data. Tetrahedra start to crowd into regions of the domain that are curvy, and then tetrahedra no longer have enough data compute the least squares affine approximation in stage 1 of the MINVAR algorithm. Some heuristics are in place to allow MINVAR to continue when a few d-simplices have too little data, but the algorithm runs into problems when this condition becomes too widespread. The competition for data between d-simplices currently places a limit on the number of parameters in the approximant. One of the areas for future work is how to deal with sparse data better.

The PL approximations in the top half of Table 7.1 use a regular grid as their initial triangulation, while the PL approximations in the lower half of the table use less structured initial conditions. Incorporating domain knowledge about color space transformations into the initial conditions admits improvement in performance. Anecdotally, the color space transformation is more complicated along the boundary of the domain. From numerical experience with the Xerox data, it was found that points having the highest error norm were generally on the exterior. Generally the more vertices a simplex has on the boundary, the higher its MSE, suggesting that extra vertices should be put on the boundary to drive down the errors. This was a motivating factor in developing constrained movement for domain vertices. Also anecdotally, there tend to be strong nonlinearities as cyan saturates in the color space transformation from CMY to L*a*b* . Examining the domain vertices in the approximation visualized in Figures 7.4 and 7.5, notice that a plane of the movable vertices position themselves close and reasonably parallel to the plane where cyan saturates. In this sense, the location of the vertices provides some intuitive insight into the structure of the underlying function. Most of the uniform grid initial conditions for MINVAR have a grid of the form $k \times k \times k$. Since it is known that the cyan direction is more complicated in general, PL(80,0,0,0) and PL(8,28,32,12) (the latter is shown in Figure 7.5) use a grid of 5x4x4 vertices, with an extra plane of vertices in the cyan direction. This provides error performance very similar to 5x5x5, but with fewer parameters.

Examining the number of parameters in an approximation is one way of estimating the amount of data needed to compute the approximation, but comparing the number of parameters in a MINVAR PL and LUT approximation is not the only manner of comparison

and not necessarily even the most appropriate. Consider a PL and with the same number of vertices as a corresponding LUT, but with more parameters due to movable domain vertices. It is true that more data will be required to compute the PL approximation initially, but after the initial printer calibration at the time of manufacture using a large amount of data, some or all of the domain vertices in the color space transformation approximation could be fixed. If the MINVAR algorithm finds the major nonlinearities in the transformation and places vertices there during the initial calibration, and if the drift in the color space transformation is mostly local, then it may be sufficient to allow successive approximations to update only the remaining movable vertices, permitting less data to be used when calibrating the printer in the field. If this is the case, then comparing PL and LUT approximation on the basis of vertices rather than parameters may be appropriate.

Many further improvements to these MINVAR PL approximations of color space transformations can be made, both through further development and experimentation with the MINVARalgorithm as well as by incorporating further domain knowledge into the approximation. It is expected that MINVAR PL will perform considerably better in terms of error as compared to LUT for approximations with a similar number of parameters, especially once the limitations in parameterization due to data competition are overcome.

CHAPTER 8

Conclusion

Changes of coordinates play an important role in design and analysis in a wide variety of fields. Finding a good coordinate system can highlight previously unnoticed structure in a problem, for example action-angle coordinates in mechanics [Arn89], the separation property of observers for linear control systems [Che84], and, in a more specific setting, analysis of biped walking [WGK03]. A change of coordinates can permit solutions developed for systems with a particular structure to be applied to other systems related through a change of coordinates, for example pole placement in linear control systems [Che84], and robot navigation [RK91] and visual servoing [CWK02] via artificial potential functions. In other situations, computing a change of coordinates is the itself objective, for example in color systems management (Chapter 7), and in pattern recognition [TY00].

Finding an appropriate change of coordinates for one's problem is in most cases an art [RK91, CWK02]. Once a candidate change of coordinates is written down, it is often difficult to determine whether the candidate is a valid change of coordinates, i.e. one-to-one and onto. Many applications that use changes of coordinates also simultaneously require the inverse change of coordinates, for example to construct a conjugate dynamical system. For most nonlinear changes of coordinates one must resort to computing the inverse numerically, either by constructing an approximation of the inverse or by using an iterative procedure to find the inverse for each query point. Using an approximation to the inverse requires more parameters and more training time. Approximating the inverse online for each query point requires more runtime computation for evaluating the approximation. For some applications, the error between the approximation of the inverse and the inverse of the forward approximation may be as critical as (or even more than) the error between the forward approximation and the data. Numerical inversion introduces a potential source of error which would be absent if the change of coordinates were closed form invertible.

This thesis proposes piecewise linear homeomorphisms (PLH) as a computationally effective, finitely parameterized family of nonlinear changes of coordinates. Though parameterizations of PLHs similar to the one presented in this thesis have been used in algebraic topology since the fifties, they have generally not been used for approximation. PLHs are

a subset of the space of continuous piecewise linear (PL) functions, and the same parameterization is used with PL functions. Given a candidate PL change of coordinates, one can check its invertibility geometrically, and if invertible, the inverse can be computed in closed form. The forward and inverse change of coordinates can thus be parsimoniously represented by a single PL function. One obvious limitation of PLHs is that they do not offer smoothness beyond continuity. They are, nevertheless, a powerful computational tool for representing changes of coordinates.

The focus of this thesis is on computing PLH approximations to a data set drawn from an invertible function, a change of coordinates. The MINVAR algorithm, the tool developed for this purpose, moves the vertices of both the domain and codomain, allowing the PL approximation to place more of its approximation power to areas of the domain that need it. This permits a more parsimonious representation than, say, a traditional lookup table. The geometry underlying PL functions permits invertibility to be checked and to invert it in closed form, but the geometry also causes complications. Triangulation tangles are a difficult problem to overcome. The λ regularization parameter in MINVAR is used to slow down the movement of vertices in order to prevent domain triangulation tangles, but currently there is no principled way to choose λ. It is hand picked based on the data set and to a lesser extent on the initial condition. In the short term, one can imagine many methods to adapt λ in order to guarantee a valid domain triangulation. In the long term, a principled way to avoid tangles may result from a better understanding of the configuration space of triangulations, the space of valid placements of vertices for a given combinatorial structure. Computational geometry research on triangulations in higher dimensions is still actively progressing [Ede01, ES96], so these sorts of results may emerge soon. Very little is understood about the interaction between the combinatorial and continuous parameters when approximating a function. When and how should retriangulation be performed? What do triangulation tangles indicate about the data generating function? What is the best way to ensure that a PL function is a PLH at each iteration? Hopefully this thesis will serve as the groundwork to address these and the many other questions.

The color systems management problem was the initial motivation for approximating changes of coordinates from data, and hence for interest in PLH approximations to data. A preliminary numerical study on a set of simulated color data yielded encouraging results. MINVAR-generated PLH approximations compare favorably to the current industry standard, lookup tables, providing better approximations with a more parsimonious parameterization. Further study is required, but the author and his collaborators at the University of Michigan and Xerox remain cautiously optimistic that these methods may have eventual impact on the color printing industry.

As the rewritten PL toolbox in C++ becomes available later this year, these PL approximation techniques can be more broadly tested. MINVAR has been preliminarily applied in several other applications not reported in this thesis, including approximation of lens

skew in camera calibration, of nonlinear spring models from experimental data, of time T maps of differential equations, specifically the Van der Pol oscillator, and of the stance map of the spring loaded inverted pendulum [SK00, Sch98]. Stance maps for more complicated models of legged systems are of immediate interest. It will be exciting to see what other applications emerge.

MINVAR could benefit from a more automated method to generate initial PL approximations. A triangulation of the domain is required in order to generate the initial approximation. If the domain is specified explicitly, e.g. as a hypercube, then it is reasonably straightforward to place vertices appropriately, for example in a grid. Triangulation software can be used to generate the initial triangulation of these vertices, though the software must be able to handle degeneracies in the vertex set. In many cases, however, the domain is not defined explicitly, but rather implicitly by the region from which data is available. One cannot make an overly large estimate of the domain, since MINVAR has difficulty with domain simplices that lack data, so the domain of the initial PL approximant must fit snugly around the data. The convex hull of the data could be used to define the domain, but this presents two immediate concerns. First, the data set might not be convex, so large regions of the domain might be left without data. Second, using the convex hull of the data as the domain may result in many fixed vertices on the domain boundary, whereas defining the domain with fewer vertices placed slightly away from the hull of the data may be more effective. Currently the initial placement of vertices is performed by hand, but automated techniques and heuristics could greatly improve the utility of the MINVAR algorithm, especially in high dimensions.

A nonparametric version of MINVAR would add and remove vertices from the approximation in order to improve the fit to the data set. This is related to automatic generation of initial conditions since both place vertices in the PL approximant. The major addition for a nonparametric version is an appropriate set of heuristics for adding and removing vertices. The author suspects that a nonparametric version may help mitigate the competition among simplices for data, discussed in the numerical results of Chapter 7.

An online (i.e. realtime) version on MINVAR is desired both for the color printing problem as well as other applications. Using the current batch version of MINVAR, a printer could occasionally update its approximation of the forward device characteristic, but printing all the test patches at once would occupy the printer for some time. An online algorithm could frequently update its approximation, printing test patches when it gets the chance, and would allow the printer to react to process drift more rapidly. Other applications, such as robot navigation, could benefit from an online algorithm. In the navigation problem, the change of coordinates between the observed and model environments could be constructed as the robot navigates. With an online version of MINVAR, the inverse of the approximation would be immediately available, thanks to the closed form invertibility of PL functions, whereas other approximation techniques would generally require a numerical inverse, which

would take additional time to compute.

There are several possible approaches to developing online algorithms for PL approximation. The easiest approach would be to mix the batch version of MINVAR with standard linear-in-parameters online approximation techniques. The batch version of MINVAR could be applied to a large set of data initially. Then, fixing the domain vertices of the MINVAR generated approximation would make the approximant linear-in-parameters, permitting application of standard techniques for adaptive approximation [Lju99]. This may work well if the data generating function does not change too dramatically over time. In essence, MINVAR is used in this case to adapt a set of basis functions to the data generating function, then standard methods are used once these basis functions are determined. On the downside, this technique fixes the domain vertices during the online phase, though numerical work with MINVAR indicates that moving the domain vertices has a substantial effect on improving the approximation. Developing an online algorithm that allows domain vertices of the PL approximation to move is more difficult. One approach would be to save all past data, and at each step compute a new batch approximation using the MINVAR algorithm. This approach may be feasible since the MINVAR algorithm runs in a relatively short amount of time, but it goes against the spirit of what is usually meant by "online algorithm," since this approach is not incremental. That is, the longer data is collected, the longer each update would take as more and more data accumulates. If the data is not stored, then computing the least squares affine approximations in stage 1 of MINVAR is tricky. While there are recursive least squares algorithms that permit incremental updates to an affine approximation, it is not clear, without explicitly storing the data, how to account for data points that would be in different domain simplices after an iteration of the MINVAR algorithm due to domain vertex movement. This is a chief challenge in generating an online version of MINVAR.

Another interesting direction for exploration is PL pseudoinverses. Many sensor fusion problems can be posed as a system where a d-dimensional state maps to a c-dimensional sensor measurement, $c > d$. The pseudoinverse, from the c-dimensional measurement space to the d-dimensional state space, can be used to estimate the system's state given the sensor measurements. If a one-to-one PL function maps from a d-dimensional space to a c-dimensional space, $c > d$, then the image of the PL function is a d-dimensional triangulation embedded in the c-dimensional space. An appropriate geometric projection from the c-dimensional space onto the d-dimensional triangulation provides a method of computing a pseudoinverse to the PL function. To compute the pseudoinverse, project to the triangulation and then find the true inverse for that point. The main tasks in this line of work would be to find geometric projections which are efficiently computable and to determine which type of projection would work best on sensor fusion problems.

The MINVAR algorithm constructs a PL approximation to a set of explicit input-output data. In many applications one knows only that a change of coordinates exists that pro-

vides specific structure to the problem, without having a method to construct or to evaluate directly the change of coordinates. For example, one may know that a given control system *can* be feedback linearized without knowing an appropriate change of coordinates and feedback to do so. This raises an approximation problem of a different sort, where input-output data is not directly available, but rather one wishes to find a change of coordinates that approximately changes a given system into a system that has some desired structural property. In the feedback linearization case, the quality of the change of coordinates is reflected in how close to linear the transformed system is. As another example, some nonconvex optimization problems can be transformed into convex optimization problems through an appropriate change of coordinates, but again input-output data is not directly available, but rather the quality of the approximation is gauged by how close to convex the transformed optimization problem is. The representation of PLHs developed in this thesis could be put to use these contexts, but requires development of entirely new algorithms.

PLHs lack smoothness beyond continuity, but smoother approximations are desired or required for many applications. In some practical settings a bounded Lipschitz constant would be as good as differentiability. A clever method of simultaneously refining the domain and codomain triangulations of a PL function could be used to find a nearby PL function with more vertices but with a lower Lipschitz constant. Alternatively, a higher order spline could be use to approximate a PL function on a refined domain triangulation. By refining the domain triangulation enough, one should be able to find a spline that is arbitrarily close to the original PL function. If the spline approximation is close enough to the PLH, then the spline will also be invertible, since invertibility is an open property.

While PLHs have been around for a long time, they seem to have been overlooked in the field of approximation as a computationally effective way to represent multidimensional nonlinear changes of coordinates. This thesis represents only the preliminary stages of the application of PLH representations in systems engineering, and there remain many lines of research to explore.

APPENDICES

APPENDIX A

Geometry and Triangulations

This appendix contains proofs of the claims from Chapter 2. A deep understanding of the underlying geometry is vital to understanding piecewise linear functions, so proofs are provided even for the first few claims of Chapter 2, which are standard results in the field of convex geometry. Also included is a comparison of simplicial complexes as defined in this dissertation, simplicial complexes as defined in traditional combinatorial topology, and δ-complexes, the modern algebraic topological generalization of simplicial complexes. The structure of this appendix roughly parallels that of Chapter 2.

A.1 Geometry

Claim (Restatement of 2.1). $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_m \in \mathbb{R}^d$ *are affinely independent if and only if the vectors* $\mathbf{p}_1 - \mathbf{p}_m, \mathbf{p}_2 - \mathbf{p}_m, \ldots, \mathbf{p}_{m-1} - \mathbf{p}_m$ *are linearly independent.*

Proof. Let $\mathbf{p}_1, \ldots, \mathbf{p}_m$ be affinely independent. Let $L = \operatorname{span}\{\mathbf{p}_1 - \mathbf{p}_m, \mathbf{p}_2 - \mathbf{p}_m, \ldots, \mathbf{p}_{m-1} - \mathbf{p}_m\}$. From linear algebra $\dim L \leq m$. It must be that $\dim L = m$, because otherwise $L + \mathbf{p}_m$ would be an affine subspace of dimension less than m containing $m+2$ points, contradicting that $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_m$ are affinely independent. It follows that $\mathbf{p}_1 - \mathbf{p}_m, \mathbf{p}_2 - \mathbf{p}_m, \ldots, \mathbf{p}_{m-1} - \mathbf{p}_m$ are linearly independent.

