
Computation for the Kolmogorov Superposition Theorem

Jonas Actor



Thesis: Master of Arts
Computational and Applied Mathematics
Rice University, Houston, Texas (May 2018)

RICE UNIVERSITY

**Computation for the Kolmogorov Superposition
Theorem**

by

Jonas Actor

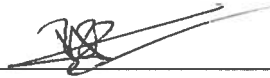
A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Master of Arts

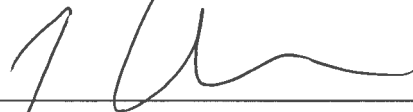
APPROVED, THESIS COMMITTEE:



Matthew G. Knepley, Committee Chair
Assistant Professor
Computational and Applied Mathematics



Beatrice Riviere
Noah G. Harding Chair
Professor
Department Chair
Computational and Applied Mathematics



Jesse Chan
Assistant Professor
Computational and Applied Mathematics

Houston, Texas

May, 2018

Abstract

This thesis presents the first known method to compute Lipschitz continuous inner functions for the Kolmogorov Superposition Theorem. While the inner functions of these superpositions can be Lipschitz continuous, previous algorithms that compute inner functions yield only Hölder continuous functions. These Hölder continuous functions suffer from high storage-to-evaluation complexity, thereby rendering the Kolmogorov Superposition Theorem impractical for computation. Once this concern is addressed, the Kolmogorov Superposition Theorem becomes an effective tool in dimension reduction to represent multivariate functions as univariate expressions, with applications in encryption, video compression, and image analysis. In this thesis, I posit sufficient criteria to iteratively construct a Lipschitz continuous inner function. I demonstrate that implementations of the predominant approach for such a Lipschitz construction do not satisfy these conditions. Instead, I manipulate Hölder continuous inner functions to create Lipschitz continuous reparameterizations. I show these reparameterizations meet my sufficient conditions, thereby guaranteeing that these reparameterized functions are inner functions for the Kolmogorov Superposition Theorem.

Acknowledgements

Thank you to the mentors, teachers, professors, colleagues, friends, and family who have made this possible. A special thank you is deserved for my adviser, Dr. Matthew Knepley, for all his work, conversations, ideas, and feedback as I have worked under his guidance on the Kolmogorov Superposition Theorem. I am also grateful to the other members of my committee, Dr. Riviere and Dr. Chan, for their helpful insight and opinions.

The author acknowledges support from the Ken Kennedy Institute Computer Science & Engineering Enhancement Fellowship, funded by the Rice Oil & Gas HPC Conference.

Contents

List of Illustrations	vii
1 Introduction	1
1.1 The Curse of Dimensionality	1
1.2 Outline of This Thesis	3
2 Background	6
2.1 Motivation: A Question of Nomography	6
2.2 Refinements to KST	9
2.2.1 Reducing the Number of Functions	9
2.2.2 Improving Smoothness	12
2.3 Computational KST	14
2.3.1 Exact KST Computation	14
2.3.2 Approximations to KST Inner Functions	15
2.3.3 Approximations using KST Structure	16
2.4 Applications of KST	18
2.5 Unresolved and Related Questions	20
3 Sufficient Conditions for Constructing KST Inner Functions	23
3.1 Kolmogorov's Original Conditions	24
3.2 Sprecher's Reduction	28
3.3 Conditions for Improved Smoothness of the Inner Function	31
3.4 A Summary of Conditions	38
3.4.1 Conditions on Spatial Decompositions	38
3.4.2 Conditions on Function Values	39

4	The Fridman Strategy	41
4.1	Characterization of the Fridman Strategy	41
4.2	The Fridman Strategy with Fridman's Disjoint Image Condition	44
4.2.1	Posing an Algorithm	45
4.2.2	Find Stage	46
4.2.3	Plug Stage	49
4.2.4	Break Stage	53
4.2.5	Proof of the Algorithm	57
4.3	The Fridman Strategy with the Conservative Disjoint Image Condition . . .	64
4.3.1	Motivation for Using the Conservative Disjoint Image Condition . . .	65
4.3.2	Find and Plug Stages	68
4.3.3	Break Stage	68
4.3.4	Implications	82
5	A Reparameterization Argument	86
5.1	Hedberg and Kahane's Reformulation	87
5.1.1	A Geometric Interpretation of KST	87
5.1.2	Guarantee of Lipschitz Inner Functions	92
5.2	A Hölder Continuous Inner Function	95
5.3	Reparameterization	99
5.4	Verification of Sufficient Conditions for KST	103
A	Statement of Hilbert's 13th Problem	110
B	Proof of the Kolmogorov Superposition Theorem	112
C	Three Aspects of Sprecher's KST Reformulation	120
C.1	Construction of ψ and $\mathcal{T}^{k,q}$	121
C.2	Verification of the Disjoint Image Condition	122
C.3	Justification of Hölder Continuity	124

D Code: The Fridman Strategy	126
E Code: Köppen's Inner Function	139
Bibliography	143

Illustrations

3.1	A set $\mathcal{S}^{k,q}$ that nearly partitions \mathbb{I}^2 . Each square $S_{ij} \in \mathcal{S}^{k,q}$ is the Cartesian product of two intervals $S_{ij} = T_i \times T_j$ where $T_i, T_j \in \mathcal{T}^{k,q}$	25
3.2	Sketch of set of squares $\mathcal{S}^{k,q}$ that satisfy the Disjoint Image Condition. Observe that the image of each square $S_{ij} \in \mathcal{S}^{k,q}$ under Ψ^q is disjoint, and that the image of each S_{ij} under Ψ^q are connected, since Ψ^q is a continuous function and S_{ij} is a connected domain.	26
3.3	Example of intervals $\mathcal{T}^{k,q}$ that satisfy the All But One Condition. Note that in this construction, each line of intervals is a translation of the line below it; this translation trick is a convenient method to make sure that the All But One Condition is satisfied, although it is not necessary.	27
3.4	Kolmogorov's self-similar refinement for $n = 2$. Note that the fundamental unit of the self-scaling includes both the interval and the gap to the right of the interval.	28
3.5	Sketch of the relationship between the sets \mathcal{S}^k and \square^k under the Conservative Disjoint Image Condition. Compare to Figure 3.2; with the Cconservative Disjoint Image Condition, the image intervals Δ_{ij}^k envelope the intervals $\Psi^q(S_{ij}^k)$. Since more space needs to be allocated for the intervals Δ_{ij}^k than $\Psi^q(S_{ij}^k)$, they deemed more <i>conservative</i> , hence the name of the condition. . .	31
3.6	Defining ψ^k during refinement level k . Note that for each $k \in \mathbb{N}$, the function ψ^k is constant on intervals and linear on gaps, where the values at the left endpoints of the intervals are fixed and held constant for all $k' > k$. This process yields a final function ψ that is differentiable almost everywhere, with derivative 0 almost everywhere, and not differentiable on a dense set of points.	34

3.7	Breaking intervals roughly in half. This breaking process is crucial to show that the functions ψ^k converge.	36
3.8	Scenario in which it is necessary to add small segments to maintain the All But One Condition. As adding these small segments increases the slope of the function ψ^{k+1} (see Figure 3.9), it is crucial to choose the segments sufficiently small.	37
3.9	Sketch of how size of plugs changes the slope of ψ^k . Note that larger plugs correspond to a greater slope in the new function ψ^{k+1}	38
4.1	Illustration of the set \mathcal{Q}^c . In this example the set $\mathcal{Q}^c = \{-4, 2\}$	48
4.2	Sketch of scenario for finding two plugs, with ψ^k solid and ψ^{k+1} dotted. Note the symmetry constraint $b_1 - \hat{p}_1 = \hat{p}_2 - a_2$ is enforced.	51
4.3	Implementation of Fridman Strategy Algorithm for $k = 11$. Note that this function is has slope bounded above by one, and is monotonic increasing.	57
4.4	Plot of $\Psi^9(\mathcal{S}^1)$. The tall lines demarcate the start of the image intervals; without them, the overlap of the image intervals would not necessarily be visible. Note that these image intervals clearly fail to be disjoint.	62
4.5	Plot of $\Psi^9(\mathcal{S}^2)$. Again, the tall lines demarcate the start of the image intervals. Due to the significant overlap, even only for the second level of refinement, it is difficult to tell how many intervals overlap with each other; they are clearly not disjoint.	63
4.6	Setup of squares after breaking step k , thereby creating squares $k + 1$. Note that the notation reflects which values are fixed to which squares during refinement: the squares that maintain index i share the function value (for that coordinate direction) as with its parent, whereas the squares with index N reflect a new value that has been assigned.	69

5.1	Relationship of homeomorphisms that enable a Lipschitz reparameterization of arbitrary KST inner functions. On the left are the coordinates as they are mapped between each other, and on the right are the spaces characterized by the homeomorphisms in question.	94
5.2	Plots of Köppen's $\widehat{\psi}^k$ for first few values of k . Note that the The functions $\widehat{\psi}^k$ are self-similar at two alternating scales. For a more detailed analysis of this self-similarity, see [26] and [7].	97
5.3	Plots of ψ^k for first few values of k . Note that the slopes appear to all be bounded, meaning that ψ^k is Lipschitz, and that they appear to converge uniformly as $k \rightarrow \infty$	101
5.4	Plot of $\widehat{\psi}^6$ and ψ^6 . Note that ψ^6 has a much more controlled slope than $\widehat{\psi}^6$. This makes sense, given ψ is Lipschitz continuous whereas $\widehat{\psi}$ is not.	102
5.5	Semilog plot of the largest interval sizes in $\mathcal{T}^k = \sigma(\mathcal{R}^k)$ for the first few values of k . Note that the largest interval size decreases at $O(2^{-k})$; this trend suggests that the Refinement Condition is met.	105
5.6	Shifted families of $\mathcal{T}^k = \sigma(\mathcal{R}^k)$ for $k = 1$. Note that the breaks are located in the same position for each set of intervals, with the breaks of $\sigma(\mathcal{R}^1)$ smaller than that of \mathcal{R}^1 alone; thus, the All But One property is preserved.	107
5.7	Shifted families of $\mathcal{T}^k = \sigma(\mathcal{R}^k)$ for $k = 2$. Note that while the breaks are no longer located in the same position for each set of intervals, the sizes of the breaks in \mathcal{T}^2 are smaller than that of \mathcal{R}^2 ; thus, the All But One property is preserved. Additionally, note that the largest intervals are only roughly half the size of the those at the previous level of refinement, demonstrating the comparative sharpness of the $O(2^k)$ bound presented in Figure 5.5.	108

Chapter 1

Introduction

In this chapter, I frame the problem of why the Kolmogorov Superposition Theorem is of relevance to modern computational science, and how I will approach the task of constructing a Lipschitz continuous inner function for use in the Kolmogorov Superposition Theorem. In Section 1.1, I describe how the Kolmogorov Superposition Theorem relates to the Curse of Dimensionality, and I state the Kolmogorov Superposition Theorem. Then, Section 1.2 outlines the approach I take in this thesis to represent continuous multivariate functions using Lipschitz inner functions. Over the course of this thesis, I describe the difficulties with the predominant existing strategy for to construct these functions, and I pose an alternative approach that uses a different underlying theory to circumvent these difficulties.

1.1 The Curse of Dimensionality

The Curse of Dimensionality manifests when modeling physical phenomena in multiple dimensions: the number of points required for accurate computation grows exponentially with the dimension (Bellman [4]). As a result, specific methods that take advantage of underlying structure are needed to solve high dimensional problems. Examples of such problems include the Hamilton-Jacobi Equations (Darbon and Osher [9]) and PDE-constrained optimal control problems (Biegler et al. [6]). Therefore, it is natural to wonder if it is possible to reduce the number of dimensions pertinent to computation, in order to expedite the solution

of these problems.

In their text on problems in analysis, Pólya and Szegő posed several questions related to representing functions of multiple variables (Pólya and Szegő [36]). Among others, they asked the following question (ibid., p.79):

Are there actually functions of three variables?