Now let $\mathbf{p}_1 - \mathbf{p}_m, \mathbf{p}_2 - \mathbf{p}_m, \ldots, \mathbf{p}_{m-1} - \mathbf{p}_m$ be linearly independent. Let L be a k-dimensional linear subspace and $\mathbf{x} \in \mathbb{R}^d$. Let $\mathbf{p}_{i_1}, \ldots, \mathbf{p}_{i_n}$ be the n points from $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_m$ contained in the affine subspace $L + \mathbf{x}$. Since $\mathbf{p}_{i_n} \in L + \mathbf{x}$, then $L + \mathbf{p}_{i_n} = L + \mathbf{x}$. Since $\mathbf{p}_1 - \mathbf{p}_m, \mathbf{p}_2 - \mathbf{p}_m, \ldots, \mathbf{p}_{m-1} - \mathbf{p}_m$ are linearly independent, so are $\mathbf{p}_{i_1} - \mathbf{p}_{i_n}, \mathbf{p}_{i_2} - \mathbf{p}_{i_n}, \ldots, \mathbf{p}_{i_{n-1}} - \mathbf{p}_{i_n}$. From linear algebra it follows that $n \leq k+1$. Thus $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_m$ are affinely independent. □

Claim (Restatement of 2.2). *Let* $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_m \in \mathbb{R}^d$. *Then* $\mathbf{p}_1, \ldots, \mathbf{p}_m$ *are affinely independent if and only if*

$$\sum_{i=1}^{m} \alpha_i \mathbf{p}_i = 0 \quad \textit{and} \quad \sum_{i=1}^{m} \alpha_i = 0 \qquad \textit{where} \;\; \alpha_i \in \mathbb{R} \tag{A.1}$$

holds only when $\alpha_i = 0$ for all i, or equivalently in matrix notation if

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \cdots & \mathbf{p}_m \\ 1 & 1 & \cdots & 1 \end{bmatrix} \tag{A.2}$$

has a trivial null space.

Proof. From Claim 2.1, $\mathbf{p}_1, \ldots, \mathbf{p}_m$ are affinely independent if and only if $\mathbf{p}_1 - \mathbf{p}_m, \mathbf{p}_2 - \mathbf{p}_m, \ldots, \mathbf{p}_{m-1} - \mathbf{p}_m$ are linearly independent. The matrix $\mathbf{PA}$,

$$\begin{aligned} \mathbf{PA} &= \begin{bmatrix} \mathbf{p}_1 - \mathbf{p}_m & \mathbf{p}_2 - \mathbf{p}_m & \cdots & \mathbf{p}_{m-1} - \mathbf{p}_m & \mathbf{p}_m \\ 0 & 0 & & 0 & 1 \end{bmatrix}, \\ \text{where } \mathbf{A} &= \begin{bmatrix} \mathbf{I}_m & \mathbf{0}_{m\times 1} \\ -1 \cdots -1 & 1 \end{bmatrix} \end{aligned}$$

is full rank if and only if $\mathbf{p}_1 - \mathbf{p}_m, \mathbf{p}_2 - \mathbf{p}_m, \ldots, \mathbf{p}_{m-1} - \mathbf{p}_m$ are linearly independent. Since $\mathbf{A}$ is full rank, the matrix $\mathbf{PA}$ is full rank if and only if $\mathbf{P}$ is full rank, and $\mathbf{P}$ is full rank if and only if $\mathbf{P}$ has a trivial nullspace. □

Claim (Restatement of 2.3). *The affine set generated by $\mathbf{p}_1, \ldots, \mathbf{p}_m \in \mathbb{R}^d$ is the affine hull of the points,* $\text{aff}\{\mathbf{p}_1, \ldots, \mathbf{p}_m\}$.

Proof. First we show that every point in the affine subspace $L + \mathbf{p}_m$, where $L = \text{span}\{\mathbf{p}_1 - \mathbf{p}_m, \ldots, \mathbf{p}_{m-1} - \mathbf{p}_m\}$ is generated by an affine combination of $\mathbf{p}_1, \ldots, \mathbf{p}_m$. Let $\mathbf{x} \in L$ and let β_i be coefficients such that $\mathbf{x} = \sum_{i=1}^{m-1} \beta_i(\mathbf{p}_i - \mathbf{p}_m)$. Let $\alpha_i = \beta_i$ for $i = 1, \ldots, m-1$ and $\alpha_m = 1 - \sum_{i=1}^{m-1} \alpha_i$. Then the affine combination $\sum_{i=1}^{m} \alpha_i \mathbf{p}_i = \mathbf{x} + \mathbf{p}_m$. Thus, any point in $L + \mathbf{p}_m$ is an affine combination of $\mathbf{p}_1, \ldots, \mathbf{p}_m$.

It remains to show that no smaller affine subspace than $L + \mathbf{p}_m$ contains $\mathbf{p}_1, \ldots, \mathbf{p}_m$. Any affine subspace containing $\mathbf{p}_1, \ldots, \mathbf{p}_m$ can be written as $L' + \mathbf{p}_m$, where L' is a linear subspace. Since $\mathbf{p}_1, \ldots, \mathbf{p}_m$ are contained in $L + \mathbf{p}_m$, we can restrict our attention to $L' \subset L$. If $\dim L' < \dim L$, then there exists i, $1 \le i \le m-1$, such that $\mathbf{p}_i - \mathbf{p}_m \notin L'$, but this implies that $\mathbf{p}_i \notin L' + \mathbf{p}_m$, which is a contradiction. Thus $\dim L' = \dim L$, and $L + \mathbf{p}_m$ is the affine hull of $\mathbf{p}_1, \ldots, \mathbf{p}_m$. □

A.1.1 Barycentric Coordinates

Claim (Restatement of 2.4). *Let $\mathbf{p}_1, \ldots, \mathbf{p}_{k+1} \in \mathbb{R}^d$ be affinely independent. Let $\mathbf{x} \in \text{aff}\{\mathbf{p}_1, \ldots, \mathbf{p}_{k+1}\}$. The barycentric coordinates of $\mathbf{x}$ with respect to $\mathbf{p}_1, \ldots, \mathbf{p}_{k+1}$ are unique.*

Proof. Let $\mathbf{x} \in \text{aff}\{\mathbf{p}_1, \ldots, \mathbf{p}_{k+1}\}$. Let $\alpha = [\alpha_1 \cdots \alpha_{k+1}]^{\mathrm{T}}$ and $\beta = [\beta_1 \cdots \beta_{k+1}]^{\mathrm{T}}$ be barycentric coordinates for $\mathbf{x}$, that is $x = \sum_{i=1}^{k+1} \alpha_i \mathbf{p}_i = \sum_{i=1}^{k+1} \beta_i \mathbf{p}_i$ and $\sum_{i=1}^{k+1} \alpha_i = \sum_{i=1}^{k+1} \beta_i = 1$. Let

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \cdots & \mathbf{p}_{k+1} \\ 1 & 1 & & 1 \end{bmatrix}.$$

Then we may equivalently write that $\mathbf{x} = \mathbf{P}\alpha = \mathbf{P}\beta$. Rewriting gives $\mathbf{P}(\alpha - \beta) = 0$. Since $\mathbf{p}_1, \ldots, \mathbf{p}_{k+1}$ are affinely independent, by Claim 2.1 the matrix $\mathbf{P}$ has a nontrivial nullspace. Thus, $\alpha = \beta$, and the barycentric coordinates of $\mathbf{x}$ with respect to $\mathbf{p}_1, \ldots, \mathbf{p}_{k+1}$ are unique. □

Claim (Restatement of 2.5). *Let $\mathbf{p}_1, \ldots, \mathbf{p}_{d+1} \in \mathbb{R}^d$ be affinely independent. Let $\mathbf{x} \in \mathbb{R}^d$. The barycentric coordinates of $\mathbf{x}$ with respect to $\mathbf{p}_1, \ldots, \mathbf{p}_{d+1}$ are given by*

$$\boldsymbol{\alpha} = \mathbf{P}^{-1} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}, \tag{A.3}$$

where

$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \cdots \\ \alpha_{d+1} \end{bmatrix}, \qquad \mathbf{P} = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \cdots & \mathbf{p}_{d+1} \\ 1 & 1 & & 1 \end{bmatrix}. \tag{A.4}$$

Proof. Since $\mathbf{p}_1, \ldots, \mathbf{p}_{d+1}$ are affinely independent, the matrix $\mathbf{P}$ is invertible as a result of Claim 2.1. Let $\boldsymbol{\alpha} = [\alpha_1 \cdots \alpha_{d+1}]^{\mathrm{T}}$ be given by Equation A.3. Equivalently, $\mathbf{P}\boldsymbol{\alpha} = [\mathbf{x}^{\mathrm{T}}\ 1]^{\mathrm{T}}$. The bottom row of this equation is $\sum_{i=1}^{d+1} \alpha_i = 1$. That is, $\sum_{i=1}^{d+1} \alpha_i \mathbf{p}_i$ is an affine combination. Moreover,

$$\begin{aligned} \begin{bmatrix} \mathbf{p}_1 & \cdots & \mathbf{p}_{d+1} \end{bmatrix} \boldsymbol{\alpha} &= \begin{bmatrix} \mathbf{p}_1 & \cdots & \mathbf{p}_{d+1} \end{bmatrix} \mathbf{P}^{-1} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{I}_d & \mathbf{0}_{d\times 1} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \\ &= \mathbf{x}. \end{aligned}$$

Thus $\alpha_1, \ldots, \alpha_{d+1}$ are the barycentric coordinates of $\mathbf{x}$ with respect to $\mathbf{p}_1, \ldots, \mathbf{p}_{d+1}$. □

Claim (Restatement of 2.6). *Let $\mathbf{x} \in \mathbb{R}^d$. Let $\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 & \cdots & \alpha_{d+1} \end{bmatrix}^{\mathrm{T}}$ be the barycentric coordinates of $\mathbf{x}$ with respect to the affinely independent points $\mathbf{p}_1, \ldots, \mathbf{p}_{d+1} \in \mathbb{R}^d$. Then $\delta_s(\mathbf{x}, H_i) = \alpha_i \delta(\mathbf{p}_i, H_i)$.*

Proof. First we establish that $\delta(\mathbf{x}, H_i) = |\alpha_i|\ \delta(\mathbf{p}_i, H_i)$, and then show that the sign of α_i indicates the side of H_i on which $\mathbf{p}_i$ lies, giving the desired result. Let $(\mathbf{a}_i, c_i)$ be an implicit representation of H_i. Then

$$\begin{aligned} \delta(\mathbf{x}, H_i) &= \frac{\left|\mathbf{a}_i{}^{\mathrm{T}}\mathbf{x} + c_i\right|}{\|\mathbf{a}_i\|} \\ &= \frac{1}{\|\mathbf{a}_i\|} \left| \mathbf{a}_i{}^{\mathrm{T}} \left(\sum_{j=1}^{d+1} \alpha_j \mathbf{p}_j \right) + c_i \right| \\ &= \left| \sum_{j=1}^{d+1} \alpha_j \left(\frac{\mathbf{a}_i{}^{\mathrm{T}}\mathbf{p}_j + c_i}{\|\mathbf{a}_i\|} \right) \right| \end{aligned}$$

For $j \neq i$, $\mathbf{p}_j \in H_i$, which implies that ${\mathbf{a}_i}^{\mathrm{T}}\mathbf{p}_j + c_i = 0$. It follows that

$$\delta(\mathbf{x}, H_i) = |\alpha_i| \;\; \delta(\mathbf{p}_i, H_i)\,.$$

Since *i*) for a given $\mathbf{x}$, $\alpha_i = 0$ if and only if $x \in H_i$, *ii*) for $\mathbf{x} = \mathbf{p}_i$, $\alpha_i = 1$, and *iii*) α_i is an affine function of $\mathbf{x}$ by Claim 2.5, it follows that $\alpha_i > 0$ for $\mathbf{x}$ on the same side of H_i as $\mathbf{p}_i$, and $\alpha_i < 0$ for $\mathbf{x}$ on the opposite side of H_i as $\mathbf{p}_i$. Thus, $\delta_s(\mathbf{x}, H_i) = \alpha_i \delta(\mathbf{p}_i, H_i)$. □

Claim (Restatement of 2.7). *Let* $\mathbf{x} \in \mathbb{R}^d$. *Let* $\boldsymbol{\alpha} = \begin{bmatrix}\alpha_1 & \cdots & \alpha_{d+1}\end{bmatrix}^{\mathrm{T}}$ *be barycentric coordinates of* $\mathbf{x}$ *with respect to the affinely independent points* $\mathbf{p}_1, \ldots, \mathbf{p}_{d+1}$. *The distance from* $\mathbf{x}$ *to the affine subspace* $A = \mathrm{aff}(\{\mathbf{p}_1, \ldots \mathbf{p}_k\})$, *is given by* $\delta(\mathbf{x}, A) = {\bar{\boldsymbol{\alpha}}_s}^{\mathrm{T}} G \bar{\boldsymbol{\alpha}}_s$, *where* $\bar{\boldsymbol{\alpha}}_s = \begin{bmatrix}\alpha_{k+1} & \alpha_{k+2} & \cdots & \alpha_{d+1}\end{bmatrix}^{\mathrm{T}}$ *and* $G \in \mathbb{R}^{(d-k+1)\times(d-k+1)}$, *defined in Equation A.6, is a positive definite matrix whose entries depend only on* $\mathbf{p}_1, \ldots, \mathbf{p}_k$.

Proof. The distance of a point to a linear subspace can be computed using Gram determinants [Lue69]. A translation by $\mathbf{p}_k$ turns the affine subspace A into a linear subspace. Let $\tilde{A} = A - \mathbf{p}_k$. Let $\tilde{\mathbf{p}}_i = \mathbf{p}_i - \mathbf{p}_k$ for $i = 1, \ldots, d+1$. Let $\tilde{\mathbf{x}} = \mathbf{x} - \mathbf{p}_k$, and note that

$$\tilde{\mathbf{x}} = \left(\sum_{i=1}^{d+1} \alpha_i \mathbf{p}_i\right) - \mathbf{p}_k = \sum_{i=1}^{d+1} \alpha_i (\mathbf{p}_i - \mathbf{p}_k) = \sum_{i=1}^{d+1} \alpha_i \tilde{\mathbf{p}}_i. \tag{A.5}$$

Let

$$g(\mathbf{y}, \mathbf{z}) = \begin{vmatrix} {\tilde{\mathbf{p}}_1}^{\mathrm{T}}\tilde{\mathbf{p}}_1 & {\tilde{\mathbf{p}}_2}^{\mathrm{T}}\tilde{\mathbf{p}}_1 & \cdots & {\tilde{\mathbf{p}}_{k-1}}^{\mathrm{T}}\tilde{\mathbf{p}}_1 & y^{\mathrm{T}}\tilde{\mathbf{p}}_1 \\ {\tilde{\mathbf{p}}_1}^{\mathrm{T}}\tilde{\mathbf{p}}_2 & {\tilde{\mathbf{p}}_2}^{\mathrm{T}}\tilde{\mathbf{p}}_2 & & {\tilde{\mathbf{p}}_{k-1}}^{\mathrm{T}}\tilde{\mathbf{p}}_2 & y^{\mathrm{T}}\tilde{\mathbf{p}}_2 \\ \vdots & & & & \vdots \\ {\tilde{\mathbf{p}}_1}^{\mathrm{T}}\tilde{\mathbf{p}}_{k-1} & {\tilde{\mathbf{p}}_2}^{\mathrm{T}}\tilde{\mathbf{p}}_{k-1} & & {\tilde{\mathbf{p}}_{k-1}}^{\mathrm{T}}\tilde{\mathbf{p}}_{k-1} & \mathbf{y}^{\mathrm{T}}\tilde{\mathbf{p}}_{k-1} \\ {\tilde{\mathbf{p}}_1}^{\mathrm{T}}\mathbf{z} & {\tilde{\mathbf{p}}_2}^{\mathrm{T}}\mathbf{z} & \cdots & {\tilde{\mathbf{p}}_{k-1}}^{\mathrm{T}}\mathbf{z} & \mathbf{y}^{\mathrm{T}}\mathbf{z} \end{vmatrix},$$

$$b = \begin{vmatrix} {\tilde{\mathbf{p}}_1}^{\mathrm{T}}\tilde{\mathbf{p}}_1 & {\tilde{\mathbf{p}}_2}^{\mathrm{T}}\tilde{\mathbf{p}}_1 & \cdots & {\tilde{\mathbf{p}}_{k-1}}^{\mathrm{T}}\tilde{\mathbf{p}}_1 \\ {\tilde{\mathbf{p}}_1}^{\mathrm{T}}\tilde{\mathbf{p}}_2 & {\tilde{\mathbf{p}}_2}^{\mathrm{T}}\tilde{\mathbf{p}}_2 & & {\tilde{\mathbf{p}}_{k-1}}^{\mathrm{T}}\tilde{\mathbf{p}}_2 \\ \vdots & & & \vdots \\ {\tilde{\mathbf{p}}_1}^{\mathrm{T}}\tilde{\mathbf{p}}_{k-1} & {\tilde{\mathbf{p}}_2}^{\mathrm{T}}\tilde{\mathbf{p}}_{k-1} & \cdots & {\tilde{\mathbf{p}}_{k-1}}^{\mathrm{T}}\tilde{\mathbf{p}}_{k-1} \end{vmatrix}.$$