It is not difficult to write an expression involving three variables. What this question asks is whether it is systematically possible to use compositions of functions of two (or fewer) variables to express any function of three variables. If so, this reduces the number of inputs required for function evaluation from three inputs, to a sequence of computations involving only two inputs at any given time. Regarding continuous functions, the Kolmogorov Superposition Theorem (KST) answers Pólya and Szegő's question in the negative: there are no true continuous functions of three variables, and there are no true continuous functions of two variables other than addition, as seen in the following theorem.

Theorem 1.1.1 (KST). *Let $f \in C(\mathbb{I}^n) : \mathbb{I}^n \rightarrow \mathbb{R}$ be a continuous multivariate function. There exist univariate continuous functions $\chi_q : \mathbb{I} \rightarrow \mathbb{R}$ and $\psi_{p,q} : \mathbb{R} \rightarrow \mathbb{R}$, where $p = 1, \dots, n$ and $q = 0, \dots, 2n$, such that for all $x = (x_1, \dots, x_p) \in \mathbb{I}^n$,*

$$f(x_1, \dots, x_n) = \sum_{q=0}^{2n} \chi_q \left(\sum_{p=1}^n \psi_{p,q}(x_p) \right). \quad (1.1)$$

The functions $\psi_{p,q}$ do not depend on the function f (Kolmogorov [25]).

It is difficult to utilize this theorem for efficient computation. The functions in Kolmogorov's construction are pathological, being not differentiable on a dense set of points.

Later refinements to KST use functions that are Hölder continuous, which suffer from increasing storage-to-evaluation ratios, thereby limiting the computational use of KST. While there are theoretical results that state KST is possible using functions that are Lipschitz continuous- which do not suffer from such poor ratios- there has not been a successful attempt to perform computation with such Lipschitz continuous functions, to the author's knowledge. This thesis describes a method to enable such computation; the steps this thesis takes to do so are outlined in the next section.

1.2 Outline of This Thesis

This thesis formulates an algorithm that defines a Lipschitz continuous function for the inner summation of Equation 1.1 in KST. Chapter 2 summarizes the relevant background to this task. Chapter 2 steps through the history of this problem, beginning with the problem as posed by Hilbert, and proceeding through Kolmogorov's proof and the successive refinements to KST. Equally important are the previous attempts to compute functions using KST, including those that utilize Hölder continuous functions, as they are the foundation for this thesis. Also mentioned are various approximations related to KST, some being closer than others. This chapter concludes with a description of various applications to which KST can be applied, along with a list of some related open questions whose answers would advance the goal of computing using KST.

Next, this thesis outlines the sufficient conditions for computing a Lipschitz continuous inner function. Chapter 3 dissects the various conditions, given by Kolmogorov and others, that provide the theoretical foundation for computational KST. Ultimately, these conditions

can be grouped into two categories: conditions on a spatial partition defined during the computational process, and conditions on the function values that the inner function attains. These conditions build a framework for an algorithmic approach to KST.

Chapter 4 uses the framework in the previous chapter to formulate two algorithms for computational KST. Both of these algorithms use the Fridman Strategy, the predominant method to try to construct Lipschitz inner functions for KST (Fridman [15]). These approaches differ in how they enforce the sufficient conditions described in Chapter 3. Algorithms to enforce these conditions are described in detail, with particular care being given to prove that these algorithm accomplish their goal. These algorithms make several significant design choices, which determine how strictly the algorithm’s resulting function adheres to the sufficient conditions for KST. Chapter 4 concludes that both approaches fail to construct Lipschitz continuous inner functions, and the chapter discusses the structural difficulties of the Fridman Strategy that limit its usefulness.

After discussing the general failures of the Fridman Strategy, Chapter 5 revisits the task of computing a Lipschitz inner function using a different proof of KST. The arguments in this chapter rely on a geometric formulation of KST, first provided by Hedberg and Kahane ([19], [22]). This geometric viewpoint emphasizes that a Lipschitz inner function can be constructed from a Hölder continuous inner function using a reparameterization using the arclength of the inner function. I describe how such a task can be approached computationally, and I show the resulting function satisfies the sufficient conditions for KST previously discussed in Chapter 3.

While this thesis describes various implementations and algorithms to construct inner functions in the Kolmogorov Superposition Theorem, this thesis does not attempt to imple-

ment the construction of the corresponding outer functions. For the results of this thesis to be of benefit, an efficient scheme to compute the outer functions in KST is required. This topic is left for now as future work to the wider community, although I aim to revisit this topic soon.

Chapter 2

Background

This chapter frames the task of developing an algorithm to compute Lipschitz-continuous variants of the Kolmogorov Superposition Theorem (KST), thereby enabling KST to be used as a computational tool to reduce multivariate functions to univariate expressions. In Section 2.1 I motivate the original problem in nomography, from which grew Kolmogorov's theorem. Section 2.2 explores the refinements to KST that allowed KST to be approached computationally. In Section 2.3 I detail these computational approaches, primarily as a means of compression, and describe how these solutions have been previously applied to multivariate approximation. After highlighting several successful applications of KST in Section 2.4, I conclude this chapter in Section 2.5 with a brief description of open problems related to the Kolmogorov Superposition Theorem and its computation, highlighting how these open questions reflect upon the goal of computing Lipschitz continuous inner functions for KST.

2.1 Motivation: A Question of Nomography

The mathematician Andrei Kolmogorov was motivated to examine which types of functions could be constructed out of superpositions of other functions so as to answer a fundamental question posed by the mathematician David Hilbert in his 1900 Paris address at the second International Congress of Mathematics. This question, Hilbert's 13th problem, is a problem

in the field of nomography, the systematic parameterization of curves to define solutions of equations in terms of fewer variables than the original formulation (Hilbert [20]). Specifically, Hilbert's 13th problem inquires about the representability of the roots of a 7th degree polynomial in terms of operations that involve functions of only two variables (Hilbert [20]); a transcription of Hilbert's statement of this problem from the Paris address is included in appendix A. Hilbert originally conjectured that such a representation was not possible, although it is unclear whether he had limited his consideration to the smaller class of nomographic parameterizations given by algebraic curves.

Kolmogorov approached this problem from a fundamentally different angle by examining the level sets of space-filling curves, instead of algebraic curves. Such space-filling curves were considered pathological at the time Hilbert first posed the 13th problem, so it is unsurprising that Hilbert did not consider such functions in the context of nomography. Using these curves, Kolmogorov proved that any functions of more than three variables, such as an equation describing the roots of monic 7th degree polynomials in terms of its coefficients, could be reduced to expressions involving strictly three variables (Kolmogorov [24]). By refining this approach, Kolmogorov's student Vladimir Arnold proved that functions of three variables could be reduced to expressions involving only two variables (Arnold [1]), thereby answering Hilbert's problem.

However, Kolmogorov was unsatisfied with the complexity of his and Arnold's proofs, and he continued to work at this problem to reduce the number of variables required even further. His work culminated in his Superposition Theorem, which he proved in 1957 (Kolmogorov [25]). This theorem stated that any continuous multivariate function could be reduced to a superposition of univariate functions and addition. Precisely, the Kolmogorov Superposition

Theorem (KST) is as follows:

Theorem 1.1.1 (KST). *Let $f \in C(\mathbb{I}^n) : \mathbb{I}^n \rightarrow \mathbb{R}$ be a continuous multivariate function.*

There exist univariate continuous functions $\chi_q : \mathbb{I} \rightarrow \mathbb{R}$ and $\psi_{p,q} : \mathbb{R} \rightarrow \mathbb{R}$, where $p = 1, \dots, n$ and $q = 0, \dots, 2n$, such that for all $x = (x_1, \dots, x_n) \in \mathbb{I}^n$,

$$f(x_1, \dots, x_n) = \sum_{q=0}^{2n} \chi_q \left(\sum_{p=1}^n \psi_{p,q}(x_p) \right). \quad (1.1)$$

The functions $\psi_{p,q}$ do not depend on the function f (Kolmogorov [25]).

Using refinements to this theorem, the $2n + 1$ functions χ_q can be reduced to a single function univariate function χ ; this refinement is discussed in Section 2.2. As a result, KST describes a mapping between continuous functions of multiple variables to a single univariate function - the outer function in the superposition above. In Equation 1.1, this maps f to χ . This mapping enables representing multivariate equations via their equivalent univariate expressions; if this mapping can be conducted systematically, solving multivariate equations can be achieved by solving the corresponding univariate equations, significantly reducing the number of variables from n to one. The construction of outer KST functions remains an outstanding task that this thesis fails to address, although the author aims to return to this problem in future work.

Kolmogorov proved his theorem constructively, using a self-similar tiling of \mathbb{I}^n to define iteratively the inner functions $\psi_{p,q}$ and the outer functions χ_q . This constructive proof is the foundation of all known computational approaches to KST and is described in detail in section 3.1.

2.2 Refinements to KST

While Kolmogorov’s proof outlines a constructive approach to superpositions, other mathematicians later worked to reformulate KST in ways that make KST more amenable to algorithmic computation. These reformulations focused on reducing the number of functions needed for KST representation and on improving the smoothness of those functions. Only with these improvements is computational KST a possibility; to design an algorithm to compute Lipschitz continuous inner functions, it is necessary to understand the theory underlying these improvements.

2.2.1 Reducing the Number of Functions

Several of these reformulations reduced the number of functions needed for KST representation, as one of the drawbacks of Kolmogorov’s constructivist argument was the necessity to define $2n^2 + n$ distinct inner functions and $2n + 1$ outer functions. Unfortunately, Sternfeld showed that the number of terms in each sum is the best possible, i.e. it is impossible to represent all continuous functions using fewer than $2n + 1$ terms in the summand for the outer function, and with fewer than n terms in the inner summand (Sternfeld [45]). This result provides a lower bound on the number of functions I must define algorithmically in my efforts to turn KST into a computational tool. Because of Sternfeld’s work, mathematicians turned to reducing the number of unique functions needed, attempting to reuse variants of the same function to encode the representation instead.

Subsequent papers reduced the number of unique inner functions for KST from $2n^2 + n$ to a single function (Sprecher [38]). Instead of defining functions $\psi_{p,q}$ individually, Sprecher

showed that it is sufficient to define a single function ψ on a larger (still compact) domain and then reconstruct $\psi_{p,q}$ from this parent function using shifting and scaling by preassigned coefficients ε and $\lambda_1, \dots, \lambda_n$, which depend only on the spatial dimension n :

$$\psi_{p,q}(x) = \lambda_p \psi(x + q\varepsilon). \quad (2.1)$$

The shift ε is only required to be no larger than $\frac{1}{2n}$. The scaling factors $\lambda_1, \dots, \lambda_n$ are required to be integrally independent*, which is the following condition on rational numbers $x_1, \dots, x_n \in \mathbb{Q}$:

$$\text{If } \sum_{i=1}^n \lambda_i x_i = 0, \quad \text{then } x_i = 0 \text{ for } i = 1, \dots, n. \quad (2.2)$$

This property is necessary to complete Sprecher's proof that his formulation demonstrates the sufficient conditions Kolmogorov had previously established (Kolmogorov [25]). In this thesis, particularly in my analysis of the Fridman Strategy in Chapter 4, I utilize this idea of shifting and scaling to define only one inner function; I describe this process in further detail in section 3.2.

Using shifted arguments to reduce the number of unique functions necessary was developed to reduce the number of outer functions as well (Lorentz [32]). By observing that the images of the inner functions were bounded, Lorentz noted that it was sufficient to collect the various outer functions χ_q and to assign shifts by a factor δ sufficiently large, so that $\chi_q(y) = \chi(y + \delta q)$. This new function χ is defined by the same process that was used to define χ_q , so this change makes little difference from a computational standpoint. This single

*By rescaling by the greatest common denominator, this is identical to being rationally independent.

function χ encodes the information described by the function f , and it therefore makes sense to talk about χ as the Kolmogorov Representation of f in univariate form.