The quantity $g(\mathbf{y}, \mathbf{y})$ is the Gram determinant for $\tilde{\mathbf{p}}_1, \ldots, \tilde{\mathbf{p}}_{k-1}, \mathbf{y}$, while b is the Gram determinant for $\tilde{\mathbf{p}}_1, \ldots, \tilde{\mathbf{p}}_{k-1}$. The squared distance from $\mathbf{x}$ to A, $\delta^2(\mathbf{x}, A)$, is given by the ratio of Gram determinants

$$\delta^2(\mathbf{x}, A) = \delta^2\left(\tilde{\mathbf{x}}, \tilde{A}\right) = \frac{g(\tilde{\mathbf{x}}, \tilde{\mathbf{x}})}{b}.$$

Using Equation A.5, $\delta^2(\mathbf{x}, A)$ can be rewritten as

$$\delta^2(\mathbf{x}, A) = \frac{1}{b} \sum_{i=1}^{d+1} \sum_{j=1}^{d+1} g(\tilde{\mathbf{p}}_i, \tilde{\mathbf{p}}_j) \;\; \alpha_i \alpha_j$$

Notice that g is symmetric in its arguments, $g(\mathbf{y}, \mathbf{z}) = g(\mathbf{z}, \mathbf{y})$. Also, for $1 \leq i \leq k$, $g(\tilde{\mathbf{p}}_i, \mathbf{y}) = 0$. Thus

$$\delta^2(\mathbf{x}, A) = \frac{1}{b} \sum_{i=k+1}^{d+1} \sum_{j=k+1}^{d+1} g(\tilde{\mathbf{p}}_i, \tilde{\mathbf{p}}_j)\ \alpha_i \alpha_j = \bar{\boldsymbol{\alpha}}_s{}^{\mathrm{T}} G \bar{\boldsymbol{\alpha}}_s$$

where G is given by

$$G = \frac{1}{b} \begin{bmatrix} g(\tilde{\mathbf{p}}_{k+1}, \tilde{\mathbf{p}}_{k+1}) & g(\tilde{\mathbf{p}}_{k+2}, \tilde{\mathbf{p}}_{k+1}) & \cdots & g(\tilde{\mathbf{p}}_{d+1}, \tilde{\mathbf{p}}_{k+1}) \\ g(\tilde{\mathbf{p}}_{k+1}, \tilde{\mathbf{p}}_{k+2}) & & & g(\tilde{\mathbf{p}}_{d+1}, \tilde{\mathbf{p}}_{k+2}) \\ \vdots & & & \vdots \\ g(\tilde{\mathbf{p}}_{k+1}, \tilde{\mathbf{p}}_{d+1}) & g(\tilde{\mathbf{p}}_{k+2}, \tilde{\mathbf{p}}_{d+1}) & \cdots & g(\tilde{\mathbf{p}}_{d+1}, \tilde{\mathbf{p}}_{d+1}) \end{bmatrix}. \tag{A.6}$$

It remains to show that G is positive definite. Consider $\mathbf{x}$ such that $\alpha_i \neq 0$, for some $i = k+1, \ldots, d+1$. By the previous claim, x must lie some nonzero distance away from H_i. Since $A \subseteq H_i$, $\delta(\mathbf{x}, A) > 0$. Thus, G must be positive definite. □

A.1.2 Convexity and Simplices

Claim (Restatement of 2.8). *Let $S \subset \mathbb{R}^d$. Let A be the set of all convex combinations of points in S,*

$$A = \left\{ \mathbf{x} \in \mathbb{R}^d \,\middle|\, \mathbf{x} = \sum_{i=1}^{k} \alpha_i \mathbf{p}_i,\ \sum_{i=1}^{k} \alpha_i = 1\ ,\ \alpha_i \geq 0,\ \mathbf{p}_i \in S \right\}. \tag{A.7}$$

Then $A = \mathrm{conv}(S)$.

Proof. (Adapted from [Web94]) First we show that $\mathrm{conv}(S) \subset A$. Clearly $S \subset A$, so it is sufficient to show that A is convex. Consider points $\mathbf{u}, \mathbf{v} \in A$ with convex combinations $\mathbf{u} = \sum_{i=1}^{m} \beta_i \mathbf{u}_i$ and $\mathbf{v} = \sum_{i=1}^{n} \gamma_i \mathbf{v}_i$, where the $\mathbf{u}_i, \mathbf{v}_i \in S$. Let $\alpha \in [0, 1]$. Then $\alpha\mathbf{u} + (1-\alpha)\mathbf{v} = \alpha \sum_{i=1}^{m} \beta_i \mathbf{u}_i + (1-\alpha) \sum_{i=1}^{n} \gamma_i \mathbf{v}_i$ is a convex combination of points from S. It follows that $[\mathbf{u}, \mathbf{v}] \subset A$ whenever $\mathbf{u}, \mathbf{v} \in A$, and thus A is convex. Since $\mathrm{conv}(S)$ is the smallest convex set containing S, $\mathrm{conv}(S) \subset A$.

Next we show that $A \subset \mathrm{conv}(S)$. Let $\mathbf{u} \in A$ be expressed as the convex combination $\mathbf{u} = \sum_{i=1}^{m} \alpha_i \mathbf{u}_i$ where $\mathbf{u}_i \in S$. Without loss of generality $\alpha_i > 0$ (otherwise drop the corresponding terms from the sum). We will prove that $\mathbf{u} \in \mathrm{conv}(S)$ by induction on m. If $m = 1$, clearly $\mathbf{u} \in S \subset \mathrm{conv}(S)$. Now assume that a convex combination of $m-1$ points in S is in $\mathrm{conv}(S)$, and consider a convex combination of m, $\mathbf{u} = \sum_{i=1}^{m} \alpha_i \mathbf{u}_i$, where $\mathbf{u}_i \in S$. This may be rewritten as $\mathbf{u} = \alpha_m \mathbf{u}_m + (1-\alpha_m)\mathbf{p}$, where $\mathbf{p} = \sum_{i=1}^{m-1} \frac{\alpha_i}{1-\alpha_m} \mathbf{u}_i$. Since $\sum_{i=1}^{m-1} \frac{\alpha_i}{1-\alpha_m} = 1$, $\mathbf{p}$ is a convex combination of $m-1$ points in S, and thus $\mathbf{p} \in \mathrm{conv}(S)$ by the induction hypothesis. Since $\mathbf{u}_m, \mathbf{p} \in \mathrm{conv}(S)$, and $\mathbf{u} \in [\mathbf{u}_m, \mathbf{p}]$ and $\mathrm{conv}(S)$ is convex, it follows that $\mathbf{u} \in \mathrm{conv}(S)$. Thus $A \subset \mathrm{conv}(S)$.

Thus $A = \mathrm{conv}(S)$. □

Claim (Restatement of 2.9). *Let* $\mathbf{p}_1, \ldots, \mathbf{p}_{k+1} \in \mathbb{R}^d$ *be affinely independent. The simplex* $s = \text{conv}(\{\mathbf{p}_1, \ldots, \mathbf{p}_{k+1}\})$ *has the vertices* $\text{vert}(s) = \{\mathbf{p}_1, \ldots, \mathbf{p}_{k+1}\}$.

Proof. Let $\mathbf{z} \in s$, and let $\boldsymbol{\alpha}$ be the barycentric coordinates of $\mathbf{z}$ with respect to $\mathbf{p}_1, \ldots, \mathbf{p}_{d+1}$. Since $\mathbf{z} \in s$ it follows that $\boldsymbol{\alpha} \boldsymbol{\Delta}_k$. We will show that $\mathbf{z}$ can only be an extreme point of s if $\mathbf{z} \in \{\mathbf{p}_1, \ldots, \mathbf{p}_{d+1}\}$.

Let $\mathbf{x}, \mathbf{y} \in s$ be such that $\frac{1}{2}(\mathbf{x} + \mathbf{y}) = \mathbf{z}$. Let $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$ be the barycentric coordinates of $\mathbf{x}$ and $\mathbf{y}$ respectively. Since $\mathbf{x}, \mathbf{y} \in s$, then $\boldsymbol{\beta}, \boldsymbol{\gamma} \in \boldsymbol{\Delta}_k$. We then have the relationship in coordinates $\boldsymbol{\alpha} = \frac{1}{2}(\boldsymbol{\beta} + \boldsymbol{\gamma})$, with $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma} \in \boldsymbol{\Delta}_k$. By inspection, this implies that $\boldsymbol{\alpha} = \boldsymbol{\beta} = \boldsymbol{\gamma}$ only if all components of $\boldsymbol{\alpha}$ are zero except for one that is 1. It follows that the extreme points of s are $\mathbf{p}_1, \ldots, \mathbf{p}_{k+1}$.

□

Claim (Restatement of 2.10). *Let* $\mathbf{p}_1, \ldots, \mathbf{p}_{d+1} \in \mathbb{R}^d$ *be affinely independent. The signed volume of the simplex* $\bar{s} = \text{conv}(\{\mathbf{p}_1, \ldots, \mathbf{p}_{d+1}\})$ *is* $V(\bar{s}) = \frac{1}{d!} \det \tilde{\mathbf{P}} = \frac{1}{d!} \det \mathbf{P}$*, where*

$$\tilde{\mathbf{P}} = \begin{bmatrix} \mathbf{p}_1 - \mathbf{p}_{d+1} & \mathbf{p}_2 - \mathbf{p}_{d+1} & \cdots & \mathbf{p}_d - \mathbf{p}_{d+1} \end{bmatrix} \qquad \mathbf{P} = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \cdots & \mathbf{p}_{d+1} \\ 1 & 1 & & 1 \end{bmatrix}.$$

Proof. The change of coordinates $\bar{\mathbf{x}} = \tilde{\mathbf{P}}^{-1}(\mathbf{x} - \mathbf{p}_{d+1})$ transforms the simplex s into $s' = \text{conv}(\{\mathbf{0}, \mathbf{e}_1, \ldots, \mathbf{e}_d\})$, where $\mathbf{e}_i$ is the i^{th} vector of the standard basis for $\mathbb{R}^d$. $V(s)$ is given by

$$V(s) = \int_s 1 \, d\mathbf{x} = \int_{s'} \det \tilde{\mathbf{P}} \, d\bar{\mathbf{x}}$$

From calculus,

$$\int_{s'} 1 \, d\bar{\mathbf{x}} = \int_0^1 \int_0^{1-\bar{x}_1} \int_0^{1-\bar{x}_1-\bar{x}_2} \cdots \int_0^{1-\bar{x}_1-\cdots\bar{x}_{d-1}} 1 \; d\bar{x}_d \cdots d\bar{x}_1 = \frac{1}{d!},$$

thus $V(s) = \frac{1}{d!} \det \tilde{\mathbf{P}}$. Let

$$\mathbf{A} = \begin{bmatrix} \mathbf{I}_{d\times d} & \mathbf{0}_{d\times 1} \\ -1 \cdots -1 & 1 \end{bmatrix},$$

but $\det \mathbf{A} = 1$, so $\det \mathbf{P} = \det \mathbf{PA}$. Examining the structure of $\mathbf{PA}$, we see

$$\mathbf{PA} = \begin{bmatrix} \tilde{\mathbf{P}} & \mathbf{p}_{d+1} \\ \mathbf{0}_{1\times d} & 1 \end{bmatrix}.$$

Expanding the determinant $\mathbf{PA}$ about the bottom row gives $\det \mathbf{PA} = \det \tilde{\mathbf{P}}$. Thus, $V(s) = \frac{1}{d!} \det \tilde{\mathbf{P}} = \frac{1}{d!} \det \mathbf{P}$. □

A.2 Properties of the Dilation

Claim (Restatement of 2.11). *Let* $\bar{s} \subseteq \mathbb{R}^d$ *be a d-simplex with vertices* $\mathbf{p}_1, \ldots, \mathbf{p}_{d+1}$*. Let*

$$\epsilon_{\min} = -\left(\sum_{i=1}^{d+1} 1/\delta(\mathbf{p}_i, H_i) \right)^{-1}, \tag{A.8}$$

where H_i is the opposing hyperplane to $\mathbf{p}_i$. $\mathcal{D}(\bar{s}, \epsilon_{\min})$ is a single point. For $\epsilon > \epsilon_{\min}$, $\mathcal{D}(\bar{s}, \epsilon)$ is a d-simplex, with faces parallel to and translated distance $|\epsilon|$ away from the faces of $\bar{s}$. For $\epsilon_{\min} \leq \epsilon \leq 0$, $\mathcal{D}(\bar{s}, \epsilon) \subseteq \bar{s}$, while for $\epsilon \geq 0$, $\bar{s} \subseteq \mathcal{D}(\bar{s}, \epsilon)$.

Proof. Recall that the definition of the dilation is

$$\mathcal{D}(\bar{s}, \epsilon) := \left\{ x = \sum_{j=1}^{d+1} \alpha_j \mathbf{p}_j \middle| \sum_{j=1}^{d+1} \alpha_j = 1 \text{ with } \alpha_j \geq \frac{-\epsilon}{\delta(\mathbf{p}_j, H_j)} \right\},$$

First we will show that $\mathcal{D}(\bar{s}, \epsilon_{\min})$ is a single point. Note that

$$\sum_{i=1}^{d+1} \frac{-\epsilon_{\min}}{\delta(\mathbf{p}_i, H_i)} = 1,$$

and any point in in $\mathcal{D}(\bar{s}, \epsilon_{\min})$ has $\alpha_i \geq -\epsilon_{\min}/\delta(\mathbf{p}_i, H_i)$ for all i. It follows that $\mathcal{D}(\bar{s}, \epsilon_{\min})$ is a single point.

For $\epsilon > \epsilon_{\min}$, $\mathcal{D}(\bar{s}, \epsilon_{\min})$ can be viewed in light of Claim 2.6 as the intersection of $d+1$ closed halfspaces, where the boundary hyperplane of each halfspaces is parallel to a respective H_i. Moreover the intersection is nonempty, since each of the halfspaces contains $\mathcal{D}(\bar{s}, \epsilon_{\min})$, and bounded, since the α_i's are bounded. By Theorem 3.2.5 of [Web94], a polyhedral set (intersection of closed halfspaces) is a polytope (convex hull of a finite number of points) if and only if the set is bounded. Thus $\mathcal{D}(\bar{s}, \epsilon)$ is a polytope. Since $\mathcal{D}(\bar{s}, \epsilon)$ is the intersection of $d+1$ closed halfspaces, it can have at most $\binom{d+1}{d} = d+1$ extreme points. For $\epsilon > \epsilon_{\min}$, inspection of the α_i's shows that $\mathcal{D}(\bar{s}, \epsilon)$ must have a nonempty interior, in which case $\mathcal{D}(\bar{s}, \epsilon)$ must have at least $d+1$ vertices. Thus $\mathcal{D}(\bar{s}, \epsilon)$ is a polytope with exactly $d+1$ vertices, i.e. $\mathcal{D}(\bar{s}, \epsilon)$ is a d-simplex.

That $\mathcal{D}(\bar{s}, \epsilon) \subseteq \bar{s}$ for $\epsilon_{\min} \leq \epsilon \leq 0$ and $\bar{s} \subseteq \mathcal{D}(\bar{s}, \epsilon)$ for $\epsilon \geq 0$ follows from the definition of the dilation and Claim 2.6. □

Claim (Restatement of 2.12). *Let $\bar{s}_a, \bar{s}_b \subseteq \mathbb{R}^d$ be incident d-simplices, that is $\bar{s}_a \cap \bar{s}_b = s_{ab}$, where $s_{ab} \preceq \bar{s}_a, \bar{s}_b$ is a $(k-1)$-simplex, $1 \leq k < d+1$. Let $\text{vert}(\bar{s}_a) = \{\mathbf{p}_1, \ldots, \mathbf{p}_{d+1}\}$, $\text{vert}(\bar{s}_b) = \{\mathbf{p}_1, \ldots, \mathbf{p}_k, \mathbf{q}_{k+1}, \ldots, \mathbf{q}_{d+1}\}$, and $\text{vert}(s_{ab}) = \{\mathbf{p}_1, \ldots, \mathbf{p}_k\}$. Let $A = \text{aff}\, s_{ab}$. Then $\exists \kappa_{a,b} > 0$ such that $\forall \epsilon > 0$, if $\mathbf{x} \in \mathcal{D}(\bar{s}_a, \epsilon) \cap \bar{s}_b$ then $\delta(\mathbf{x}, A) < \kappa_{a,b}\epsilon$.*

Proof. Let

$$\mathbf{V}_1 = \begin{bmatrix} \mathbf{p}_1 & \cdots & \mathbf{p}_k \\ 1 & \ldots & 1 \end{bmatrix}, \quad \mathbf{V}_3 = \begin{bmatrix} \mathbf{p}_{k+1} & \cdots & \mathbf{p}_{d+1} \\ 1 & \ldots & 1 \end{bmatrix}, \quad \mathbf{V}_2 = \begin{bmatrix} \mathbf{q}_{k+1} & \cdots & \mathbf{q}_{d+1} \\ 1 & \ldots & 1 \end{bmatrix},$$

$$\mathbf{V}_a = \begin{bmatrix} V_1 & V_3 \end{bmatrix}, \qquad \mathbf{V}_b = \begin{bmatrix} V_1 & V_2 \end{bmatrix}.$$

The barycentric coordinates of $x \in \mathbb{R}^d$ with respect to $\bar{s}_a$ and $\bar{s}_b$ are given respectively by

$$\begin{bmatrix} \alpha_1 & \cdots & \alpha_{d+1} \end{bmatrix}^{\mathrm{T}} = {\mathbf{V}_a}^{-1} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}, \qquad \begin{bmatrix} \beta_1 & \cdots & \beta_{d+1} \end{bmatrix}^{\mathrm{T}} = {\mathbf{V}_b}^{-1} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}.$$

For $\mathbf{x} \in \mathcal{D}(\bar{s}_a, \epsilon)$, the barycentric coordinates of $\mathbf{x}$ with respect to $\bar{s}_a$ must satisfy

$$\begin{bmatrix} \alpha_1 & \cdots & \alpha_{d+1} \end{bmatrix}^{\mathrm{T}} \geq \begin{bmatrix} -\epsilon/\delta(\mathbf{p}_1, H_1) & \cdots & -\epsilon/\delta(\mathbf{p}_{d+1}, H_{d+1}) \end{bmatrix}^{\mathrm{T}} \tag{A.9}$$

by the definition of the dilation of a simplex. For $\mathbf{x} \in \bar{s}_b$, the barycentric coordinates of $\mathbf{x}$ with respect to $\bar{s}_b$ must satisfy

$$\begin{bmatrix} \beta_1 & \cdots & \beta_{d+1} \end{bmatrix}^{\mathrm{T}} \geq \begin{bmatrix} 0 & \cdots & 0 \end{bmatrix}^{\mathrm{T}} \tag{A.10}$$

We are interested in $\mathbf{x} \in \mathcal{D}(\bar{s}_a, \epsilon) \cap \bar{s}_b$, so both Equation A.9 and Equation A.10 must hold. The barycentric coordinates of $\mathbf{x}$ with respect to $\bar{s}_a$ and $\bar{s}_b$ are related by

$$\begin{bmatrix} \alpha_1 & \cdots & \alpha_{d+1} \end{bmatrix}^{\mathrm{T}} = {\mathbf{V}_a}^{-1}\mathbf{V}_b \begin{bmatrix} \beta_1 & \cdots & \beta_{d+1} \end{bmatrix}^{\mathrm{T}}. \tag{A.11}$$

Since $\mathbf{V}_a$ and $\mathbf{V}_b$ have the first k columns in common, ${\mathbf{V}_a}^{-1}\mathbf{V}_b$ has the structure

$${\mathbf{V}_a}^{-1}\mathbf{V}_b = \begin{bmatrix} \mathbf{I}_k & \mathbf{M}_1 \\ \mathbf{0} & \mathbf{M}_2 \end{bmatrix}$$

where $\mathbf{M}_1 \in \mathbb{R}^{k \times (d+1-k)}$ and $\mathbf{M}_2 \in \mathbb{R}^{(d+1-k)\times(d+1-k)}$.

The vertices of $\bar{s}_b$ are partitioned into two sets, $\sigma_1 = \{\mathbf{p}_1, \ldots, \mathbf{p}_k\}$ and $\sigma_2 = \{\mathbf{q}_{k+1}, \ldots, \mathbf{q}_{d+1}\}$. Any point $\mathbf{x} \in \bar{s}_b$ can be described as a triplet $(t, \boldsymbol{\beta}^1, \boldsymbol{\beta}^2)$, where $t \in [0, 1]$, $\boldsymbol{\beta}^1 \in \boldsymbol{\Delta}_{k-1}$, and $\boldsymbol{\beta}^2 \in \boldsymbol{\Delta}_{d-k}$. $\boldsymbol{\beta}^1$ and $\boldsymbol{\beta}^2$ are barycentric coordinates of two points, $\mathbf{x}_1 \in \mathrm{conv}(\sigma_1)$ and $\mathbf{x}_2 \in \mathrm{conv}(\sigma_2)$, respectively. The parameter t gives the position of $\mathbf{x}$ along the line through $\mathbf{x}_1$ and $\mathbf{x}_2$, that is $\mathbf{x} = (1-t)\mathbf{x}_1 + t\mathbf{x}_2$. This representation is unique for $\mathbf{x} \in \mathbb{R}^d \backslash (\mathrm{aff}\sigma_1 \cup \mathrm{aff}\sigma_2)$. The relationship between $\mathbf{x}$ and $(t, \boldsymbol{\beta}^1, \boldsymbol{\beta}^2)$ is given by

$$\begin{bmatrix} \mathbf{x}^{\mathrm{T}} & 1 \end{bmatrix}^{\mathrm{T}} = (1-t)\mathbf{V}_1\boldsymbol{\beta}^1 + t\mathbf{V}_3\boldsymbol{\beta}^2.$$

The relationship between the barycentric coordinates of $\mathbf{x}$ with respect to $\bar{s}_b$ and the triplet $(t, \boldsymbol{\beta}^1, \boldsymbol{\beta}^2)$ is

$$\begin{bmatrix} \beta_1 \\ \vdots \\ \beta_k \\ \beta_{k+1} \\ \vdots \\ \beta_{d+1} \end{bmatrix} = (1-t) \begin{bmatrix} \beta_1^1 \\ \vdots \\ \beta_k^1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + t \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \beta_1^2 \\ \vdots \\ \beta_{d+1-k}^2 \end{bmatrix}$$

and $(t, \boldsymbol{\beta}^1, \boldsymbol{\beta}^2)$ may be computed from $\boldsymbol{\beta}$ by

$$\begin{aligned} t &= \textstyle\sum_{i=k+1}^{d+1} \beta_i \\ \beta_i^1 &= \beta_i/(1-t) \qquad i = 1, \ldots, k \\ \beta_i^2 &= \beta_{i+k}/t \qquad i = 1, \ldots, d+1-k \end{aligned}$$

Rewriting the right-hand side of (A.11) using this representation gives

$$\begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_{d+1} \end{bmatrix} = (1-t) \begin{bmatrix} \mathbf{I}_k \\ \mathbf{0} \end{bmatrix} \begin{bmatrix} \beta_1^1 \\ \vdots \\ \beta_k^1 \end{bmatrix} + t \begin{bmatrix} \mathbf{M}_1 \\ \mathbf{M}_2 \end{bmatrix} \begin{bmatrix} \beta_1^2 & \cdots & \beta_{d+1-k}^2 \end{bmatrix}^{\mathrm{T}}.$$

The desired result will be achieved by bounding $\sum_{i=k+1}^{d+1} \beta_i$ using the following two subclaims and then applying Claim 2.7. The proofs of the subclaims follow the present proof.

Subclaim 1. *Let ζ be given by*

$$\zeta = -\max_{\boldsymbol{\beta}^2 \in \boldsymbol{\Delta}_{d-k}} f(\boldsymbol{\beta}^2) \tag{A.12}$$

$$f(\boldsymbol{\beta}^2) = \min_{1 \le i \le d+1-k} \left[\mathbf{M}_2 \boldsymbol{\beta}^2\right]_i \delta(\mathbf{p}_{i+k}, H_{i+k}), \tag{A.13}$$

where $[\cdot]_i$ is the i-th vector component. Then ζ is well defined and $\zeta > 0$.

Note that $t\mathbf{M}_2\boldsymbol{\beta}^2 = \begin{bmatrix} \alpha_{k+1} \cdots \alpha_{d+1} \end{bmatrix}^{\mathrm{T}}$ and recall from Claim 2.6 that $\delta_s(\mathbf{x}, H_i) = \alpha_i \delta(\mathbf{p}_i, H_i)$. Consider a point $\mathbf{x}$ with representation $(t, \boldsymbol{\beta}^1, \boldsymbol{\beta}^2)$. Then ζ in Subclaim 1 can be interpreted as a bound (which scales with t) on how far $\mathbf{x}$ is on the negative side of hyperplanes $H_{k+1}, \ldots, H_{d+1}$. This is made formal with the following subclaim.

Subclaim 2. *Let $\mathbf{x} \in \bar{s}_b$ have representation $(\boldsymbol{\beta}^1, \boldsymbol{\beta}^2, t)$. If $t > \frac{1}{\zeta}\epsilon$, then there exists i such that $\alpha_i < -\epsilon/\delta(\mathbf{p}_i, H_i)$.*

Let $\epsilon > 0$ be given. Let $\mathbf{x} \in \mathcal{D}(\bar{s}_a, \epsilon) \cap \bar{s}_b$. It follows that the barycentric representation of $\mathbf{x}$ with respect to $\bar{s}_a$ satisfies $\alpha_i \ge -\epsilon/\delta(\mathbf{p}_i, H_i)$ for all i. By the contrapositive of Subclaim 2, the representation of $\mathbf{x}$ as $(t, \boldsymbol{\beta}^1, \boldsymbol{\beta}^2)$ must have $t \le \frac{1}{\zeta}\epsilon$. It follows that the barycentric coordinates of $\mathbf{x}$ with respect to $\bar{s}_b$ satisfy

$$\sum_{i=k+1}^{d+1} \beta_i = t \le \frac{1}{\zeta}\epsilon. \tag{A.14}$$

Let $\bar{\boldsymbol{\beta}}_s{}^{\mathrm{T}} = \begin{bmatrix} \beta_{k+1} & \cdots & \beta_{d+1} \end{bmatrix}^{\mathrm{T}}$. By Claim 2.7 and the properties of symmetric positive definite matrices,

$$\delta(\mathbf{x}, A) = \sqrt{\bar{\boldsymbol{\beta}}_s \mathbf{G} \bar{\boldsymbol{\beta}}_s} \le \sqrt{\lambda_{\max}(\mathbf{G})} \left\|\bar{\boldsymbol{\beta}}_s\right\|_2,$$

where $\mathbf{G}$ is the positive definite matrix given by Equation A.6. The infinity norm dominates the 2 norm, and $\mathbf{x} \in \bar{s}_b$, $\beta_i \ge 0$, so

$$\delta(\mathbf{x}, A) \le \sqrt{\lambda_{\max}(\mathbf{G})} \sum_{i=k+1}^{d+1} \beta_i$$

Finally, applying Equation A.14 gives

$$\delta(\mathbf{x}, A) \le \frac{\sqrt{\lambda_{\max}(\mathbf{G})}}{\zeta}\epsilon,$$

which is the desired result, with

$$\kappa_{a,b} = \frac{\sqrt{\lambda_{\max}(\mathbf{G})}}{\zeta}. \tag{A.15}$$

□

Proof of Subclaim 1. First we show that $\mathbf{M}_2\boldsymbol{\beta}^2$ has a strictly negative component $\forall \boldsymbol{\beta}^2 \in \boldsymbol{\Delta}_{d-k}$. Let $\boldsymbol{\beta}^2$ be given. Let $\boldsymbol{\beta}^1 \in \boldsymbol{\Delta}_{k-1}$ be

$$\boldsymbol{\beta}^1 = \begin{bmatrix} 1/k & \cdots & 1/k \end{bmatrix}^{\mathrm{T}}$$

For $t = 0$, the point with representation $(t, \boldsymbol{\beta}^1, \boldsymbol{\beta}^2)$ has barycentric coordinates with respect to $\bar{s}_a$ given by

$$\begin{bmatrix} \alpha_1 & \cdots & \alpha_k & \alpha_{k+1} & \cdots & \alpha_{d+1} \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} 1/k & \cdots & 1/k & 0 & \cdots & 0 \end{bmatrix}^{\mathrm{T}}$$

For any point with $0 < t \leq 1$, the point with representation $(t, \boldsymbol{\beta}^1, \boldsymbol{\beta}^2)$ lies strictly outside of $\bar{s}_a$. Thus, in the barycentric coordinates of this point with respect to $\bar{s}_a$, there exists i such that $\alpha_i < 0$. Consider a point given by $(t, \boldsymbol{\beta}^1, \boldsymbol{\beta}^2)$ with $t > 0$, but very small, then

$$\begin{bmatrix} \alpha_1 \cdots \alpha_k & \alpha_{k+1} \cdots \alpha_{d+1} \end{bmatrix}^{\mathrm{T}} = (1-t) \begin{bmatrix} \frac{1}{k} \cdots \frac{1}{k} & 0 \cdots 0 \end{bmatrix}^{\mathrm{T}} + t \begin{bmatrix} \left(\mathbf{M}_1\boldsymbol{\beta}^2\right)^{\mathrm{T}} & \left(\mathbf{M}_2\boldsymbol{\beta}^2\right)^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}.$$

For t very small, the first k components of α must be positive by continuity. Since some component α_i must be strictly negative when $t > 0$, it must be that a component of $t\mathbf{M}_2\boldsymbol{\beta}^2$ is strictly negative. Since t is positive, it follows that a component of $\mathbf{M}_2\boldsymbol{\beta}^2$ is strictly negative.