Combining these results of Sprecher and Lorentz, the following version of KST (first noted by Sprecher) requires the definition of only one inner function and one outer function, and it provides the basis for subsequent computational approaches to KST.

Theorem 2.2.1. *Let $f \in C(\mathbb{I}^n) : \mathbb{I}^n \rightarrow \mathbb{R}$ be a continuous multivariate function. Fix $\varepsilon \leq \frac{1}{2n}$ a positive real number, and choose $\lambda_1, \dots, \lambda_n$ positive and integrally independent. Set $\delta \geq \sum_{p=1}^n \lambda_p$. Then, there exist univariate continuous functions $\chi : \mathbb{R} \rightarrow \mathbb{R}$ and $\psi : \mathbb{R} \rightarrow \mathbb{R}$, such that for all $x \in \mathbb{I}^n$,*

$$f(x_1, \dots, x_n) = \sum_{q=0}^{2n} \chi \left(\sum_{p=1}^n \lambda_p \psi(x_p + \varepsilon q) + \delta q \right),$$

where the function ψ and choices of coefficients do not depend on the function f in question.

While Sprecher and Lorentz posed a significant reformulation of KST, they were by no means the only mathematicians to do so. Hedberg and Kahane framed KST as a statement concerning mappings between function spaces in the context of the Baire Category Theorem. They reformulated Theorem 1.1.1 as follows (Hedberg [19]):

Theorem 2.2.2 (Hedberg and Kahane KST). *Let $\Phi \subset C(\mathbb{I})$ be the set of continuous non-decreasing functions on \mathbb{I} such that $\forall \psi \in \Phi$,*

$$\psi(0) = 0 \qquad \psi(1) = 1.$$

Let $\lambda_1, \dots, \lambda_n$ be rationally independent positive numbers, such that $\sum_{p=1}^n \lambda_p = 1$. Then,

for quasi-all tuples $(\psi_1, \dots, \psi_{2n+1}) \in \Phi^{2n+1}$, it follows that for any $f \in C(\mathbb{I}^n) : \mathbb{I}^n \rightarrow \mathbb{R}$ multivariate continuous function, there exists some continuous univariate function $\chi : \mathbb{I} \rightarrow \mathbb{R}$ such that

$$f(x_1, \dots, x_n) = \sum_{q=0}^{2n} \chi \left(\sum_{p=1}^n \lambda_p \psi_q(x_p) \right).$$

This reformulation makes use of both Lorentz's and Sprecher's shifting and scaling arguments; here, Hedberg explicitly requires the integral independence of the coefficients $\lambda_1, \dots, \lambda_n$. This reformulation differs from Theorem 1.1.1 by expressing the inner functions as a tuple in the function space Φ^{2n+1} . Various properties of the space Φ make this formulation amenable to tools in functional analysis, and turn KST into a tool in functional analysis in and of itself. These functional analysis arguments are useful: they provide insight into the properties that the inner and outer functions must satisfy as to enable a KST-style representation. This version of this theorem highlights that for any given function f , KST representation is not unique; quasi-all[†] nondecreasing continuous functions can be used as part of a KST-style representation. The choice of these tuples of functions therefore plays an important role in the definition of χ . I revisit this theorem as the basis for a geometric characterization of KST in Chapter 5.

2.2.2 Improving Smoothness

While these reformulations were significant in advancing both the theory and computational practicality of KST, they did not address the difficulty posed by the lack of smoothness of the inner functions invoked in the superposition theorem. Sprecher proved that Kolmogorov's

[†]A property holds for *quasi-all* points in a complete metric space X if it holds for every point in a countable intersection of open dense sets in X .

original inner functions have derivative of zero almost everywhere (Kolmogorov [25], Sprecher [39]). Despite this, Kolmogorov’s original construction yields inner functions that are only Hölder continuous, with exponent $\log_{2n+2}(2)$ (Sprecher [38]). This exponent relates fundamentally to the self-similarity that Kolmogorov used to define the spatial tiling needed to complete his proof (Sprecher [40]); the construction of this spatial tiling is explained in section 3.3. The lack of stronger continuity properties limits the usefulness of Kolmogorov’s functions in computation, as computing values for Hölder continuous functions requires an increasingly finer spatial resolution per unit of function accuracy. As a result, I avoid using Hölder continuous functions in my construction, although this significantly complicates the task of defining such functions algorithmically.

This issue of increasing computational burden per unit accuracy vanishes when the inner functions are constructed to be Lipschitz continuous or smoother. Even before Kolmogorov proved KST, it had been shown that such a representation was impossible if the functions involved were continuously differentiable (Vitushkin [48], Vitushkin [47]), even if the multivariate function in question is analytic (Ostrowski [34]). However, Fridman later showed that it was possible to construct a Lipschitz continuous inner function (Fridman [15]). To do so, Fridman replaced the self-similarities in spatial tiling with a dynamic strategy. Using this approach, Sprecher later extended his reduction of the number of inner functions from $2n^2 + n$ to one using the same shifting and scaling argument that he used in the Hölder case (Sprecher [41]). Because the involved dynamic strategies are comparatively ad-hoc, to the author’s knowledge, there have been no attempts to compute KST representations of functions using Lipschitz inner functions, and the issue of increased computational burden for computational KST remains. It is this task - constructing Lipschitz-continuous inner func-

tions for computational KST - that this thesis addresses; Chapter 4 is devoted to discussing implementations of the strategy that Fridman proposed in [15].

2.3 Computational KST

There have been other attempts to construct computational schemes for KST representation, albeit none known that use Lipschitz inner functions. In developing computational schemes, a distinction emerged between those who maintain the exact representation properties of KST (such as this work), and those who construct approximations of multivariate functions using techniques and structures taken from KST. Techniques for the computation and analysis of KST representation have been advanced by both approaches, and I utilize techniques from each side to compute my Lipschitz KST functions. In the following subsections, I describe attempts for exact KST computation and also several of the approximation schemes that relate to KST.

2.3.1 Exact KST Computation

The first well-publicized algorithm for computing KST directly was developed by Sprecher, after interest in KST was revived due to the theorem’s connection with neural networks; see Section 2.4 (Sprecher [42], Sprecher [43]). These algorithms were motivated by taking smart choices for Kolmogorov’s self-similar refinements, so that the inner function ψ could be defined iteratively by assigning $\psi(x)$ based on each digit in a decimal approximation of a point $x \in \mathbb{I}^n$. These decimal assignments were later corrected so that ψ stayed monotonic at each stage of its iterative construction (Köppen [26]). A closed form of the Köppen’s corrected inner function was later developed; while having a closed form helped to analyze

the computational complexity of ψ , it did not advance the computation of such a function (Braun and Griebel, [7]). This closed-form analysis constitutes one of the few examples where the provided functions meet the criteria of Kolmogorov’s constructed proof, thereby making this analysis a useful blueprint for future work. I discuss how this closed-form analysis must be adapted to be relevant in defining Lipschitz-continuous inner functions in Chapter 3. However, neither closed form nor iterative definitions exist for Lipschitz-continuous inner functions, which I discuss in greater detail in Section 3.3.

2.3.2 Approximations to KST Inner Functions

As these exact KST computations are complicated to define and difficult to execute, various researchers have developed approximations of multivariate functions using KST, thereby forgoing exact representation. These approximations are done by truncating the number of iterations taken to define the inner function, and then constructing a spline that interpolates the obtained function values. These approaches have the benefit of maintaining smoothness while still having a KST-like structure.

For example, Coppejans describes conditions on the coefficients of cubic splines as to enforce monotonicity (Coppejans [8]). With mildly stronger smoothness conditions on the multivariate function f , this spline approximation for the KST functions requires only a spline approximation of the univariate outer function, and therefore Coppejans claims to require fewer point evaluations of f in evaluation, although the author does not provide any computational results to support this claim.

Similarly, Igelnik and Parikh also approximate the inner function using splines, instead using ninth degree splines with an assumption that the inner and outer functions are ap-

proximated by functions that are four times continuously differentiable (Igel'nik and Parikh [21]). The resulting spline approximation is then compared in structure to a single hidden layer feed-forward neural network (see Subsection 2.3.3). With a comparable number of parameters, this spline network achieves an approximation error that is asymptotically better than a standard neural network.

However, in both Coppejans's and Igel'nik and Parikh's spline approximations, convergence is achieved only as the number of knots in the spline approximations increases towards infinity. It is unclear how these approaches are better than simply stopping Sprecher's or Köppen's algorithms after a certain point, and in both cases, there are only theoretical guarantees on smoothness if we assume that our original function f is sufficiently smooth. As a result, this thesis refrains from making a spline approximation and instead seeks to maintain as exact a representation as possible.

2.3.3 Approximations using KST Structure

Even as computational power improved, KST mostly remained relegated to being a theoretical tool due to the complexity of defining effective approximations to the inner and outer functions. However, various other approximation methods, such as neural networks and ridge functions, have developed by using the superposition framework of KST as justification for their approximation properties. Many of these approximations are common, with substantial literature describing their theory and performance. This literature is not discussed in the rest of this thesis, although as ultimately these approximations are the standards to which KST will be compared, they deserve to be mentioned in this section. This comparison will only be fair once the problem of computing outer functions for Kolmogorov representation

is addressed, but this thesis does not focus on this task.

One of the original approximation schemes to use KST as its justification for its approximation capabilities was neural networks. Feed-forward neural networks leverage the superposition structure by replacing the inner function ψ in KST with some activation function, and then compensating by taking extra terms in the sum in order to achieve approximation. In this context, Hecht-Nielsen framed KST as an existence proof for a neural network that approximates any continuous multivariate function. This approach leverages KST to obtain the first proof of the Universal Approximation Theorem for neural networks (Hecht-Nielsen [18]). Kůrkova was later able to refine this construction as to specify how many neurons were required in a single layer hidden feed-forward neural network (using a sigmoid activation function) to achieve arbitrary approximation accuracy (Kůrkova [27]). While these results are limited to neural networks with a single hidden layer, it is possible that this underlying similarity between neural networks and KST can be used to characterize deeper neural networks as well; the ability of deep networks to generalize better than shallower neural networks still lacks a definite explanation. In this context, an algorithmic approach to KST would be a useful tool for analyzing the theoretical representation capabilities of neural networks. However, there is some skepticism as to the applicability of KST to actual learning problems. For a given multivariate function f , KST does not effectively ‘learn’ the outer function χ in KST from real-world data; it requires the evaluation of f at specific points in the construction process (Girosi and Poggio [17]).

Another approximation scheme that uses a similar argument to simplify the functions in KST is that of ridge functions. Ridge functions arise in many physics applications that describe plane waves and other natural phenomena. This approximation relates to KST by

replacing each inner function with a linear kernel, resulting in the approximation

$$f(\mathbf{x}) = \sum_q g_q(A_q \mathbf{y}),$$

where for $\mathbf{x} \in \mathbb{R}^n$, the matrices A_q are of size $m \times n$ with $m \ll n$, and then the functions $g_q : \mathbb{R}^m \rightarrow \mathbb{R}$ are chosen a priori often based on the physical application. If the functions g_q are all chosen to be identical, then the resulting approximation yields ridgelets; if g_q are chosen to be a neuronal activation function, this approximation reduces to a feed-forward neural network. See [35] for a rigorous theoretical analysis. Although some techniques to analyze ridge functions are applicable to KST, ridge functions will not be highlighted in the rest of this thesis, as a comparison of the methods I present to ridge function approximations requires the implementations of outer KST functions as well as inner functions.