Since $\delta(\mathbf{p}_{i+k}, H_{i+k}) > 0$ for $1 \leq i \leq d+1-k$, and at least one component of $\mathbf{M}_2\boldsymbol{\beta}^2$ is strictly negative, it follows that $f(\boldsymbol{\beta}^2) < 0$ for all $\boldsymbol{\beta}^2$. Since sum, multiplication, and minimum are continuous, f is a continuous function. Since $\boldsymbol{\Delta}_{d-k}$ is compact, f must take its maximum for some $\boldsymbol{\beta}^2$, thus ζ is well defined. As we argued above, $f(\boldsymbol{\beta}^2) < 0$ for all $\boldsymbol{\beta}^2$, thus

$$\max_{\boldsymbol{\beta}^2 \in \boldsymbol{\Delta}_{d-k}} f(\boldsymbol{\beta}^2) < 0.$$

Thus $\zeta > 0$. □

Proof of Subclaim 2. By the definition of ζ there exists i such that $[\mathbf{M}_2\boldsymbol{\beta}^2]_i \, \delta(\mathbf{p}_{i+k}, H_{i+k}) \leq -\zeta$ and by hypothesis $t > \frac{1}{\zeta}\epsilon$. Thus

$$\begin{aligned} \alpha_{k+i} &= t \, [\mathbf{M}_2\boldsymbol{\beta}^2]_i \\ &< -\frac{1}{-\zeta}\epsilon \, \frac{-\zeta}{\delta(\mathbf{p}_{i+k}, H_{i+k})} \\ &= -\frac{\epsilon}{\delta(\mathbf{p}_{i+k}, H_{i+k})}, \end{aligned}$$

Thus, there exist i such that $\alpha_i < \epsilon / \delta(\mathbf{p}_i, H_i)$. □

A.3 Triangulations

A.3.1 Comparison of Simplicial- and Δ- Complexes

The definition of simplicial complex presented in 2.2.1 is geometrically motivated, adapted from [Ede01]. It is slightly more constrained (and significantly less abstract) than the definition of simplicial complex found in the traditional algebraic topology literature such as [Spa66], which in turn is more constrained than the notion of a $\mathbf{\Delta}$-complex found in modern algebraic topology [Hat01]. In order to distinguish between the definitions of simplicial complex in the following discussion, we will use "geometric" to refer to the definitions presented in section 2.2.1 and "combinatorial" for the algebraic topology definition.

The definition of simplicial complex from traditional algebraic topology, such as [Spa66, page 108] is as follows. A simplicial complex $\mathcal{K}$ consists of a set $\mathcal{V}$ of vertices and a set $\mathcal{S}$ of finite nonempty subsets of $\mathcal{V}$ called simplices such that

i) any set consisting of exactly one vertex is a simplex, and

ii) any nonempty subset of a simplex is a simplex.

The dimension of the $\mathcal{K}$ is $\dim \mathcal{K} := \sup_{s\in\mathcal{S}} \operatorname{card}(s) - 1$. The combinatorial simplicial complex is equivalent to the abstract simplicial complex presented in section 2.2.1. The underlying space $|\mathcal{K}|$ of a combinatorial simplicial complex is constructed and depends only on the combinatorial structure of vertices of $\mathcal{K}$. Let $|\mathcal{K}|$ be the set of all functions $\alpha : \mathcal{V} \to I$, where I is the closed interval $[0, 1]$, such that

i) For any α, the set $\{v \in \mathcal{V} | \alpha(v) \neq 0\}$ is a simplex of $\mathcal{K}$, and

ii) For any α, $\sum_{\alpha\in\mathcal{V}} \alpha(v) = 1$.

The *closed simplex* $|s| \subseteq |\mathcal{K}|$ is similarly defined by $|s| = \{\alpha \in |\mathcal{K}| \mid \alpha(v) \neq 0 \Rightarrow v \in s\}$. A metric can be defined on $|\mathcal{K}|$ by $d(\alpha, \beta) = \sqrt{\sum_{v\in\mathcal{V}}[\alpha(v) - \beta(v)]^2}$. Applying this metric to $|\mathcal{K}|$, it can be shown that for any $s \in \mathcal{K}$ the closed simplex $|s|$ is homeomorphic to the standard simplex $\mathbf{\Delta}_d$ (defined in Equation 2.7), where $d = \operatorname{card}(s) - 1$. Notice that a closed simplex is uniquely determined by its vertices, the consequences of which will be examined in the example below.

Consider a geometric simplicial complex $\mathcal{L}$ defined as in section 2.2.1. Then the set $\mathcal{V} = \operatorname{vert}(\mathcal{L})$ and the collection of sets $\mathcal{S} = \{\{\operatorname{vert}(s)\} \,|\, s \in \mathcal{L}\}$ define a combinatorial simplicial complex $\mathcal{K}$. Moreover, using the metric defined above, $|\mathcal{L}|$ is homeomorphic to $|\mathcal{K}|$, and each simplex in $\mathcal{L}$ is homeomorphic to the corresponding closed simplex of $\mathcal{K}$. So any geometric simplicial complex has a corresponding combinatorial simplicial complex. By Theorem 3.2.9 of [Spa66, page 120], if a combinatorial simplicial complex $\mathcal{K}$ is countable (i.e. the collection of simplices $\mathcal{S}$ is countable) and locally finite (each vertex $v \in \mathcal{V}$ is in a finite number of simplices in $\mathcal{S}$) and $\dim \mathcal{K} \leq n$, then $\mathcal{K}$ has a realization as a closed subset in $\mathbb{R}^{2n+1}$. That is, there exists a geometric simplicial complex equivalent to $|\mathcal{K}|$. Since this dissertation is concerned with computable piecewise linear functions, it is reasonable to restrict attention to the geometric definition of simplicial complex at the loss of representing complexes that

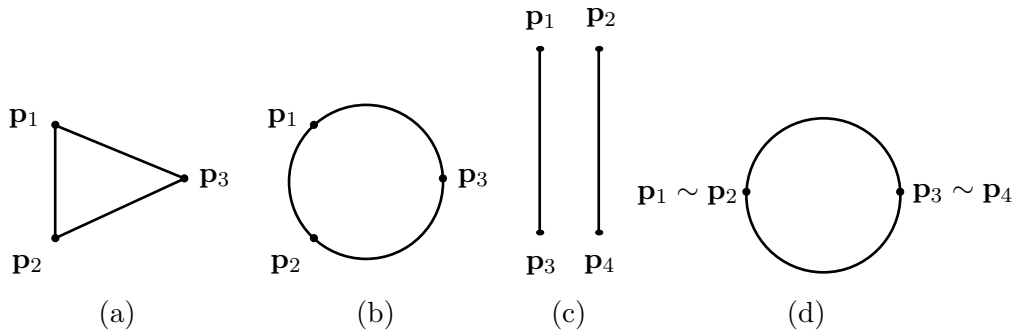

Figure A.1: Embeddings of an abstract simplicial complex and $\mathbf{\Delta}$-complex. A linear embedding (a) and a general embedding (b) of $|\mathcal{K}|$ for the simplicial complex given in Equation A.16. The disjoint simplices (c) and an embedding of the $\mathbf{\Delta}$-complex (d) from Equation A.17.

would not be computable.

Consider the combinatorial simplicial complex $\mathcal{K}$ defined by

$$\mathcal{V} = \{p_1, p_2, p_3\}, \quad \mathcal{S} = \{\{p_1, p_2\}, \{p_2, p_3\}, \{p_1, p_3\}, \{p_1\}, \{p_2\}, \{p_3\}\}. \tag{A.16}$$

Then $|\mathcal{K}|$ is homeomorphic to S^1. Figure A.1 (a) and (b) show a linear embedding of $|\mathcal{K}|$ and the image of a homeomorphism of $|\mathcal{K}|$ to S^1. Further contemplating the example, it is clear that a simplicial complex equivalent to S^1 cannot be constructed with less than 3 vertices (Because the underlying space of a simplicial complex with two vertices contains at most one closed 1-simplex). Although it might seem natural to represent the circle with two vertices and two edges as in Figure A.1 (d), this is not possible with a simplicial complex, however it is possible with a $\mathbf{\Delta}$-complex, the modern generalization of simplicial complex.

A $\mathbf{\Delta}$-complex is a quotient space of a collection of disjoint simplices obtained by identifying certain sets of faces. Somewhat more formally [Hat01, page 103], a $\mathbf{\Delta}$-complex is based on a collection of disjoint (geometric) simplices s_α of various dimensions along with sets $\mathcal{F}_i$ of faces of the s_α, all faces in a given $\mathcal{F}_i$ having the same dimension. The $\mathbf{\Delta}$-complex is the quotient space formed by taking the disjoint union of the simplices, $\coprod_\alpha s_\alpha$ and identifying the faces of each $\mathcal{F}_i$ through a transformation to the canonical simplex (i.e. the ordering of the vertices in the face affect the way the faces are "glued" together through the identification). As an example, consider a $\mathbf{\Delta}$-complex consisting of two edges and two sets of faces identified,

$$\mathcal{S} = \{[p_1, p_3], [p_2, p_4]\}, \quad \mathcal{F}_1 = \{p_1, p_2\}, \ \mathcal{F}_2 = \{p_3, p_4\}, \tag{A.17}$$

illustrated in Figure A.1 (c). This $\mathbf{\Delta}$-complex is homeomorphic to S^1, but contains only two vertices and two 1-simplices. Figure A.1 (d) shows the image of a homeomorphism of the $\mathbf{\Delta}$-complex to S^1.

Since the present work involves only PL functions in Euclidean space, the distinction between simplicial complexes and $\mathbf{\Delta}$-complexes does not arise. Several potential applications involve PL functions on non-Euclidean space, such as gait-generation [WBGK03], which requires PLHs from T^k (the k-torus) into itself. In this context it may well prove useful to switch from simplicial complexes to $\mathbf{\Delta}$-complexes.

A.3.2 Parameterization of Triangulations

Claim (Restatement of 2.13). *$\mathcal{K}(P, \mathcal{S})$ is a geometric simplicial complex if*

1. *$\forall \alpha \in \mathcal{S}, P(\alpha)$ are affinely independent*
2. *$s_1, s_2 \in \mathcal{K}(P, \mathcal{S}) \implies s_1 \cap s_2 \preceq s_1, s_2$.*

Moreover, if these properties hold, then $\text{vert}(\mathcal{K}(P, \mathcal{S})) = P$.

Proof. Let $s_1 \in \mathcal{K}(P, \mathcal{S})$ and let $s_2 \preceq s_1$. Since $s_1 \in \mathcal{K}(P, \mathcal{S})$, there exists α_1 such that $s_1 = \text{conv}(P(\alpha_1))$. By Property 1, $P(\alpha_1)$ is affinely independent, thus $\text{vert}(s_1) = P(\alpha_1)$. Since $s_2 \preceq s_1$, $\text{vert}(s_2) \subseteq \text{vert}(s_1)$. Since $\mathcal{S}$ is an abstract complex, $\exists \alpha_2 \in \mathcal{S}$ such that $\alpha_2 \subseteq \alpha_1$ and $P(\alpha_2) = \text{vert}(s_2)$. Thus $s_2 \in \mathcal{K}(P, \mathcal{S})$. Thus, $s_1 \in \mathcal{K}$ and $s_2 \preceq s_1 \implies s_2 \in \mathcal{K}$. Since $\mathcal{K}(P, \mathcal{S})$ is a collection of simplices satisfying this and Property 2, it follows that $\mathcal{K}(P, \mathcal{S})$ is a geometric simplicial complex.

Since the vertex set of $\mathcal{S}$ is $\{1, \ldots, n\}$, it follows from the definition of $\mathcal{K}(\cdot, \cdot)$ that $\{\mathbf{p}_i\} \in \mathcal{K}(P, \mathcal{S})$ for $i = 1, \ldots, n$, and moreover these are the only singleton sets in $\mathcal{K}(P, \mathcal{S})$. If follows that $\text{vert}(\mathcal{K}(P, \mathcal{S})) = P$. □

A.3.3 Delaunay Triangulation

Claim A.1. *Let $\mathbf{p}_1, \ldots, \mathbf{p}_{d+1} \in \mathbb{R}^d$ be affinely independent. Then there is a unique sphere through these points, with center $\mathbf{c} = \frac{1}{2}\mathbf{A}^{-1}\mathbf{b}$ and radius $r = \|\mathbf{p}_i - \mathbf{c}\|$, where*

$$\mathbf{A} = \begin{bmatrix} (\mathbf{p}_1 - \mathbf{p}_{d+1})^{\mathrm{T}} \\ (\mathbf{p}_2 - \mathbf{p}_{d+1})^{\mathrm{T}} \\ \vdots \\ (\mathbf{p}_d - \mathbf{p}_{d+1})^{\mathrm{T}} \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} \|\mathbf{p}_1\|^2 - \|\mathbf{p}_{d+1}\|^2 \\ \|\mathbf{p}_2\|^2 - \|\mathbf{p}_{d+1}\|^2 \\ \vdots \\ \|\mathbf{p}_d\|^2 - \|\mathbf{p}_{d+1}\|^2 \end{bmatrix}. \tag{A.18}$$

Proof. If there is a unique point which is equidistant from the points $\mathbf{p}_1, \ldots, \mathbf{p}_{d+1}$, then there is a unique sphere through these points. The hyperplane defined by

$$H = \left\{ \mathbf{x} \in \mathbb{R}^d \mid (\mathbf{p}_i - \mathbf{p}_{d+1})^{\mathrm{T}}(\mathbf{x} - \mathbf{p}_{d+1}) = \frac{1}{2}\|\mathbf{p}_i - \mathbf{p}_{d+1}\|^2 \right\}$$

is the set of all points equidistant from $\mathbf{p}_i$ and $\mathbf{p}_{d+1}$. Algebraic manipulation shows that $\mathbf{x} \in H$ if and only if

$$(\mathbf{p}_i - \mathbf{p}_{d+1})^{\mathrm{T}}\mathbf{x} = \frac{1}{2}\left(\|\mathbf{p}_i\|^2 - \|\mathbf{p}_{d+1}\|^2\right)$$

Expressing this relationship in matrix form for $i = 1, \ldots, d$ gives $\mathbf{A}\mathbf{x} = \frac{1}{2}\mathbf{b}$, with $\mathbf{A}$ and $\mathbf{b}$ defined as above. Since the points are affinely independent, the rows of $\mathbf{A}$ are linearly independent by Claim 2.1, so $\mathbf{A}$ is invertible. Thus there is a unique point, $\mathbf{c} = \frac{1}{2}\mathbf{A}^{-1}\mathbf{b}$, equidistant from $\mathbf{p}_1, \ldots, \mathbf{p}_{d+1}$. A hypersphere with center $\mathbf{c}$ and radius $\|\mathbf{p}_i - \mathbf{c}\|$ contains $\mathbf{p}_1, \ldots, \mathbf{p}_{d+1}$. □

Claim A.2. *Let $\mathbf{p}_1, \ldots, \mathbf{p}_{d+1} \in \mathbb{R}^d$ be affinely independent, and let $\tilde{\mathbf{p}}_1, \ldots, \tilde{\mathbf{p}}_{d+1} \in \mathbb{R}^{d+1}$ be the projection to the paraboloid, $\tilde{\mathbf{p}}_i = \begin{bmatrix} \mathbf{p}_i{}^{\mathrm{T}} & \|\mathbf{p}_i\|^2 \end{bmatrix}^{\mathrm{T}}$. Then $\mathrm{conv}(\{\tilde{\mathbf{p}}_1, \ldots, \tilde{\mathbf{p}}_{d+1}\})$ has a downward pointing normal vector given by*

$$\mathbf{n} = \begin{bmatrix} \mathbf{A}^{-1}\mathbf{b} \\ -1 \end{bmatrix} \tag{A.19}$$

where $\mathbf{A}$ and $\mathbf{b}$ are given by Equation A.18.

Proof. $\mathbf{n}$ is normal to $\mathrm{conv}(\{\tilde{\mathbf{p}}_1, \ldots, \tilde{\mathbf{p}}_{d+1}\})$ if and only if it is orthogonal to $\tilde{\mathbf{p}}_1 - \tilde{\mathbf{p}}_{d+1}, \ldots, \tilde{\mathbf{p}}_d - \tilde{\mathbf{p}}_{d+1}$. Moreover, the last component of $\mathbf{n}$ should be -1. These conditions can be expressed in matrix form as

$$\begin{bmatrix} (\tilde{\mathbf{p}}_1 - \tilde{\mathbf{p}}_{d+1})^{\mathrm{T}} \\ (\tilde{\mathbf{p}}_2 - \tilde{\mathbf{p}}_{d+1})^{\mathrm{T}} \\ \vdots \\ (\tilde{\mathbf{p}}_d - \tilde{\mathbf{p}}_{d+1})^{\mathrm{T}} \\ \mathbf{0}_{1\times d} \quad 1 \end{bmatrix} \mathbf{n} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ -1 \end{bmatrix} \iff \begin{bmatrix} (\mathbf{p}_1 - \mathbf{p}_{d+1})^{\mathrm{T}} & \|\mathbf{p}_1\|^2 - \|\mathbf{p}_{d+1}\|^2 \\ (\mathbf{p}_2 - \mathbf{p}_{d+1})^{\mathrm{T}} & \|\mathbf{p}_2\|^2 - \|\mathbf{p}_{d+1}\|^2 \\ \vdots & \\ (\mathbf{p}_d - \mathbf{p}_{d+1})^{\mathrm{T}} & \|\mathbf{p}_d\|^2 - \|\mathbf{p}_{d+1}\|^2 \\ \mathbf{0}_{1\times d} & 1 \end{bmatrix} \mathbf{n} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ -1 \end{bmatrix}$$

$$\iff \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0}_{1\times d} & 1 \end{bmatrix} \mathbf{n} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ -1 \end{bmatrix}.$$

But

$$\begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0}_{1\times d} & 1 \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{b} \\ \mathbf{0}_{1\times d} & 1 \end{bmatrix}$$

from which the result follows directly. □

Claim A.3. *The algorithm from Section 2.3.2 for computing the Delaunay triangulation by computing the convex hull of the projection to the paraboloid is correct.*

Proof. Let $\mathcal{T}(P, \mathcal{S})$ be the triangulation given by the algorithm. Assume it is not Delaunay. Then there exists a d-simplex whose circumsphere contains a point of P that is not one of its vertices. Without loss of generality assume that the vertices of the d-simplex are $\mathbf{p}_1, \ldots, \mathbf{p}_{d+1}$ and that $\mathbf{p}_i$, $i > d+1$, is the point that lies within the circumsphere of the d-simplex. Let $\mathbf{c} = \frac{1}{2}\mathbf{A}^{-1}\mathbf{b}$ be the center of the circumsphere, with $\mathbf{A}$ and $\mathbf{b}$ as defined in

Equation A.18. Let $\mathbf{n}$ be the downward pointing normal to $\text{conv}(\{\tilde{\mathbf{p}}_1, \tilde{\mathbf{p}}_2, \ldots, \tilde{\mathbf{p}}_{d+1}\})$ that has the last coordinate of -1. By Claim A.2 above,

$$\mathbf{n} = \begin{bmatrix} \mathbf{A}^{-1}\mathbf{b} \\ -1 \end{bmatrix} = \begin{bmatrix} 2\mathbf{c} \\ -1 \end{bmatrix}$$

where $\mathbf{A}$ and $\mathbf{b}$ are the same as for the center calculation, and $\mathbf{c}$ is the center of the circumsphere. Since this d-simplex was put in the triangulation by the algorithm, $\mathbf{n}$ must also be an outward pointing normal for the facet $\text{conv}(\{\tilde{\mathbf{p}}_1, \tilde{\mathbf{p}}_2, \ldots, \tilde{\mathbf{p}}_{d+1}\})$ of the convex hull of the projected points. Now consider the quantity $\mathbf{n}^{\mathrm{T}}(\tilde{\mathbf{p}}_i - \tilde{\mathbf{p}}_1)$. If this quantity is positive, then the point $\mathbf{p}_i$ must lie outside the convex hull, which would be a contradiction. Algebraic manipulation gives

$$\begin{aligned} \mathbf{n}^{\mathrm{T}}(\tilde{\mathbf{p}}_i - \tilde{\mathbf{p}}_1) &= 2\mathbf{c}^{\mathrm{T}}(\mathbf{p}_i - \mathbf{p}_1) - (\|\mathbf{p}_i\|^2 - \|\mathbf{p}_1\|^2) \\ &= (\|\mathbf{p}_1\|^2 - 2\mathbf{c}^{\mathrm{T}}\mathbf{p}_1 + \|\mathbf{c}\|^2) - (\|\mathbf{p}_i\|^2 - 2\mathbf{c}^{\mathrm{T}}\mathbf{p}_i + \|\mathbf{c}\|^2) \\ &= \|\mathbf{p}_1 - \mathbf{c}\|^2 - \|\mathbf{p}_i - \mathbf{c}\|^2 \\ &> 0 \end{aligned}$$

which is greater than 0 since $\mathbf{p}_i$ lies within the circumsphere of $\mathbf{p}_1, \ldots, \mathbf{p}_{d+1}$. But this implies that $\tilde{\mathbf{p}}_i$ lies outside the convex hull of the projected points, a contradiction. Thus all simplexes in $\mathcal{T}(P, \mathcal{S})$ satisfy the circumsphere condition. By a similar argument, any $d+1$ points in P that satisfy the circumsphere criterion will be a face of the convex hull. Thus the algorithm constructs the Delaunay triangulation. □

APPENDIX B

MINVAR Calculations

This appendix presents the details of calculations involved in the MINVAR algorithm. Specifically, Section B.1 presents how to perform the "min var" computation in closed form, while Section B.2 shows how to compute the least squares and best L_2 affine approximations by "completing the square."

B.1 Solving the "min var" Equation

The following claim shows how to perform the "min var" calculation of the MINVAR algorithm. For the purpose of this claim, it is assumed that $\lambda > 0$, which guarantees that the matrix $\mathbf{H}_i$ is positive definite. This condition can be relaxed to allow $\lambda = 0$ so long as conditions are imposed on the affine maps to cause $\mathbf{H}_i$ to be positive definite rather than just positive semidefinite.

Claim B.1. *Let L^i be a collection of N_i affine maps, $L_j(\mathbf{x}) = \hat{\mathbf{A}}_j\mathbf{x} + \hat{\mathbf{b}}_j$, $j = 1, \ldots, N_i$. Let $\mathbf{p}_i \in \mathbb{R}^d$ and let $\lambda > 0$. Let*

$$\mathbf{x}_{min} = \arg\min_{\mathbf{x}\in\mathbb{R}^d} \ \operatorname{var} L^i(\mathbf{x}) + \lambda\|\mathbf{x} - \mathbf{p}_i\|^2 \tag{B.1}$$

where

$$\operatorname{var} L^i(\mathbf{x}) = \frac{1}{N_i}\sum_{j=1}^{N_i}\left\| L_j(\mathbf{x}) - \frac{1}{N_i}\sum_{k=1}^{N_i} L_k(\mathbf{x})\right\|^2. \tag{B.2}$$

Then $\mathbf{x}_{min}$ is given by

$$\mathbf{x}_{min} = -{\mathbf{H}_i}^{-1}\mathbf{h}_i \tag{B.3}$$

where

$$\begin{aligned}
\mathbf{H}_i &= \frac{1}{N_i}\sum_{j=1}^{N_i}\left(\hat{\mathbf{A}}_j - \overline{\hat{\mathbf{A}}_i}\right)^{\mathrm{T}}\left(\hat{\mathbf{A}}_j - \overline{\hat{\mathbf{A}}_i}\right) \;+\; \lambda\mathbf{I} \qquad \text{(B.4)}\\
&= \frac{1}{N_i}\sum_{j=1}^{N_i}\hat{\mathbf{A}}_j^{\mathrm{T}}\hat{\mathbf{A}}_j \;-\; \overline{\hat{\mathbf{A}}_i}^{\mathrm{T}}\overline{\hat{\mathbf{A}}_i} \;+\; \lambda\mathbf{I}\\
\mathbf{h}_i &= \frac{1}{N_i}\sum_{j=1}^{N_i}\left(\hat{\mathbf{A}}_j - \overline{\hat{\mathbf{A}}_i}\right)^{\mathrm{T}}\left(\hat{\mathbf{b}}_j - \overline{\hat{\mathbf{b}}_i}\right) \;-\; \lambda\mathbf{p}_i \qquad \text{(B.5)}\\
&= \frac{1}{N_i}\sum_{j=1}^{N_i}\hat{\mathbf{A}}_j{}^{\mathrm{T}}\hat{\mathbf{b}}_j \;-\; \overline{\hat{\mathbf{A}}_i}^{\mathrm{T}}\overline{\hat{\mathbf{b}}_i} \;-\; \lambda\mathbf{p}_i
\end{aligned}$$

and

$$\overline{\hat{\mathbf{A}}_i} = \frac{1}{N_i}\sum_{j=1}^{N_i}\hat{\mathbf{A}}_j, \qquad \overline{\hat{\mathbf{b}}_i} = \frac{1}{N_i}\sum_{j=1}^{N_i}\hat{\mathbf{b}}_j.$$

Proof. The proof relies on a series of algebraic manipulations. The key step is "completing the square," which happens in the last steps of the proof.

$$\begin{aligned}
&\operatorname{var} L^i(\mathbf{x}) + \lambda\left\|\mathbf{x} - \mathbf{p}_i^{(k)}\right\|^2\\
=\;& \frac{1}{N_i}\sum_{j=1}^{N_i}\left\|\hat{\mathbf{A}}_j\mathbf{x} + \hat{\mathbf{b}}_j - \frac{1}{N_i}\sum_{j=1}^{N_i}(\hat{\mathbf{A}}_k\mathbf{x} + \hat{\mathbf{b}}_k)\right\|^2 \;+\; \lambda\left\|\mathbf{x} - \mathbf{p}_i\right\|^2\\
=\;& \frac{1}{N_i}\sum_{j=1}^{N_i}\left\|(\hat{\mathbf{A}}_j - \overline{\hat{\mathbf{A}}_i})\mathbf{x} + (\hat{\mathbf{b}}_j - \overline{\hat{\mathbf{b}}_i})\right\|^2 \;+\; \lambda\left\|\mathbf{x} - \mathbf{p}_i\right\|^2
\end{aligned}$$

To simplify notation, let

$$\begin{aligned}
\mathbf{E}_j &= \hat{\mathbf{A}}_j - \overline{\hat{\mathbf{A}}_i},\\
\mathbf{f}_j &= \hat{\mathbf{b}}_j - \overline{\hat{\mathbf{b}}_i}.
\end{aligned}$$

Then, continuing from above,

$$
\begin{aligned}
&= \frac{1}{N_i}\sum_{j=1}^{N_i}\left(\mathbf{E}_j\mathbf{x}+\mathbf{f}_j\right)^{\mathrm{T}}\left(\mathbf{E}_j\mathbf{x}+\mathbf{f}_j\right) \quad + \quad \lambda\left(\mathbf{x}-\mathbf{p}_i\right)^{\mathrm{T}}\left(\mathbf{x}-\mathbf{p}_i\right) \\
&= \mathbf{x}^{\mathrm{T}}\left(\frac{1}{N_i}\sum_{j=1}^{N_i}{\mathbf{E}_j}^{\mathrm{T}}\mathbf{E}_j \;+\; \lambda\mathbf{I}\right)\mathbf{x} + 2\mathbf{x}^{\mathrm{T}}\left(\frac{1}{N_i}\sum_{j=1}^{N_i}{\mathbf{E}_j}^{\mathrm{T}}\mathbf{f}_j \;-\; \lambda\mathbf{p}_i\right) \\
&\qquad\qquad + \frac{1}{N_i}\sum_{j=1}^{N_i}{\mathbf{f}_j}^{\mathrm{T}}\mathbf{f}_j \;+\; \lambda\,{\mathbf{p}_i}^{\mathrm{T}}\mathbf{p}_i \\
&= \mathbf{x}^{\mathrm{T}}\mathbf{H}_i\mathbf{x} + 2\mathbf{x}^{\mathrm{T}}\mathbf{h}_i \;+\; \frac{1}{N_i}\sum_{j=1}^{N_i}{\mathbf{f}_j}^{\mathrm{T}}\mathbf{f}_j \;+\; \lambda{\mathbf{p}_i}^{\mathrm{T}}\mathbf{p}_i \qquad \text{(B.6)} \\
&= \left(\mathbf{x}+{\mathbf{H}_i}^{-1}\mathbf{h}_i\right)^{\mathrm{T}}\mathbf{H}_i\left(\mathbf{x}+{\mathbf{H}_i}^{-1}\mathbf{h}_i\right) - {\mathbf{h}_i}^{\mathrm{T}}\mathbf{H}_i\mathbf{h}_i + \frac{1}{N_i}\sum_{j=1}^{N_i}{\mathbf{f}_j}^{\mathrm{T}}\mathbf{f}_j \;+\; \lambda{\mathbf{p}_i}^{\mathrm{T}}\mathbf{p}_i
\end{aligned}
$$

The matrix $\mathbf{H}_i$ is positive definite since $\lambda > 0$. If λ were zero, $\mathbf{H}_i$ would still be at least positive semidefinite, and additional conditions on the linear maps could be imposed to achieve definiteness. The manipulation in the last step is "completing the square," in which all appearances of $\mathbf{x}$ are grouped into the first quadratic term. Thus the overall quantity is minimized when the first term is minimized. Since $\mathbf{H}_i$ is positive definite, the first term is minimized when $\mathbf{x}+{\mathbf{H}_i}^{-1}\mathbf{h}_i = \mathbf{0}$. Then $\mathbf{x}_{min} = -{\mathbf{H}_i}^{-1}\mathbf{h}_i$. □

Claim B.2. *Let L^i be a collection of N_i affine maps, $L_j(\mathbf{x}) = \hat{\mathbf{A}}_j\mathbf{x}+\hat{\mathbf{b}}_j$, $j = 1,\ldots,N_i$. Let $\mathbf{p}_i \in \mathbb{R}^d$ and let $\lambda > 0$. Let $\mathbf{C} \in \mathbb{R}^{d\times k}$, $k < d$, be full rank and let $\mathbf{d} \in \mathbb{R}^d$. Let*

$$
\mathbf{x}_{min} = \underset{\mathbf{x}=\mathbf{C}\hat{\mathbf{x}}+\mathbf{d},\,\hat{\mathbf{x}}\in\mathbb{R}^k}{\arg\min} \operatorname{var} L^i(\mathbf{x}) + \lambda\left\|\mathbf{x}-\mathbf{p}_i^{(k)}\right\|^2 \qquad \text{(B.7)}
$$

where $\operatorname{var} L^i(\mathbf{x})$ *is given by Equation B.2. Then* $\mathbf{x}_{min}$ *is given by*

$$
\mathbf{x}_{min} = -\mathbf{C}{\mathbf{H}_{c,i}}^{-1}\mathbf{h}_{c,i} + \mathbf{d} \qquad \text{(B.8)}
$$

where

$$
\mathbf{H}_{c,i} = \mathbf{C}^{\mathrm{T}}\mathbf{H}_i\mathbf{C}, \qquad \mathbf{h}_{c,i} = \mathbf{C}^{\mathrm{T}}\left(\mathbf{H}_i\mathbf{d}+\mathbf{h}_i\right), \qquad \text{(B.9)}
$$

and $\mathbf{H}_i$ and $\mathbf{h}_i$ are given by Equations 4.7 and 4.8.