2.4 Applications of KST

Many applications take advantage of these approximation schemes that take advantage of KST. However, exact KST approaches to these problems have not become widespread, in part because the lack of Lipschitz-continuous functions for exact representation makes computation a burden using KST directly (Sprecher [44]). At its core, KST says that there are no continuous multivariate functions other than addition and function composition; as a result, encoding a function f by its outer function χ in a KST representation would consist of a form of dimension reduction or function compression. Such a representation scheme, of using the univariate analogues of multivariate functions, would represent a method to break the Curse of Dimensionality, the phenomena in which if the dimension of a problem

increases, the quantity of data required to represent a function needs to grow exponentially so as not to become sparse (Figueiredo [10], Bellman [4]). Towards this end, Figueiredo describes three specific applications where KST could have substantial impact: circuit design, image compression, and statistical regression and estimation (Figueiredo [10]). With each of these applications, algorithms to compute Lipschitz KST representations provide a novel computational approach to solving the underlying multidimensional problems. These applications all require an efficient method to compute the outer KST functions in addition to Lipschitz inner functions, but since this thesis does not discuss how the outer functions are computed, I do not implement any new attempts to solve these applications. In the rest of this section, I describe how KST can be applied to each of the applications mentioned above.

Regarding circuit design, Figueiredo suggests to decompose a current variable with n input variables into $2n + 1$ voltage controllers, one for each outer function, and then directly constructs a voltage system to implement KST (Figueiredo [10]). Later works describe the implementation of KST-style boolean gates for circuit design. Fewer of these KST boolean gates are required, and they consume less power, making them ideal for applications in nanoelectric circuits (Bern and Zawadzki [5]). However, it is unclear whether KST gates are actually able to be constructed in a cost-efficient manner, and they are not currently in practice.

The idea of using KST for multivariate function compression has also been explored in the context of image analysis. Previous work has applied Igelnik and Parikh’s KST spline approximation to image compression. By capturing the lower frequency terms of the outer function approximation, a sample image was compressed by a peak signal-to-noise ratio

(PSNR) of above 30 db; the outer function approximation used approximately half of the pixels in the original image (Leni et al. [28]). The authors suggest pairing this approach with a wavelet decomposition, although to date there is no literature available that does so.

Third, Friedman and Stuetzle used the same structures as ridge functions to develop projection-pursuit regression (Friedman and Stuetzle [16]), a statistical technique that estimates a regression surface using sums of univariate unbiased estimators. Using this regression framework, it is possible to describe which functions defined as linear superpositions are approximable using statistical regression (Diaconis and Shahshahani [12]). This approach is favorable to compute, as it removes from consideration any second-order interaction effects; there are currently many forms of regression that make use of this approach, and this approach is an active topic of research.

2.5 Unresolved and Related Questions

This thesis describes a new method to compute Lipschitz inner functions for KST. However, before KST can be used extensively for application, many questions need to be answered, including how to efficiently compute the outer functions in KST. Even though Kolmogorov proved his seminal theorem in 1957, many open questions still surround Hilbert’s 13th problem. Regarding Hilbert’s 13th problem, it is still unknown if it is possible to represent the solutions of a 7th degree polynomial in terms of algebraic expressions of three variables. As a result, this Hilbert problem is considered only partially solved. The resolution of any of these open problems would provide substantial benefit towards computing KST representations of multivariate functions.

Kolmogorov’s theorem involves a very specific setup, requiring functions f to be continuous and defined only on a compact domain. This was later extended to arbitrary domains in \mathbb{R}^n , including all of \mathbb{R}^n , but requires twice as many functions (Doss [14], Demko [11]), and was also extended to other metric spaces instead of separable Banach spaces (Ostrand [33]). However, there are no algorithms that implement these cases, and it is unclear which components of Kolmogorov’s original construction (and Sprecher’s reduction to only one inner function) apply in this context.

Of similar note, it is unknown how to construct KST-style representations if one was to add additional smoothness constraints to multivariate continuous functions, i.e. to consider functions in $C^k(\mathbb{I}^n)$ in place of $C(\mathbb{I}^n)$. An early result by Vitushkin characterizes whether superpositions are possible when done with smooth functions (Vitushkin [47]). He proved that there is some $f \in C^k(\mathbb{I}^n)$ that cannot be represented by finite superpositions of functions in $C^{k'}(\mathbb{I}^{n'})$ if

$$\frac{n}{k} > \frac{n'}{k'}. \quad (2.3)$$

However, there are no known papers that use this statement for either approximation or exact representation. Similar results could be used to represent functions of high-dimensional spaces with analogous problems in lower-dimensional settings while maintaining some smoothness, rendering unnecessary any concern over currently being only able to compute Hölder or Lipschitz continuous KST representations.

Other problems to consider relate to approximation of discontinuous functions, such as those in the function space $L^p(\mathbb{I}^n)$. Much work has been done towards this goal, both before and after Kolmogorov’s original paper; see Khavinson [23] for more details. An important

result in the algorithmic approximation of discontinuous functions via superpositions is that of Diliberto and Straus, which developed an algorithm to decompose bivariate functions in $L^2(\mathbb{I}^n)$ into sums using univariate functions (Diliberto and Straus [13]). Various improvements have been noted since, such as extension to $L^1(\mathbb{I}^n)$ and more recently to any algebra of functions (Light [29], Asgarova et al. [3]). These advances in approximations and algorithms for representations via composition could be used to create new approaches to KST, potentially superseding the results of this thesis.

Various cases of determining the density of the set of functions given by superpositions contained in common function spaces such as $L^p(\mathbb{I}^n)$ and $C(\mathbb{I}^n)$ are given in Sanguineti's review (Sanguineti [37]). Applications of these algorithms for functions in $L^p(\mathbb{I}^n)$ are important for e.g. tomographic reconstruction, which can be viewed as calculating ridge functions in $L^p(\mathbb{I}^n)$ along certain observed directions along an x-ray (Logan [30], Logan and Shepp [31]). If algorithms for KST become better developed, the work in this thesis could be extended to problems in this domain.

Chapter 3

Sufficient Conditions for Constructing KST Inner Functions

This chapter establishes sufficient conditions for computing the inner functions of the Kolmogorov Superposition Theorem. These conditions constitute the theoretical and algorithmic framework for executing KST. In Section 3.1, I describe Kolmogorov's initial sufficient conditions for a constructive proof of KST. I then restate these conditions, with two substantial changes. The first change, in Section 3.2, reduces the number of unique functions needed for KST. This decreases the burden of KST from defining many functions to needing only one inner function. The second update, in Section 3.3, proves that such a construction can be adapted to produce Lipschitz continuous functions. Ultimately, the algorithm described by this thesis in Chapter 4 and Chapter 5 will satisfy the conditions provided by this chapter.

I restate the Kolmogorov Superposition Theorem (KST), as given in Chapter 1.

Theorem 1.1.1 (KST). *Let $f \in C(\mathbb{I}^n) : \mathbb{I}^n \rightarrow \mathbb{R}$ be a continuous multivariate function. There exist univariate continuous functions $\chi_q : \mathbb{I} \rightarrow \mathbb{R}$ and $\psi_{p,q} : \mathbb{R} \rightarrow \mathbb{R}$, where $p = 1, \dots, n$ and $q = 0, \dots, 2n$, such that for all $x = (x_1, \dots, x_p) \in \mathbb{I}^n$,*

$$f(x_1, \dots, x_n) = \sum_{q=0}^{2n} \chi_q \left(\sum_{p=1}^n \psi_{p,q}(x_p) \right). \quad (1.1)$$

The functions $\psi_{p,q}$ do not depend on the function f (Kolmogorov [25]).

3.1 Kolmogorov's Original Conditions

This section reviews the sufficient conditions that Kolmogorov utilizes in his original proof of KST (Kolmogorov [25]). These conditions are revisited in light of the improvements to KST described in Section 3.2 and Section 3.3. Kolmogorov's proof of KST is completed in Appendix B.

Kolmogorov describes two fundamental conditions for KST in his original paper (Kolmogorov [25]). These conditions describe a suitable spatial partition of \mathbb{I}^n , and they define inner functions $\psi_{p,q}$ in relation to these spatial partitions. Both the spatial partition and the inner functions are defined iteratively; the superscript $k \in \mathbb{N}$ denotes the *refinement level* of this iterative process. The superscript $q \in \{0, \dots, 2n\}$ is an index for the outer sum in Equation 1.1. Kolmogorov's proof of KST requires a spatial partition for each index q , which induces different inner functions $\psi_{p,q}$. The index $p \in \{1, \dots, n\}$ always relates to the p^{th} coordinate direction.

The spatial partition provided by Kolmogorov describes sets $\mathcal{S}^{k,q}$ of n -dimensional squares that nearly partition \mathbb{I}^n . These sets are not complete partitions (hence the modifier 'nearly'), due to gaps between the squares that make the partition an incomplete covering of \mathbb{I}^n . Each set $\mathcal{S}^{k,q}$ is the n -fold Cartesian product of a set of intervals $\mathcal{T}^{k,q}$ that nearly partition the unit interval \mathbb{I} :

$$\mathcal{S}^{k,q} = \prod_{p=1}^n \mathcal{T}^{k,q}. \quad (3.1)$$

An example of such a spatial partition, formed by a Cartesian product, can be seen in Figure 3.1.

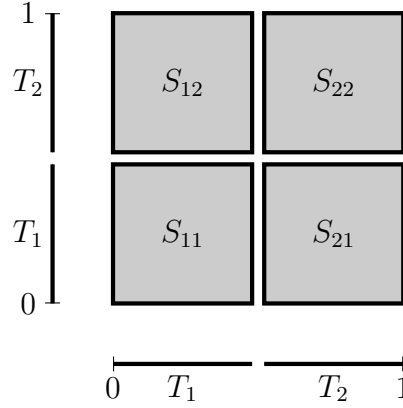


Figure 3.1 : A set $\mathcal{S}^{k,q}$ that nearly partitions \mathbb{I}^2 . Each square $S_{ij} \in \mathcal{S}^{k,q}$ is the Cartesian product of two intervals $S_{ij} = T_i \times T_j$ where $T_i, T_j \in \mathcal{T}^{k,q}$.

As k increases, the size of the intervals in $\mathcal{T}^{k,q}$ uniformly shrinks to 0. The sets $\mathcal{T}^{k,q}$ relate to each other through the index q , which denotes a shift of a ‘prototype set’ \mathcal{T}^k by a factor of q times a fixed amount $\varepsilon \in (0, \frac{1}{2n}]$:

$$\mathcal{T}^{k,q} = \{T + q\varepsilon : T \in \mathcal{T}^k\}. \quad (3.2)$$

Addition here is pointwise addition for each point in a given set:

$$T + q\varepsilon = \{t + q\varepsilon : t \in T\}. \quad (3.3)$$

Kolmogorov places two conditions on the sets $\mathcal{S}^{k,q}$. Kolmogorov’s first condition, the **More Than Half Condition**, is a statement on the system of squares $\mathcal{S}^{k,q}$ for a given refinement level k :

Condition 1 (More Than Half). *For any $x \in I^n$, there are $n + 1$ values for q such that*

$x \in S$ for some square $S \in \mathcal{S}^{k,q}$.

Kolmogorov's second condition, the **Disjoint Image Condition**, relates the inner functions $\psi_{p,q}$ to the system of squares $\mathcal{S}^{k,q}$. Define $\Psi_q(x_1, \dots, x_n) = \sum_{p=1}^n \psi_{p,q}(x_p)$. For fixed refinement level k and shift q , the condition is as follows.

Condition 2 (Disjoint Image). *For any $S_1, S_2 \in \mathcal{S}^{k,q}$,*

$$\Psi^q(S_1) \cap \Psi^q(S_2) = \emptyset.$$

This condition is depicted in Figure 3.2.