Proof. Since the function to be minimized is the same as in Claim B.1, just over a different

domain, we jump directly to the equivalent of Equation B.6, substituting $\mathbf{x} = \mathbf{C}\hat{\mathbf{x}} + \mathbf{d}$,

$$\begin{aligned}
& \mathbf{x}^{\mathrm{T}}\mathbf{H}_i\mathbf{x} + 2\mathbf{x}^{\mathrm{T}}\mathbf{h}_i \;+\; \frac{1}{N_i}\sum_{j=1}^{N_i}\mathbf{f}_j{}^{\mathrm{T}}\mathbf{f}_j \;+\; \lambda\mathbf{p}_i{}^{\mathrm{T}}\mathbf{p}_i \\
= \;& (\mathbf{C}\hat{\mathbf{x}} + \mathbf{d})^{\mathrm{T}}\mathbf{H}_i(\mathbf{C}\hat{\mathbf{x}} + \mathbf{d}) + 2(\mathbf{C}\hat{\mathbf{x}} + \mathbf{d})^{\mathrm{T}}\mathbf{h}_i \;+\; \frac{1}{N_i}\sum_{j=1}^{N_i}\mathbf{f}_j{}^{\mathrm{T}}\mathbf{f}_j \;+\; \lambda\mathbf{p}_i{}^{\mathrm{T}}\mathbf{p}_i \\
= \;& \hat{\mathbf{x}}^{\mathrm{T}}\mathbf{C}^{\mathrm{T}}\mathbf{H}_i\mathbf{C}\hat{\mathbf{x}} + 2\hat{\mathbf{x}}\mathbf{C}^{\mathrm{T}}(\mathbf{H}_i\mathbf{d} + \mathbf{h}_i) + \mathbf{d}^{\mathrm{T}}\mathbf{H}_i\mathbf{d} + 2\mathbf{d}^{\mathrm{T}}\mathbf{h}_i \;+\; \frac{1}{N_i}\sum_{j=1}^{N_i}\mathbf{f}_j{}^{\mathrm{T}}\mathbf{f}_j \;+\; \lambda\mathbf{p}_i{}^{\mathrm{T}}\mathbf{p}_i \\
= \;& \hat{\mathbf{x}}^{\mathrm{T}}\mathbf{H}_{c,i}\hat{\mathbf{x}} + 2\hat{\mathbf{x}}\mathbf{h}_{c,i} + \mathbf{d}^{\mathrm{T}}\mathbf{H}_i\mathbf{d} + 2\mathbf{d}^{\mathrm{T}}\mathbf{h}_i \;+\; \frac{1}{N_i}\sum_{j=1}^{N_i}\mathbf{f}_j{}^{\mathrm{T}}\mathbf{f}_j \;+\; \lambda\mathbf{p}_i{}^{\mathrm{T}}\mathbf{p}_i \\
= \;& \left(\hat{\mathbf{x}} + \mathbf{H}_{c,i}{}^{-1}\mathbf{h}_{c,i}\right)^{\mathrm{T}}\mathbf{H}_{c,i}\left(\hat{\mathbf{x}} + \mathbf{H}_{c,i}{}^{-1}\mathbf{h}_{c,i}\right) - \mathbf{h}_{c,i}{}^{\mathrm{T}}\mathbf{H}_{c,i}{}^{-1}\mathbf{h}_{c,i} \\
& \qquad + \mathbf{d}^{\mathrm{T}}\mathbf{H}_i\mathbf{d} + 2\mathbf{d}^{\mathrm{T}}\mathbf{h}_i \;+\; \frac{1}{N_i}\sum_{j=1}^{N_i}\mathbf{f}_j{}^{\mathrm{T}}\mathbf{f}_j \;+\; \lambda\mathbf{p}_i{}^{\mathrm{T}}\mathbf{p}_i
\end{aligned}$$

Since $\mathbf{H}_i$ is positive definite, as argued in Claim B.1, the quantity above is minimized by $\hat{\mathbf{x}} = -\mathbf{H}_{c,i}{}^{-1}\mathbf{h}_{c,i}$. But $\mathbf{x} = \mathbf{C}\hat{\mathbf{x}} + \mathbf{d}$, thus $\mathbf{x}_{min}$ is given by Equation B.8. □

B.2 Best Affine Approximations by Completing the Square

This section describes how the best affine approximations, namely the least squares and best L_2, may be computed by "completing the square," the algebraic manipulation applied in the quadratic minimization of the previous section. These computations also illustrate the similarities between the least squares and best L_2 approximations, namely that the best L_2 approximation may be viewed as the infinite data limit of the least squares approximation.

Let $\Theta \in \mathbb{R}^{(d+1)\times c}$,

$$\mathbf{\Theta} = \begin{bmatrix} \hat{\mathbf{A}}^{\mathrm{T}} \\ \hat{\mathbf{b}}^{\mathrm{T}} \end{bmatrix}, \tag{B.10}$$

be the parameterization of the affine function $f_\Theta : \mathbb{R}^d \to \mathbb{R}^c$ given by $f_\Theta(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}$. Note that f_Θ can also be written

$$f_\Theta(\mathbf{x}) = \mathbf{\Theta}^{\mathrm{T}}\begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}. \tag{B.11}$$

Let $\mathcal{Z} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ be the input output data for which we wish to compute the least squares affine approximation. Let

$$\mathbf{d}_i = \begin{bmatrix} \mathbf{y}_i \\ \mathbf{x}_i \\ 1 \end{bmatrix}$$

for $i = 1, \ldots, N$. Let $\mathbf{M}$ be

$$\mathbf{M} = \begin{bmatrix} -\mathbf{I} \\ \boldsymbol{\Theta} \end{bmatrix}.$$

The summed squared error for the data set is given by

$$\sum_{i=1}^{N} \|\mathbf{y}_i - f_\Theta(\mathbf{x}_i)\|^2 = \sum_{i=1}^{N} \mathbf{d}_i{}^{\mathrm{T}} \mathbf{M}^{\mathrm{T}} \mathbf{M} \mathbf{d}_i$$

and the objective is to find $\boldsymbol{\Theta}$ to minimize this quantity.

$$\begin{aligned} \sum_{i=1}^{N} \mathbf{d}_i{}^{\mathrm{T}} \mathbf{M}^{\mathrm{T}} \mathbf{M} \mathbf{d}_i &= \sum_{i=1}^{N} \mathrm{tr}\left(\mathbf{M}^{\mathrm{T}} \mathbf{d}_i \mathbf{d}_i{}^{\mathrm{T}} \mathbf{M}\right) \\ &= \mathrm{tr}\left(\mathbf{M}^{\mathrm{T}} \left(\sum_{i=1}^{N} \mathbf{d}_i \mathbf{d}_i{}^{\mathrm{T}}\right) \mathbf{M}\right) \\ &= \mathrm{tr}\left(\mathbf{M}^{\mathrm{T}} \mathbf{S} \mathbf{M}\right) \end{aligned}$$

where

$$\mathbf{S} = \sum_{i=1}^{N} \mathbf{d}_i \mathbf{d}_i{}^{\mathrm{T}} = \begin{bmatrix} \mathbf{S}_{yy} & \mathbf{S}_{xy}{}^{\mathrm{T}} \\ \mathbf{S}_{xy} & \mathbf{S}_{xx} \end{bmatrix},$$

$$S_{yy} = \sum_{i=1}^{N} \mathbf{y}_i \mathbf{y}_i{}^{\mathrm{T}}, \qquad S_{xy} = \sum_{i=1}^{N} \begin{bmatrix} \mathbf{x}_i \\ 1 \end{bmatrix} \mathbf{y}_i{}^{\mathrm{T}}, \qquad S_{xx} = \sum_{i=1}^{N} \begin{bmatrix} \mathbf{x}_i \\ 1 \end{bmatrix} \begin{bmatrix} \mathbf{x}_i{}^{\mathrm{T}} & 1 \end{bmatrix}. \tag{B.12}$$

Continuing from above,

$$\begin{aligned} \mathrm{tr}\left(\mathbf{M}^{\mathrm{T}} \mathbf{S} \mathbf{M}\right) &= \mathrm{tr}\left(\mathbf{S}_{yy} - \boldsymbol{\Theta}^{\mathrm{T}} \mathbf{S}_{xy} - \mathbf{S}_{xy}{}^{\mathrm{T}} \boldsymbol{\Theta} + \boldsymbol{\Theta}^{\mathrm{T}} \mathbf{S}_{xx} \boldsymbol{\Theta}\right) \\ &= \mathrm{tr}\left(\mathbf{S}_{yy} + \left(\boldsymbol{\Theta} - \mathbf{S}_{xx}{}^{-1} \mathbf{S}_{xy}\right)^{\mathrm{T}} \mathbf{S}_{xx} \left(\boldsymbol{\Theta} - \mathbf{S}_{xx}{}^{-1} \mathbf{S}_{xy}\right) - \mathbf{S}_{xy}{}^{\mathrm{T}} \mathbf{S}_{xx}{}^{-1} \mathbf{S}_{\mathbf{xy}}\right) \\ &= \mathrm{tr}\left(\mathbf{S}_{yy} - \mathbf{S}_{xy}{}^{\mathrm{T}} \mathbf{S}_{xx}{}^{-1} \mathbf{S}_{\mathbf{xy}}\right) + \mathrm{tr}\left(\left(\boldsymbol{\Theta} - \mathbf{S}_{xx}{}^{-1} \mathbf{S}_{xy}\right)^{\mathrm{T}} \mathbf{S}_{xx} \left(\boldsymbol{\Theta} - \mathbf{S}_{xx}{}^{-1} \mathbf{S}_{xy}\right)\right) \end{aligned}$$

Notice that $\boldsymbol{\Theta}$ only appears in the second term. The matrix $\mathbf{S}_{xx}$ is positive semi-definite, and positive definite if the affine hull of the $\mathbf{x}_i$'s is $\mathbb{R}^d$, a condition on the richness of the data set. Thus the second term is greater than or equal to zero, since it is the trace of a positive semi-definite matrix. Thus the quantity is minimized by choosing

$$\boldsymbol{\Theta} = \mathbf{S}_{xx}{}^{-1} \mathbf{S}_{xy}, \tag{B.13}$$

where $\mathbf{S}_{xx}$ and $\mathbf{S}_{xy}$ are given by Equation B.12, giving the least squares solution. Note that thankfully this solution is exactly the same as given in Section 4.1.1.

Next we proceed to the analogous problem of computing the best L_2 affine approximation to a function $f : D \to \mathbb{R}^c$, $D \subset \mathbb{R}^d$. The affine function f_Θ is parameterized as above, and $\mathbf{M}$ is defined as above. The square of the L_2 norm between f and f_Θ is given by

$$\|f - f_\Theta\|^2 = \int_D \begin{bmatrix} f(\mathbf{x})^{\mathrm{T}} & \mathbf{x}^{\mathrm{T}} & 1 \end{bmatrix} \mathbf{M} \mathbf{M}^{\mathrm{T}} \begin{bmatrix} f(\mathbf{x}) \\ \mathbf{x} \\ 1 \end{bmatrix} d\mathbf{x},$$

which is analogous to the expression for the squared error in the least squares case. Basically, the summation is replaced with the integral, and $\mathbf{y}_i$ is replaced with $f(\mathbf{x})$. Thus an argument analogous to that for least squares above shows that the parameterization $\boldsymbol{\Theta}$ for the best L_2 affine approximation to f is given by

$$\boldsymbol{\Theta} = {\mathbf{S}_{xx}}^{-1}\mathbf{S}_{xy}, \tag{B.14}$$

where

$$\mathbf{S}_{xx} = \int_D \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \begin{bmatrix} \mathbf{x}^{\mathrm{T}} & 1 \end{bmatrix} d\mathbf{x}, \qquad \mathbf{S}_{xy} = \int_D \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} f(\mathbf{x})^{\mathrm{T}}\, d\mathbf{x}. \tag{B.15}$$

The similarity in form between the least squares and best L_2 affine approximations provides insight into why, under appropriate statistical assumptions, the limit as the quantity of data goes to infinity of the least squares affine approximation is the best L_2 approximation.

BIBLIOGRAPHY

BIBLIOGRAPHY

[AM94] Y. S. Abu-Mostafa, *Learning from hints*, Journal of Complexity **10** (1994), 165–178.

[AMS97] C. G. Atkeson, A. W. Moore, and S. Schaal, *Locally weighted learning*, Artificial Intelligence Review **11** (1997), no. 1–5, 11–73.

[Arn89] V. I. Arnold, *Mathematical methods of classical mechanics*, Springer, 1989.

[BA76] I. Babuska and A. K. Aziz, *On the angle condition in the finite element method*, SIAM Journal on Numerical Analysis **13** (1976), no. 2, 214–226.

[Bai94] M. J. Baines, *Algorithms for optimal discontinuous piecewise linear and constant L_2 fits to continuous functions with adjustable nodes in one and two dimensions*, Mathematics of Computation **62** (1994), no. 206, 645–669.

[Bal96] Raja Balasubramanian, *The use of spectral regression in modeling halftone color printers*, Optical Society of America Annual Meeting, Rochester NY, October, 1996.

[Bal03] R. Bala, *Inverse problems in color device characterization*, IS&T/SPIE Annual Symposium on Electronic Imaging: Science and Technology, 2003.

[Bar93] A. R. Barron, *Universal approximation bounds for superpositions of a sigmoidal function*, IEEE Transactions on Information Theory **39** (1993), no. 3, 930–945.

[Bar02] Alexander Barvinok, *A course in convexity*, American Mathematical Society, 2002.

[Bat96] Klaus-Jürgen Bathe, *Finite element procedures*, Prentice Hall, New Jersey, 1996.

[BCSW78] D. L. Barrow, C. K. Chui, P. W. Smith, and J. D. Ward, *Unicity of best mean approximation by second order splines with variable knots*, Mathematics of Computation **32** (1978), no. 144, 1131–1143.

[BDH96] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa, *The quickhull algorithm for convex hulls*, ACM Transactions on Mathematical Software **22** (1996), no. 4, 469–483.

[Bla83] Raymond N. Blair (ed.), *The lithographers manual*, 7 ed., The Graphic Arts Technical Foundation, inc., 1983.

[Bow81] A. Bowyer, *Computing Dirichlet tessellations*, The Computer Journal **24** (1981), no. 2, 162–166.

[Bro79] Kevin Q. Brown, *Voronoi diagrams from convex hulls*, Information Processing Letters **9** (1979), no. 5, 223–228.

[Che66] E. W. Cheney, *Introduction to approximation theory*, McGraw-Hill Book Co., 1966.

[Che84] Chi-Tsong Chen, *Linear system theory and design*, Oxford University Press, 1984.

[Cho82] Jeff Chow, *On the uniqueness of best $l_2[0,1]$ approximation by piecewise polynomials with variable breakpoints*, Mathematics of Computation **39** (1982), no. 160, 571–585.

[Chu88] C. K. Chui, *Multivariate splines*, Society for Industrial and Applied Mathematics, 1988.

[CL87] C. K. Chui and M. J. Lai, *On multivariate vertex splines and applications*, Topics in Multivariate Approximation (C. K. Chui, L. L. Schumaker, and F. Utreras, eds.), Academic Press, New York, 1987, pp. 19–36.

[CWK02] Noah J. Cowan, Joel D. Weingarten, and Daniel E. Koditschek, *Visual servoing via navigation functions*, Transactions on Robotics and Automation (2002), 521–533.

[Dav75] P. J. Davis, *Interpolation and approximation*, Dover Publications, Inc., 1975.

[dB01] Carl de Boor, *A practical guide to splines*, Springer, 2001.

[dBD83] C. de Boor and R. DeVore, *Approximation by smooth multivariate splines*, Transactions of the American Mathematical Society **276** (1983), no. 2, 775–788.

[dBHR93] C. de Boor, K. Höllig, and S. Riemenschneider, *Box splines*, Springer-Verlag, 1993.

[dBvKOS97] Mark de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational geometry: algorithms and applications*, Springer-Verlag, 1997.

[DEG99] Tamal K. Dey, Herbert Edelsbrunner, and Sumanta Guha, *Computational topology*, Advances in Discrete and Computational Geometry, American Mathematical Society, 1999, pp. 109–143.

[DLR90a] N. Dyn, D. Levin, and S. Rippa, *Algorithms for the construction of data dependent triangulations*, Algorithms for Approximation, II, Chapman and Hall, 1990, pp. 185–192.

[DLR90b] ———, *Data dependent triangulations for piecewise linear interpolation*, IMA Journal of Numerical Analysis **10** (1990), no. 1, 137–54.

[DNT02] C. B. Duke, J. Noolandi, and T. Thieret, *The surface science of xerography*, Surface Science **500** (2002), 1005–1023.

[Ede01] Herbert Edelsbrunner, *Geometry and topology for mesh generation*, Cambridge University Press, 2001.

[EM90] Herbert Edelsbrunner and Ernst Peter Mücke, *Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms*, ACM Transactions on Graphics **9** (1990), no. 1, 66–104.

[ES96] H. Edelsbrunner and N.R. Shah, *Incremental topological flipping works for regular triangulations*, Algorithmica **15** (1996), no. 3, 223–241.

[FB01] Simone Fiori and Paolo Bucciarelli, *Probability density estimation using adaptive activation function neurons*, Neural Processing Letters **13** (2001), no. 1, 31–42.

[Fio01] Simone Fiori, *Probability density function learning by unsupervised neurons*, International Journal of Neural Systems **11** (2001), no. 5, 399–417.

[FK98] S. Furry and J. Kainz, *Rapid algorithm development applied to engine management systems*, Proc. 1998 SAE Intl Congress & Exposition, 1998.

[Fri91] J. H. Friedman, *Multivariate adaptive regression splines*, The Annals of Statistics **19** (1991), no. 1, 1–141.

[GKK+99] Richard E. Groff, Pramod P. Khargonekar, Daniel E. Koditschek, Tracy E. Thieret, and L.K. Mestha, *Modeling and control of color xerographic processes*, IEEE Conference on Decision and Control, vol. 2, 1999, pp. 1697–1702.

[GKK00] Richard E. Groff, Daniel E. Koditschek, and Pramod P. Khargonekar, *Piecewise linear homeomorphisms: the scalar case*, International Joint Conference on Neural Networks, 2000.

[GKK03] Richard E. Groff, Pramod P. Khargonekar, and Daniel E. Koditschek, *A local convergence proof for the* MINVAR *algorithm for computing continuous piecewise linear approximations*, SIAM Journal on Numerical Analysis (2003), to appear.

[GKKT00] Richard E. Groff, Daniel E. Koditschek, Pramod P. Khargonekar, and Tracy E. Thieret, *Representation of color space transformations for effective calibration and control*, IS&T's NIP16: International Conference on Digital Printing Technology, 2000, pp. 255–260.

[Gre75] John A. Gregory, *Error bounds for linear interpolation on triangles*, The Mathematics of Finite Elements and Applications II (J. R. Whiteman, ed.), Academic Press, 1975, pp. 163–170.

[Gro03] Richard E. Groff, MINVAR*: A C++ implementation*, Tech. report, Computer Science, University of Michigan, `http://www.cs.umich.edu/eecs/research/techreports/compsci.html`, 2003.

[GT03] Richard E. Groff and Tracy E. Thieret, *Printers*, Wiley Encyclopedia of Electrical and Electronics Engineering Online (J. Webster, ed.), John Wiley & Sons, Inc., 2003, `http://www.mrw.interscience.wiley.com/eeee/`.

[GV96] Gene E. Golub and Charles F. Van Loan, *Matrix computations*, John Hopkins University Press, 1996.

[GW96] R. C. Gayle and J. M. Wolfe, *Unicity in piecewise polynomial L_1-approximation via an algorithm*, Mathematics of Computation **65** (1996), no. 214, 647–660.

[Háj69] Jaroslav Hájek, *A course in nonparametric statistics*, Holden-Day, 1969.

[Hat01] Allen Hatcher, *Algebraic topology*, Cambridge University Press, 2001.

[Hei00] Janne Heikkilä, *Geometric camera calibration using circular control points*, IEEE Transactions on Pattern Analysis and Machine Intelligence **22** (2000), no. 10, 1066–1076.

[HNY98] N. Hai, J. Ni, and J. Yuan, *Generalized model formulation technique for error synthesis and error compensation on machine tools*, Proceedings of the NAMRX XXVI Conference, Society of Manufacturing Engineers, 1998.

[Isi95] Alberto Isidori, *Nonlinear control systems*, 3rd ed., Springer, 1995.

[Joe89] Barry Joe, *Three-dimensional triangulations from local transformations*, SIAM J. Sci. Stat. Comput. **10** (1989), 718–741.

[Jul99] Pedro Marcelo Julián, *A high level canonical piecewise linear representation: Theory and applications*, Ph.D. thesis, Universidad Nacional Del Sur, http://www.pedrojulian.com, 1999.

[JW99] M. I. Jordan and D. M. Wolpert, *Computational motor control*, The Cognitive Neurosciences (M. Gazzaniga, ed.), MIT Press, 2nd ed., 1999.

[Kan97] H. R. Kang, *Color technology for electronic imagining devices*, SPIE Optical Engineering Press, 1997.

[Kan02] ———, *Digital color halftoning*, IEEE Press, 2002.

[Kha80] O. Khatib, *Commande dynamique dans l'espace opérational des robots manipulateurs en présence d'obstacles*, Ph.D. thesis, École nationale Supéieure de l'Aeéronatique et de l'Espace, France, 1980.

[Kio80] J. B. Kioustelidis, *Optimal segmented approximations*, Computing **24** (1980), no. 1, 1–8.

[Kio81] ———, *Optimal segmented polynomial L_s-approximations*, Computing **26** (1981), no. 3, 239–246.

[KK92] K. Kanamori and H. Kotera, *Color-correction technique for hard copies by 4-neighbors interpolation method*, J. Imaging Sci. Technol. **36** (1992), no. 1, 73–80.

[KM78] O. Khatib and J.-F. Le Maitre, *Dynamic control of manipulators operating in a complex environment*, Proc. Int. CISM-IFToMM Symp. ((Udine, Italy)), 1978, pp. 267–282.

[KNPH95] J. M. Kasson, S. I. Nin, W. Plouffe, and J. L. Hafner, *Performing color space conversions with three dimensional linear interpolation*, J. Electron Imaging **4** (1995), no. 3, 226–250.

[KR90] D. E. Koditschek and E. Rimon, *Robot navigation functions on manifolds with boundary*, Advances in Applied Mathematics **11** (1990), 412–442.

[Kro84] B. H. Krogh, *A generalized potential field approach to obstacle avoidance*, Proc. SME Conf. Robotics Res., 1984.

[Law77] C. L. Lawson, *Software for C^1 surface interpolation*, Mathematical Software III (J. R. Rice, ed.), Academic Press, New York, 1977, pp. 161–194.

[Law86] Charles L. Lawson, *Properties of n-dimensional triangulations*, Computer Aided Geometric Design **3** (1986), 231–246.

[Lee91] C. Lee, *Regular triangulations of convex polytopes*, Applied Geometry and Discrete Mathematics: The Victor Klee Festschrift (P. Gritzmann and B. Sturmfels, eds.), American Mathematical Society, 1991.

[Lju99] Lennart Ljung, *System identification: Theory for the user*, 2nd ed., Prentice Hall, 1999.

[Lor66] G. G. Lorentz, *Approximation of functions*, Holt, Rhinehart and Winston, 1966.

[Lue69] David G. Luenberger, *Optimization by vector space methods*, John Wiley & Sons, Inc., 1969.

[MP69] M. Minsky and S. Papert, *Perceptrons : an introduction to computational geometry*, MIT Press, 1969.

[Mun75] James R. Munkres, *Topology: A first course*, Prentice Hall, Inc., 1975.

[MWD+96] L. K. Mestha, Y. R. Wang, S. Dianat, E. Jackson, T. Thieret, P. P. Khargonekar, and D. E. Koditschek, *Toward a control oriented model of xerographic marking engines*, IEEE Conference on Decision and Control, vol. 4, 1996, pp. 4837–4843.

[Nad86] E. Nadler, *Piecewise linear best L_2 approximation on triangulations*, Approximation Theory V (C. K. Chui, L. L. Schumaker, and J. D. Ward, eds.), Academic Press, Boston, 1986.

[Phi70] G. M. Phillips, *Error estimates for best polynomial approximations*, Approximation Theory (A. Talbot, ed.), New York: Academic Press, 1970, pp. 1–5.

[Pon52] L. S. Pontryagin, *Foundations of combinatorial topology*, Dover Publications, Inc., 1952.

[Poo94] H. Vincent Poor, *An introduction to signal detection and estimation*, Springer Verlag, 1994.

[PS85] Franco P. Preparata and Michael Ian Shamos, *Computational geometry : an introduction*, Springer-Verlag, 1985.

[PS93] Damodar M. Pai and B. E. Springett, *Physics of electrophotography*, Reviews of Modern Physics **65** (1993), no. 1, 163–211.

[Rai86] M. H. Raibert, *Legged robots that balance*, MIT Press, 1986.

[Raj91] V. T. Rajan, *Optimality of the Delaunay triangulation in* $\mathbb{R}^d$, Proc. 7th Ann. Symp. on Computational Geometry, 1991, pp. 357–363.

[Raj94] ———, *Optimality of the Delaunay triangulation in* $\mathbb{R}^d$, Discrete and Computational Geometry **12** (1994), no. 2, 189–202.

[Rip90] Samuel Rippa, *Minimal roughness property of the delaunay triangulation*, Computer Aided Geometric Design **7** (1990), 489–497.

[Rip92] Shmuel Rippa, *Long and thin triangles can be good for linear interpolation*, SIAM Journal on Numerical Analysis **29** (1992), no. 1, 257–270.

[RK91] E. Rimon and D. E. Koditschek, *The construction of analytic diffeomorphisms for exact robot navigation on star worlds*, Transactions of the American Mathematical Society **327** (1991), no. 1, 71–115.

[RK92] ———, *Exact robot navigation using artificial potential fields*, IEEE Transactions on Robotics and Automation **8** (1992), no. 5, 501–518.

[SA98] Stefan Schaal and Christopher Atkeson, *Constructive incremental learning from only local information*, Neural Computation **10** (1998), no. 8, 2047–2084.

[Sch88] L. B. Schein, *Electrophotography and development physics*, 2 ed., Springer-Verlag, 1988.

[Sch98] William Schwind, *Spring loaded inverted pendulum running : a plant model*, Ph.D. thesis, University of Michigan, 1998.

[SHKT97] C. J. Stone, M. H. Hansen, C. Kooperberg, and Y. K. Truong, *Polynomial splines and their tensor products in extended linear modeling*, The Annals of Statistics **25** (1997), no. 4, 1371–1470.

[SK00] W. J. Schwind and D. E. Koditschek, *Approximating the stance map of a 2 dof monoped runner*, Journal of Nonlinear Science **10** (2000), no. 5, 533–568.

[SP95] Eugene V. Shikin and Alexander I. Plis, *Handbook on splines for the user*, CRC Press, 1995.

[Spa66] E. H. Spanier, *Algebraic topology*, McGraw-Hill, 1966.

[Spi65] Michael Spivak, *Calculus on manifolds*, Perseus Books, 1965.

[Str88] Gilbert Strang, *Linear algebra and its applications*, 3rd ed., Harcourt, Brace, Jovanovich, Publishers, 1988.

[TB97] Y. Tourigny and M. J. Baines, *Analysis of an algorithm for generating locally optimal meshes for* L_2 *approximation by discontinuous piecewise polynomials*, Mathematics of Computation **66** (1997), no. 218, 623–650.

[TH98] Yves Tourigny and Frank Hülsemann, *A new moving mesh algorithm for the finite element solution of variational problems*, SIAM Journal on Numeral Analysis **35** (1998), no. 4, 1416–1438.

[THB95] Tracy E. Thieret, Thomas A. Henderson, and Michael A. Butler, *Method and control system architecture for controlling tone reproduction in a printing device*, U.S. Patent 5,471,313, November 28, 1995.

[Tsa87] R.Y. Tsai, *A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf tv cameras and lenses*, IEEE Transactions on Robotics and Automation **3** (1987), no. 4, 323–344.

[TY00] Alain Trouvé and Laurent Younes, *On a class of diffeomorphic matching problems in one dimension*, SIAM Journal of Control and Optimization **39** (2000), no. 4, 1112–1135.

[Uli87] R. Ulichney, *Digital halftoning*, MIT Press, 1987.

[Wat81] D. F. Watson, *Computing the n-dimensional Delaunay tesselation with application to Voronoi polytopes*, The Computer Journal **24** (1981), no. 2, 167–172.

[WBGK03] Joel D. Weingarten, Martin Buehler, Richard E. Groff, and Daniel E. Koditschek, *Gait generation and optimization for legged robots*, Tech. report, Computer Science, University of Michigan, `http://www.cs.umich.edu/eecs/research/techreports/compsci.html`, 2003.

[Web94] Roger Webster, *Convexity*, Oxford University Press, 1994.

[Wen92] Juyang Weng, *Camera calibration with distortion models and accuracy evaluation*, IEEE Transactions on Pattern Analysis and Machine Intelligence **14** (1992), no. 10, 965–980.

[WGK03] E. R. Westervelt, J. W. Grizzle, and D. Koditschek, *Hybrid zero dynamics of planar biped walkers*, Transactions on Automatic Control **48** (2003), no. 1, 42–56.

[ZD89] J. C. Ziegert and P. Datseris, *Basic considerations for robot calibration*, International Journal of Robotics and Automation **4** (1989), no. 3, 158–66.