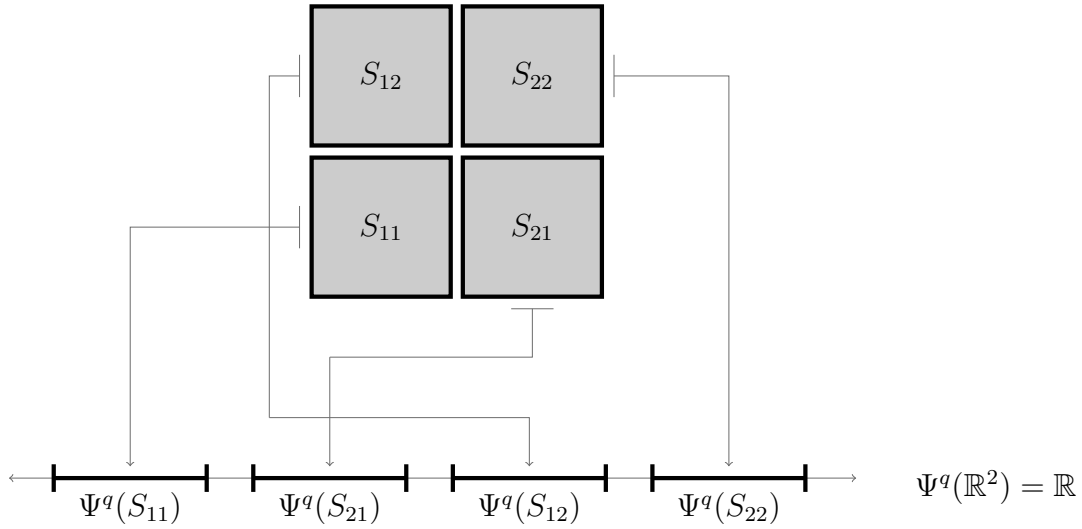


Figure 3.2 : Sketch of set of squares $\mathcal{S}^{k,q}$ that satisfy the Disjoint Image Condition. Observe that the image of each square $S_{ij} \in \mathcal{S}^{k,q}$ under Ψ^q is disjoint, and that the image of each S_{ij} under Ψ^q are connected, since Ψ^q is a continuous function and S_{ij} is a connected domain.

Before proceeding, it should be noted that it is indeed possible to satisfy these conditions simultaneously. Kolmogorov explicitly constructs the set $\mathcal{S}^{k,q}$; this explicit construction

is included in Kolmogorov's proof in Appendix B. While Kolmogorov does not explicitly construct the functions $\psi_{p,q}$, he provides two lemmas that guarantee that such functions exist. These lemmas are explained in more detail in Appendix B. One of these lemmas, which I refer to as the **All But One Condition**, is a sufficient condition for the More Than Half Condition; as this lemma is important for later analysis, it is given here as well as in the appendix.

Condition 3 (All But One). *For any point $x \in \mathbb{I}$, there are $2n$ values of q such that $x \in T$ for some $T \in \mathcal{T}^{k,q}$.*

An example of sets of intervals that satisfy this condition can be seen in Figure 3.3.

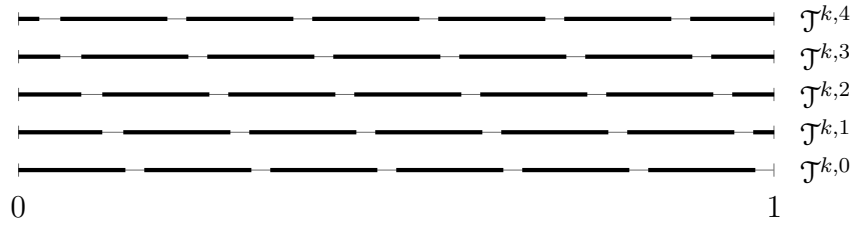


Figure 3.3 : Example of intervals $\mathcal{T}^{k,q}$ that satisfy the All But One Condition. Note that in this construction, each line of intervals is a translation of the line below it; this translation trick is a convenient method to make sure that the All But One Condition is satisfied, although it is not necessary.

For fixed q , Kolmogorov's sets of intervals $\mathcal{T}^{k,q}$ and $\mathcal{T}^{k+1,q}$ are related via self-similarity. Kolmogorov mentions that the interval $T_{i+1}^{k,q}$ is obtained from $T_i^{k,q}$ via a fixed translation,

$$T_{i+1}^{k,q} = T_i^{k,q} + \frac{1}{(9n)^k}.$$

The construction of $\mathcal{T}^{k+1,q}$ from $\mathcal{T}^{k,q}$ can be viewed as refining the intervals at level k via

scaling each $T_i^{k,q}$ down by a factor of $\frac{1}{9n}$ and translating this mini-interval $9n$ times; this process is seen in Figure 3.4. These aspects of self-similarity enable the construction of Hölder continuous inner functions, which will be discussed in Section 3.3.

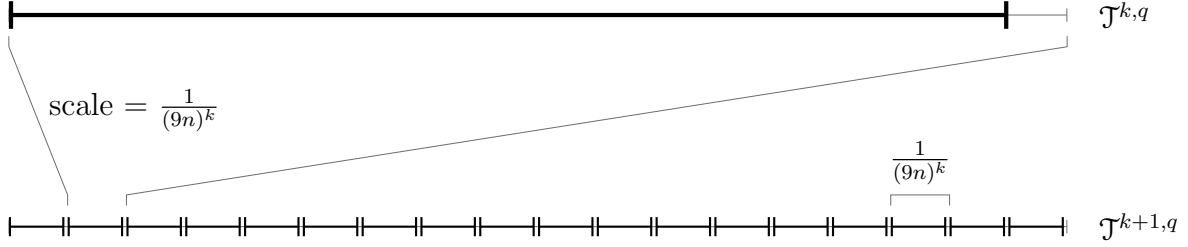


Figure 3.4 : Kolmogorov's self-similar refinement for $n = 2$. Note that the fundamental unit of the self-scaling includes both the interval and the gap to the right of the interval.

Kolmogorov uses the lemmas included in Appendix B to show inner functions are strictly monotonic increasing. This provides the following condition on $\psi^{k,q}$ and ψ^k :

Condition 4 (Monotonicity). *The functions $\psi^{k,q}$ are monotonic increasing. The functions ψ^q are strictly monotonic increasing.*

3.2 Sprecher's Reduction

From a computational viewpoint, one of the difficulties in Kolmogorov's proof of KST is the necessity to define $2n^2 + n$ individual functions $\psi_{p,q}$. Sprecher observed that by shifting and scaling the set of intervals $\mathcal{J}^{k,q}$, the number of inner functions could be reduced from $2n^2 + n$ to one (Sprecher [38]). This reduction and the accompanying proof that KST is still viable even if only one inner function is used provide additional stipulations on Kolmogorov's conditions above.

At the core of the reduction from $2n^2 + n$ inner functions $\psi_{p,q}$ to one inner function ψ is a scaling-and-shifting argument: in this reformulation, Kolmogorov's functions $\psi_{p,q}$ become

$$\psi_{p,q}(x) = \lambda_p \psi(x + q\varepsilon). \quad (3.4)$$

The scaling factors $\lambda_1, \dots, \lambda_n$ are required to be integrally independent.

Definition (Integrally Independent). *The scalars $\lambda_1, \dots, \lambda_n$ are **integrally independent** (alternatively, rationally independent), if for any rational numbers $x_1, \dots, x_n \in \mathbb{Q}$,*

$$\text{If } \sum_{i=1}^n \lambda_i x_i = 0, \quad \text{then } x_i = 0 \text{ for } i = 1, \dots, n. \quad (3.5)$$

The integral independence of the scaling factors $\lambda_1, \dots, \lambda_n$ enforces that the functions $\Psi_q(x_1, \dots, x_n) = \sum_{p=1}^n \lambda_p \psi(x_p + q\varepsilon)$ satisfy the Disjoint Image Condition.

The shift by a factor of $q\varepsilon$ in the argument of the function ψ in Equation 3.4 mirrors the translation of the prototype set of intervals \mathcal{T}^k to the translated sets $\mathcal{T}^{k,q}$. As a result, the domain of ψ is larger than that of $\psi_{p,q}$ as to accommodate the translations by $q\varepsilon$:

$$\begin{aligned} \psi_{p,q} : [0, 1] &\rightarrow \mathbb{R} && \text{from Kolmogorov} \\ \psi : [0, 2] &\rightarrow \mathbb{R} && \text{from Sprecher.} \end{aligned} \quad (3.6)$$

In light of these changes, the Disjoint Image Condition becomes a statement relating the function ψ and the set of intervals \mathcal{T}^k at each refinement level k , given $\lambda_1, \dots, \lambda_n$ integrally independent positive real numbers, and $\varepsilon \in (0, \frac{1}{2n}]$. The updated **Disjoint Image Condition** is as follows:

Condition 5 (Disjoint Image). *For all $q \in \{0, \dots, 2n\}$, it holds that for any $S_1, S_2 \in \mathcal{S}^k$, for $x^{(1)} \in S_1$ and $x^{(2)} \in S_2$,*

$$\sum_{p=1}^n \lambda_p \psi(x_p^{(1)} + q\varepsilon) \cap \sum_{p=1}^n \lambda_p \psi(x_p^{(2)} + q\varepsilon) = \emptyset.$$

Sprecher's precise reformulation of KST is given in Appendix C. As the proof of Sprecher's refinement requires a non-trivial introduction of notation and bookkeeping, the proof is not repeated there. However, three primary objectives within Sprecher's proof that relate to the sufficient conditions for constructive KST are discussed in Appendix C. The complete proof can be found in [38].

Sprecher's proof is worthy of discussion because it gives some intuition for the construction of sets of intervals \mathcal{T}^k and the inner function ψ that satisfy the All But One Condition and the Disjoint Image Condition. In his verification that his inner function ψ satisfies the Disjoint Image Condition, Sprecher constructs a set of disjoint intervals $\square^k = \{\Delta_i^k\}$, where for square $S_i^k \in \mathcal{S}^k$, the interval $\Delta_i^k \in \square^k$ contains the image of $\Psi_q(S_i^k)$. The Disjoint Image Condition is met if the following, more conservative condition is satisfied at each refinement level k ; this condition is illustrated in Figure 3.5.

Condition 6 (Conservative Disjoint Image). *For any intervals $\Delta_i^k, \Delta_{i'}^k \in \square^k$, where $\Psi^q(S_i^k) \subseteq \Delta_i^k$ and $\Psi^q(S_{i'}^k) \subseteq \Delta_{i'}^k$,*

$$\Delta_i^k \cap \Delta_{i'}^k = \emptyset.$$

Sprecher's method to enforce this condition is discussed in Appendix C. I revisit this condition in greater detail during Chapter 4. Sprecher uses the sets \square^k to analyze the

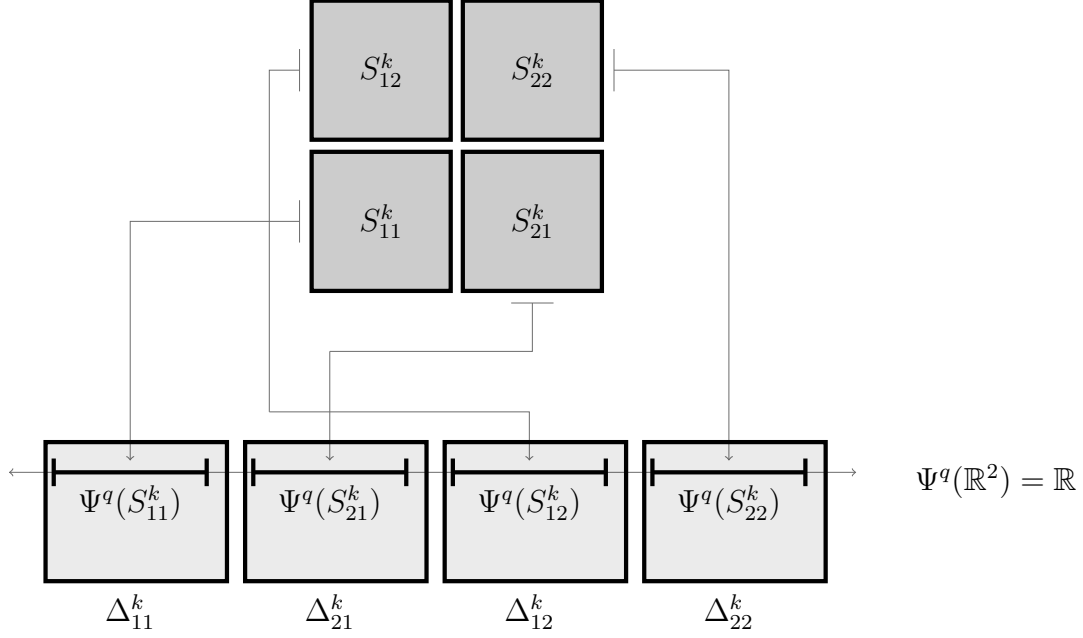


Figure 3.5 : Sketch of the relationship between the sets \mathcal{S}^k and \square^k under the Conservative Disjoint Image Condition. Compare to Figure 3.2; with the Cconservative Disjoint Image Condition, the image intervals Δ^k_{ij} envelope the intervals $\Psi^q(S^k_{ij})$. Since more space needs to be allocated for the intervals Δ^k_{ij} than $\Psi^q(S^k_{ij})$, they deemed more *conservative*, hence the name of the condition.

smoothness properties of his function ψ . This is the subject of the next section.

3.3 Conditions for Improved Smoothness of the Inner Function

During his reformulation of KST in [38], Sprecher proves that his inner function ψ is Hölder continuous.

Definition (Hölder). *Let X and Y be normed metric spaces. A function $\psi : X \rightarrow Y$ is **Hölder continuous with exponent** α if there is some constant c such that for any two points $x_1, x_2 \in X$,*

$$\|\psi(x_1) - \psi(x_2)\|_Y \leq c \|x_1 - x_2\|_X^\alpha.$$

The regular distribution of intervals at refinement level k , and the uniform size of these intervals, guarantee the Hölder continuity of the function ψ . Recall that for fixed q , Kolmogorov's sets of intervals $\mathcal{T}^{k,q}$ and $\mathcal{T}^{k+1,q}$ are related via self-similarity, scaling by a factor of $\frac{1}{9n}$ and then creating $9n$ translated copies of the shrunken intervals. Sprecher's proof replaces the factor of $9n$ with the integer γ . This self-similarity enables the construction of Hölder continuous inner functions. For verification that Sprecher's function ψ is Hölder continuous, see Section C.3 or Sprecher's original paper ([38]). As Kolmogorov's construction is similar, his inner functions $\psi_{p,q}$ presumably can be chosen to be Hölder continuous. Yet, Kolmogorov's inner functions are not exactly specified, so it may be possible to construct inner functions using Kolmogorov's spatial decomposition that fail to be Hölder continuous; see Appendix B for more details. It is thought that any near partition of \mathbb{I} using a fixed self-similar refinement scale will (at best) lead to a Hölder continuous inner function (Sprecher [44]); such a relationship seems to have been accepted in the early literature on KST despite not being published anywhere to the author's knowledge.

The reasoning for this thought, at least in the case of Sprecher's construction, is as follows. Sprecher defines his inner function ψ as the uniform limit of functions ψ^k . A sketch of the relationship between ψ^k , ψ , and \mathcal{T}^k is given in Figure 3.6. At each refinement level k , these functions are constant on intervals in \mathcal{T}^k and interpolate linearly between the gaps between the intervals. In the limit as $k \rightarrow \infty$, the functions ψ changes only minimally at each refinement level: once a value has been assigned to the function ψ^k on an interval $T \in \mathcal{T}^k$, that value remains fixed at the left endpoint of T for all future functions $\psi^{k'}$ during refinement levels $k' > k$. As a result, the length of the gaps between intervals shrinks by a factor of γ at each successive refinement level, but the function must jump between fixed

function values that are assigned at the left endpoints, and this jump does not contract by a full factor of γ at each refinement level. Therefore, the slope of ψ^k on the gaps between intervals increases by a factor proportional to γ^α for some power $\alpha > 1$, resulting in a Hölder continuous function. A rigorous version of this argument is presented in Section C.3.

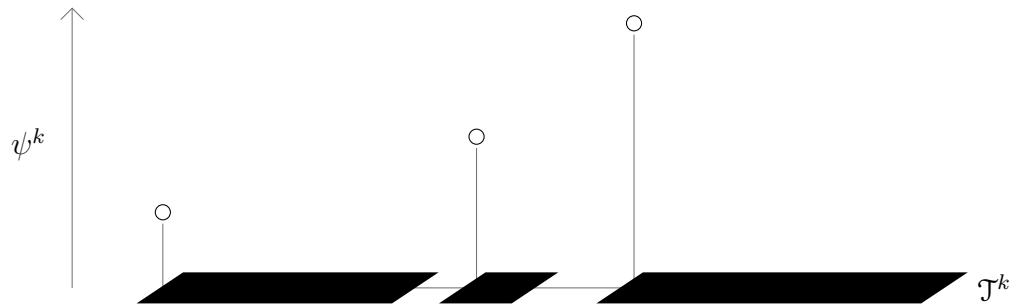
However, Hölder continuous functions are of limited usefulness in computation. Computing values for functions that are only Hölder continuous requires an increasingly finer spatial resolution per unit of function accuracy. For these functions, the ratio between evaluation accuracy and storage complexity grows exponentially. As a result, this thesis constructs inner functions that are smoother than Hölder continuous: the next smoothness class is that of Lipschitz continuous functions.

Definition (Lipschitz). *Let X and Y be normed metric spaces. A function $\psi : X \rightarrow Y$ is **Lipschitz continuous** if there is some constant c such that for any two points $x_1, x_2 \in X$,*

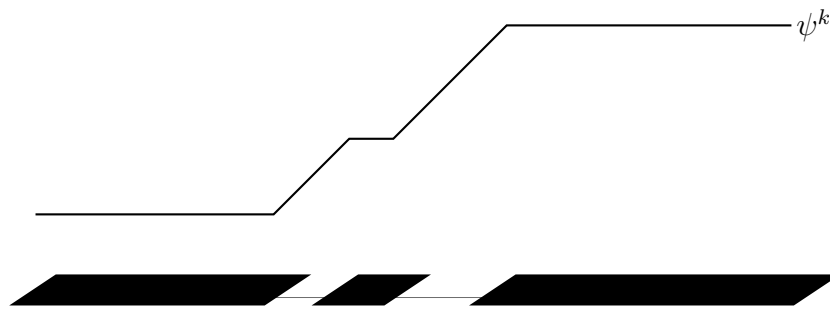
$$\|\psi(x_1) - \psi(x_2)\|_Y \leq c \|x_1 - x_2\|_X.$$

The first (and to date, only) result that claims to constructively prove the existence of Lipschitz continuous functions that suffice for KST is that of Fridman ([15]). Fridman’s Lipschitz result was later combined with Sprecher’s reduction in the number of inner functions, resulting in the following restatement of KST (Sprecher [41]).

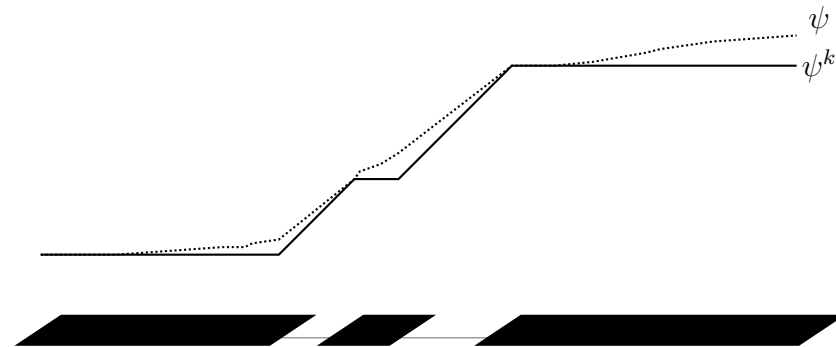
Theorem 3.3.1 (KST-Fridman). *Let $f \in C(\mathbb{I}^n) : \mathbb{I}^n \rightarrow \mathbb{R}$ be a continuous multivariate function. Let $\lambda_1, \dots, \lambda_n$ be integrally independent positive real numbers. Fix $\widehat{\varepsilon} \in (0, \frac{1}{2n}]$. There exist a rational number $\varepsilon \leq \widehat{\varepsilon}$ and univariate continuous functions $\chi_q : \mathbb{I} \rightarrow \mathbb{R}$ and*



(a) Fixing values for ψ^k on left endpoints.



(b) Function ψ^k constant on intervals and linear on gaps between intervals.



(c) Function ψ changes only minimally from ψ^k while on intervals in \mathcal{T}^k .

Figure 3.6 : Defining ψ^k during refinement level k . Note that for each $k \in \mathbb{N}$, the function ψ^k is constant on intervals and linear on gaps, where the values at the left endpoints of the intervals are fixed and held constant for all $k' > k$. This process yields a final function ψ that is differentiable almost everywhere, with derivative 0 almost everywhere, and not differentiable on a dense set of points.

$\psi : [0, 2] \rightarrow \mathbb{R}$, where $q = 0, \dots, 2n$, such that for all $x = (x_1, \dots, x_p) \in \mathbb{I}^n$,

$$f(x_1, \dots, x_n) = \sum_{q=0}^{2n} \chi_q \left(\sum_{p=1}^n \lambda_p \psi(x_p + q\varepsilon) \right). \quad (3.7)$$

The function ψ , scalars $\lambda_1, \dots, \lambda_n$, and constant ε do not depend on the function f in question. Additionally, the function ψ can be constructed to be Lipschitz continuous with constant 1 (Sprecher [41]).

Fridman's construction of the set of intervals \mathcal{T}^k and of squares \mathcal{S}^k is more complicated than either Kolmogorov's or Sprecher's sets. Both Kolmogorov's and Sprecher's squares $\mathcal{S}^{k,q}$ do not depend on the inner functions $\psi_{p,q}$. In a sense, the squares $\mathcal{S}^{k,q}$ and functions $\psi_{p,q}$ are independent: the squares can be defined at *every* refinement level before any of the function values have been assigned to the inner functions. This is a convenient feature of algorithms that construct Hölder continuous inner functions, but this independence does not occur during the construction of Lipschitz continuous inner functions. To construct an inner function ψ that is smoother than Hölder continuous, i.e. Lipschitz continuous, it is necessary to abandon the fixed geometric refinement in going from \mathcal{T}^k to \mathcal{T}^{k+1} .

At refinement level k Fridman's proof defines the set \mathcal{T}^k dynamically as to ensure that the All But One Condition is met, while keeping the slope of functions ψ^k bounded. In his proof, Fridman breaks the largest intervals in \mathcal{T}^k roughly in half by removing a segment that contains the interval's midpoint; this is illustrated in Figure 3.7. This breaking may cause the All But One Condition to be violated at some point in the removed segment; this case is illustrated in Figure 3.8a. To counter this, Fridman adds (before refinement) to the set \mathcal{T}^k an extra small interval that contains this problematic point, so that before breaking, the

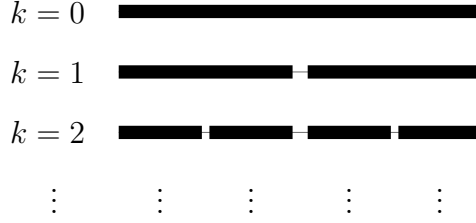


Figure 3.7 : Breaking intervals roughly in half. This breaking process is crucial to show that the functions ψ^k converge.

point in question is contained in an interval $T^q \in \mathcal{T}^{k,q}$ for every shift index q . Figure 3.8b depicts an example of this process. With this addition, removing a sufficiently small segment from any of the largest intervals will not violate the All But One condition. This approach is what I call the Fridman Strategy, which I describe in Chapter 4. For more precise details, see [15].

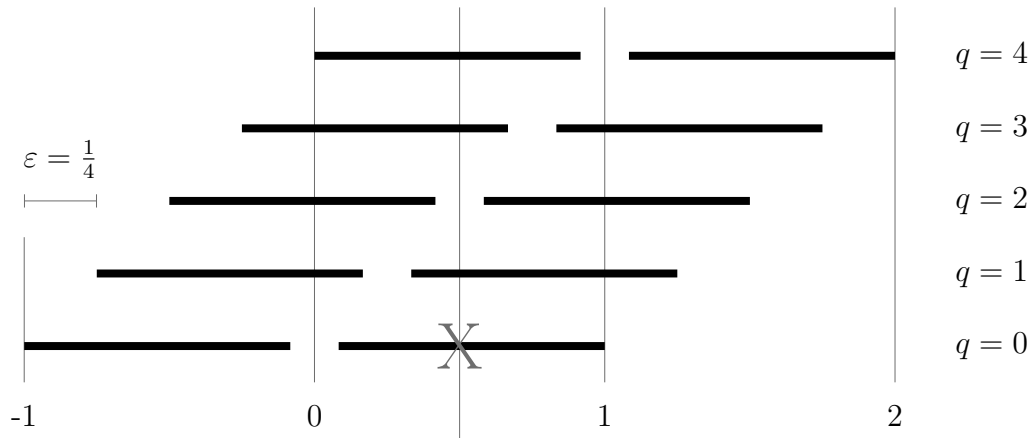
However, adding a small interval will cause the slope of ψ^k to increase on the gaps surrounding the inserted interval, so this addition cannot be too large. This scenario is reflected in Figure 3.9. The growth of the slope from these additional small segments is controlled by the **Bounded Slope Condition** at refinement level k on the functions ψ^k :

Condition 7 (Bounded Slope). *For any two points $x_1, x_2 \in [0, 2]$,*

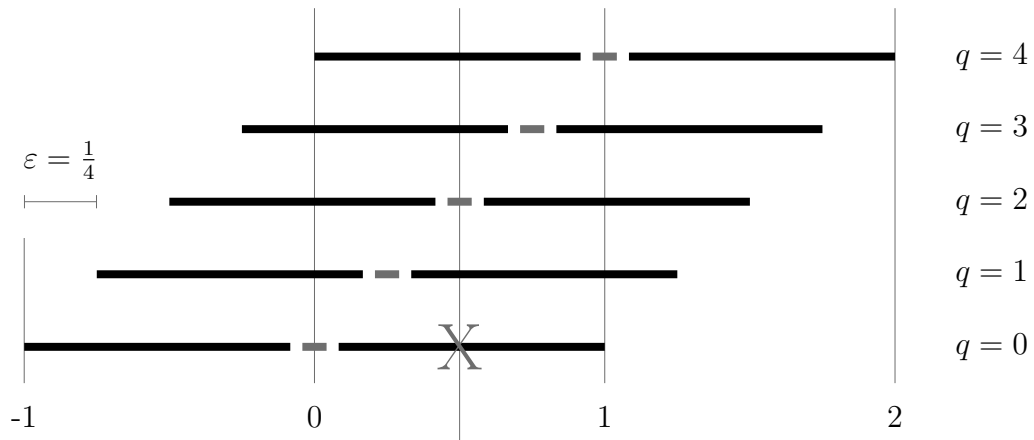
$$|\psi^k(x_1) - \psi^k(x_2)| \leq (1 - 2^{-k})|x_1 - x_2|.$$

Fridman's proof is given in its entirety in [15]. The dynamic construction of \mathcal{T}^k is described in [15]. As Fridman does not refine his intervals \mathcal{T}^k geometrically, he explicitly states a **Refinement Condition** in [15]:

Condition 8 (Refinement). *With successive refinement, the length of intervals $T \in \mathcal{T}^k$ goes*



(a) Any break at the point marked by the large X would cause the point X to not belong to some interval for both $q = 0$ and $q = 2$.



(b) With the additional interval, the point X is now included for $q = 2$. Once broken, X will not belong to an interval in the set $q = 0$, but this point is possessed by some interval in all the other sets.

Figure 3.8 : Scenario in which it is necessary to add small segments to maintain the All But One Condition. As adding these small segments increases the slope of the function ψ^{k+1} (see Figure 3.9), it is crucial to choose the segments sufficiently small.

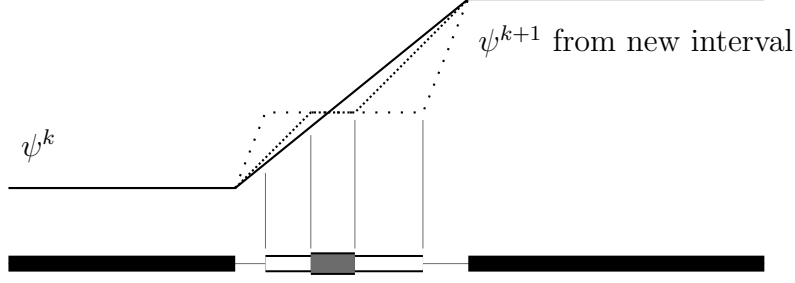


Figure 3.9 : Sketch of how size of plugs changes the slope of ψ^k . Note that larger plugs correspond to a greater slope in the new function ψ^{k+1} .

to zero uniformly, i.e.

$$\lim_{k \rightarrow \infty} \max_{T \in \mathcal{T}^k} |T| = 0.$$

Fridman's strategy is presented in depth in Chapter 4, where two different algorithmic approaches are given to meet the criteria from Fridman's paper. See [15] for more details.

3.4 A Summary of Conditions

Before proceeding in Chapter 4 to discuss an algorithm to compute KST inner functions, I review the sufficient conditions posed in this chapter.

3.4.1 Conditions on Spatial Decompositions

Conditions on the intervals \mathcal{T}^k and squares \mathcal{S}^k at refinement level k are the Refinement Condition and the More Than Half Condition. By the Pidgeon Hole Principle, it is sufficient to consider the All But One Condition instead of the More Than Half Condition.

Condition 8 (Refinement). *With successive refinement, the length of intervals $T \in \mathcal{T}^k$ goes to zero uniformly, i.e.*

$$\lim_{k \rightarrow \infty} \max_{T \in \mathcal{T}^k} |T| = 0.$$

Condition 1 (More Than Half). *For any $x \in I^n$, there are $n + 1$ values for q such that $x \in S$ for some square $S \in \mathcal{S}^{k,q}$.*

Condition 3 (All But One). *For any point $x \in \mathbb{I}$, there are $2n$ values of q such that $x \in T$ for some $T \in \mathcal{T}^{k,q}$.*

3.4.2 Conditions on Function Values

Conditions on the functions ψ , ψ^k , Ψ^q , and $\Psi^{k,q}$ at refinement level k for shift index q are the Bounded Slope Condition, the Monotonicity Condition, and the Disjoint Image Condition, where the Bounded Slope condition ensures that the resulting functions are Lipschitz continuous and the Monotonicity Condition enforces monotonicity. For intervals Δ_i^k that contain the image interval $\Psi^q(S_i^k)$, it is sufficient to consider the Conservative Disjoint Image Condition in place of the Disjoint Image Condition.

Condition 7 (Bounded Slope). *For any two points $x_1, x_2 \in [0, 2]$,*

$$|\psi^k(x_1) - \psi^k(x_2)| \leq (1 - 2^{-k})|x_1 - x_2|.$$

Condition 4 (Monotonicity). *The functions $\psi^{k,q}$ are monotonic increasing. The functions ψ^q are strictly monotonic increasing.*

Condition 2 (Disjoint Image). *For any $S_1, S_2 \in \mathcal{S}^{k,q}$,*

$$\Psi^q(S_1) \cap \Psi^q(S_2) = \emptyset.$$

Condition 6 (Conservative Disjoint Image). *For any intervals $\Delta_{\mathbf{i}}^k, \Delta_{\mathbf{i}'}^k \in \square^k$, where $\Psi^q(S_{\mathbf{i}}^k) \subseteq \Delta_{\mathbf{i}}^k$ and $\Psi^q(S_{\mathbf{i}'}^k) \subseteq \Delta_{\mathbf{i}'}^k$,*

$$\Delta_{\mathbf{i}}^k \cap \Delta_{\mathbf{i}'}^k = \emptyset.$$

Chapter 4

The Fridman Strategy

This chapter analyzes how Fridman constructs a Lipschitz inner function [15], and ultimately concludes that Fridman’s approach is unlikely to succeed in creating a Lipschitz inner function that meets all of the conditions listed in Section 3.4. Section 4.1 reviews the conditions that Fridman describes during his construction process. In Section 4.2 I pose an algorithm that implements Fridman’s approach, and I prove that this algorithm satisfies Fridman’s conditions but fails to satisfy the Disjoint Image Condition. In Section 4.3 I outline an approach that is similar to Fridman’s paper that directly enforces the Conservative Disjoint Image Condition, and I show that this approach sacrifices meeting the All But One Condition to do so. I therefore conclude that Fridman’s approach is not currently a viable method to construct a Lipschitz inner function. As a result, I return in Chapter 5 to the theory underlying KST to motivate a different approach to computing a Lipschitz inner function.

4.1 Characterization of the Fridman Strategy

Fridman’s approach to construct a Lipschitz continuous inner function was previously introduced in Chapter 3. Fridman decomposes the unit interval \mathbb{I} adaptively and iteratively, creating a set of intervals \mathcal{T}^k that meets the conditions listed previously. These conditions are restated here; they are not unique to Fridman’s approach.

Condition 8 (Refinement). *With successive refinement, the length of intervals $T \in \mathcal{T}^k$ goes*

to zero uniformly, i.e.

$$\lim_{k \rightarrow \infty} \max_{T \in \mathcal{T}^k} |T| = 0.$$

Condition 3 (All But One). *For any point $x \in \mathbb{I}$, there are $2n$ values of q such that $x \in T$ for some $T \in \mathcal{T}^{k,q}$.*

Fridman defines the functions ψ^k while generating \mathcal{T}^k :

Condition 7 (Bounded Slope). *For any two points $x_1, x_2 \in [0, 2]$,*

$$|\psi^k(x_1) - \psi^k(x_2)| \leq (1 - 2^{-k})|x_1 - x_2|.$$

Condition 4 (Monotonicity). *The functions $\psi^{k,q}$ are monotonic increasing. The functions ψ^q are strictly monotonic increasing.*

Condition 2 (Disjoint Image). *For any $S_1, S_2 \in \mathcal{S}^{k,q}$,*

$$\Psi^q(S_1) \cap \Psi^q(S_2) = \emptyset.$$

Fridman's overall strategy is characterized by iteratively breaking large intervals at successive levels of refinement in order to enforce the Refinement Condition. Each iteration, corresponding to a refinement level $k \in \mathbb{N}$, proceeds as follows.

First, intervals are selected to be broken. Create these breaks by removing a *gap* that includes the interval's midpoint. However, if there is some shift index \hat{q} such that the midpoint is not contained in any interval belonging to $\mathcal{T}^{k,\hat{q}}$, creating a gap of any size causes the midpoint to no longer be covered by at least $2n$ of the $2n + 1$ sets $\mathcal{T}^{k,q}$, thereby violating

the All But One Condition. In this case, a small *plug* interval is added to $\mathcal{T}^{k,\hat{q}}$. The width of these plugs are determined by solving a block-diagonal linear system. Once these plugs are added, every midpoint is contained in some interval (possibly a newly inserted plug) in all $2n + 1$ shifted copies of $\mathcal{T}^{k,q}$. Therefore, if the gaps are chosen to be small enough, their removal from the large intervals will not cause the All But One Condition to be violated. Additionally, if the gaps are small enough, breaking one interval has no effect on other intervals at the same refinement level.

The intervals \mathcal{T}^k and functions ψ^k are constructed iteratively over refinement level $k \in \mathbb{N}$. After initialization, the Fridman Strategy can be summarized as consisting of the following steps at each refinement level $k \in \mathbb{N}$:

1. **Find Stage:** Find the intervals $T \in \mathcal{T}^k$ that need to be broken.
2. **Plug Stage:** Determine whether the break points $b \in T$ lies in all the shifted families $\mathcal{T}^{k,q}$, or all but one. If there is some shift \hat{q} such that for all $T \in \mathcal{T}^{k,\hat{q}}$, the shifted point $b + \hat{q}\varepsilon \notin T$, then add the necessary small *plug* segment to into the set $\mathcal{T}^{k,q}$.
3. **Break Stage:** Remove from T some small segment that includes the break point b , thereby breaking T in half.

Choices made regarding the execution of these steps can cause substantial differences in the resulting function. In the following sections, I describe algorithms that specify some of the choices made for each of these stages. In Section 4.2, I describe an algorithm that specifies choices for each of these stages in order to satisfy Fridman's version of the Disjoint Image Condition. This algorithm concretizes the steps Fridman makes in his original paper.

In contrast, the approach taken in Section 4.3 enforces the Conservative Disjoint Image Condition instead. The ramifications of the distinctions between these two choices are discussed at the end of this chapter.

4.2 The Fridman Strategy with Fridman's Disjoint Image Condition

Before I delve into how Fridman executed each of the stages in the Fridman Strategy, I describe the changes to the sufficient conditions that Fridman made in his original paper [15]. Fridman makes a subtle alteration to the Disjoint Image condition, compared to the version described previously. Fridman's original proof did not make use of Sprecher's refinement to require only one inner function, so the notation will change to include the q shift indices where appropriate. The following discussion is valid when using Sprecher's reduction, with the appropriate shift in notation e.g. from $\mathcal{T}^{k,q}$ to \mathcal{T}^k (Sprecher [41]).

Let $\psi^{k,p,q} : \mathbb{I} \rightarrow \mathbb{R}$ be a continuous, monotonic nondecreasing function, constant on intervals in the set $\mathcal{T}^{k,q}$ and linear in between; this function is constructed in a manner similar to Sprecher's functions ψ^k but without making use of Sprecher's reduction in the number of needed inner functions. Fridman's variant of the Disjoint Image Condition restricts the image of the functions $\Psi^{k,q} = \sum_{p=1}^n \psi^{k,p,q}$ on the set of squares $\mathcal{S}^{k,q}$ at refinement level k for any fixed shift index q . From its original statement, the condition is later relaxed to ultimately read as follows [15]:

Condition 9 (Fridman's Disjoint Image). *There is some $\varepsilon_k > 0$ such that for any squares $S_1, S_2 \in \mathcal{S}^{k,q}$, the distance between the intervals $\Psi^{k,q}(S_1)$ and $\Psi^{k,q}(S_2)$ is larger than ε_k .*

During his sketch of the three stages of the Fridman Strategy, Fridman chooses to enforce this relaxed condition instead of Kolmogorov's original Disjoint Image Condition. The algorithm I present in Subsection 4.2.1 constructs sets of intervals \mathcal{T}^k and functions ψ^k that satisfy Fridman's relaxed condition along with the other conditions repeated in Section 4.1.

4.2.1 Posing an Algorithm

Algorithm 1 outlines an implementation of the Fridman Strategy that satisfies the conditions in Section 4.1, replacing the Disjoint Image Condition with Fridman's variant. This algorithm is defined to be robust to choices made during its computation, such as which intervals to process first at a given refinement level. If this construction proceeds one interval at a time, it is possible to create plugs or gaps in $\mathcal{T}^{k,q}$ that intersect with other plug or gaps for a different shift index $q' \neq q$ of the same refinement level, thereby altering the function values assigned during this stage of the construction. This concern is addressed by solving a linear system that describes the size of the plugs. As a result, the algorithm is robust to a permutation of the order in which it processes the intervals that need to be split during refinement level k .

Algorithm 1 presents pseudocode of this algorithm. Begin with $\psi^0 \equiv 0$ and $\mathcal{T}^0 = \{[-1, 1]\}$. Fix $\theta \in (0, 1)$, and fix $\varepsilon = \frac{1}{2n}$. As outlined in Section 4.1, each refinement level $k \in \mathbb{N}$ consists of three primary stages:

1. Find Stage
2. Plug Stage
3. Break Stage

Each stage of this implementation is described in greater detail below; this algorithm is

analyzed in Section 4.2.5. Each refinement level k is considered subsequently. As a result the k superscript is not emphasized in the following notation, and it has been removed when the refinement level is apparent from the provided context*. Omitting this superscript declutters some of the notation that will be introduced later.

4.2.2 Find Stage

Finding which holes need plugs is straightforward. Take \mathcal{T}^k and ψ^k as defined at refinement level $k \in \mathbb{N}$.

Denote the set of intervals to break as $\mathcal{B} \subset \mathcal{T}^k$, given by

$$\mathcal{B} = \{T \in \mathcal{T}^k : |T| \geq \theta^k\}.$$

For each $T \in \mathcal{B}$, denote its midpoint by $p = p(T)$. Let the set of break points be

$$\mathcal{P} = \{p(T) : T \in \mathcal{B}\}.$$

I define several sets that relate the interval T and its midpoint p to possible shifting indices $q \in \{0, \dots, 2n\}$. These sets will be useful for the rest of this section. The dependence of these sets on T is dropped when it is clear which interval is being considered.

$$\begin{aligned} \Upsilon(T) &= \{T' \in \mathcal{T}^k : \exists q \in \{-2n, \dots, 2n\} \text{ such that } p(T) + q\varepsilon \in T'\} \\ \mathcal{Q}(T) &= \{q \in \{-2n, \dots, 2n\} : \exists T' \in \mathcal{T}^k \text{ such that } p(T) + q\varepsilon \in T'\} \\ \mathcal{Q}^c(T) &= \{q \in \{-2n, \dots, 2n\} : \forall T' \in \mathcal{T}^k, p(T) + q\varepsilon \notin T' \text{ yet } p(T) + q\varepsilon \in [-1, 1]\}. \end{aligned} \tag{4.1}$$

*The exceptions to this rule for notation are the sets of intervals \mathcal{T}^k and the functions ψ^k .

Algorithm 1 Fridman's Lipschitz Construction

```

procedure LIP( $n$ ) ▷  $n$  spatial dimension

   $\mathcal{T} \leftarrow \{[-1, 1]\}$  ▷ Initialization
   $\psi^0 \equiv 0$ 
   $k = 0$ 

  while  $k < \infty$  do

     $\mathcal{B} \leftarrow \{T \in \mathcal{T}^k : |T| \geq \theta^k\}$ 

     $\mathcal{H} \leftarrow \emptyset$  ▷ Find Stage
    for  $T \in \mathcal{B}$  do
       $p \leftarrow p(T)$ 
       $\Upsilon \leftarrow \Upsilon(T)$ 
       $\mathcal{Q} \leftarrow \mathcal{Q}(T)$ 
       $\mathcal{Q}^c \leftarrow \mathcal{Q}^c(T)$ 
      if  $|\mathcal{Q}| < 2n + 1$  then
        for all  $\hat{q} \in \mathcal{Q}^c$  do
           $\mathcal{H} \leftarrow \mathcal{H} \cup \{H(T, \hat{q})\}$ 

     $\Pi \leftarrow \emptyset$  ▷ Plug Stage
    for all  $H \in \mathcal{H}$  do
       $\nu \leftarrow \nu(H)$ 
       $x \leftarrow C^{-1}z$ 
      for  $i \in 0, \dots, \nu - 1$  do
         $\pi_i \leftarrow [x_{2i}, x_{2i+1}]$ 
         $\psi^{k+1}(\pi_i) \leftarrow \psi^k(p_i + \hat{q}_i \varepsilon)$ 
       $\Pi \leftarrow \Pi \cup \{\pi_i\}_{i=0, \dots, \nu-1}$ 
     $\mathcal{T} \leftarrow \mathcal{T} \cup \Pi$ 

    for  $T \in \mathcal{B}$  do ▷ Break Stage
       $a, b \leftarrow T = [a, b]$ 
       $\rho \leftarrow \min\{\alpha\rho_+, \alpha\rho_-, \beta\delta_+, \beta\delta_-\}$ 

       $T_- \leftarrow [a, p(T) - \rho]$ 
       $T_+ \leftarrow [p(T) + \rho, b]$ 
       $\mathcal{T} \leftarrow \mathcal{T} \setminus \{T\} \cup \{T_-, T_+\}$ 
       $T_N \leftarrow \text{next interval after } T$ 
       $m \leftarrow \max\{1 - \frac{1}{2^k}, \frac{1}{2}(\text{slope between interval } T \text{ and } T_N)\}$ 
       $\psi^{k+1}(T_-) \leftarrow \psi^k(T)$ 
       $\psi^{k+1}(T_+) \leftarrow \psi^k(T) + 2\rho m$ 

     $\psi^{k+1} \leftarrow \text{linearly connect values of } \psi^{k+1}$ 

```

The set $\Upsilon = \Upsilon(T) \subseteq \mathcal{T}^k$ marks the intervals in the shifted set $\mathcal{T}^{k,q}$ that contain the shifted midpoint $p + q\varepsilon$, where the shift index $q \in \{-2n, \dots, 2n\}$. The set $\mathcal{Q} = \mathcal{Q}(T)$ is the set of shift indices where such an interval exists. Note that

$$|\mathcal{Q}| = |\Upsilon|.$$

The set \mathcal{Q}^c is the complement of \mathcal{Q} , in the sense that it denotes valid shift indices where such an interval does not exist. However, it does not make sense to discuss possible shifts that place the point $p + q\varepsilon$ outside of the interval $[-1, 1]$, since every interval $T' \in \mathcal{T}^k \subseteq [-1, 1]$ will fail to contain this point. Hence, these are excluded from being in the complement set.

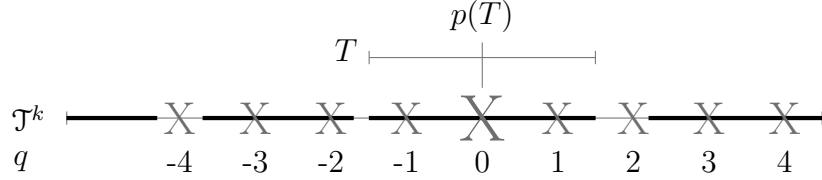


Figure 4.1 : Illustration of the set \mathcal{Q}^c . In this example the set $\mathcal{Q}^c = \{-4, 2\}$.

If $|\mathcal{Q}| < 2n + 1$, then the point p falls in a hole, i.e. there is some shift index \hat{q} such that every interval in $\mathcal{T}^{k,\hat{q}}$ does not contain p . Therefore, if $|\mathcal{Q}| < 2n + 1$, an additional small interval must be added to \mathcal{T}^k to ensure the All But One Condition is still met after a segment of T is removed. Note that

$$2n + 1 - |\mathcal{Q}| \leq |\mathcal{Q}^c|,$$

and if $|\mathcal{Q}| = 2n + 1$, then $|\mathcal{Q}^c| = 0$. For each shift $\hat{q} \in \mathcal{Q}^c$, an additional small interval must

be added; this interval must include the point $p + \widehat{q}\varepsilon$. The *hole* into which this small interval is added, i.e. the open interval containing $p + \widehat{q}\varepsilon$ that is not included in $\mathcal{T}^{k,\widehat{q}}$, is denoted as $H = H(T, \widehat{q})$. Let \mathcal{H} be the set of all holes to plug, that is,

$$\mathcal{H} = \{H(T, \widehat{q}) : T \in \mathcal{B}, \widehat{q} \in \mathcal{Q}^c(T)\}. \quad (4.2)$$

In the case $|\mathcal{Q}^c| > 1$, a different hole H must be added to \mathcal{H} for each shift index $\widehat{q} \in \mathcal{Q}^c$. It is feasible for a single hole $H \in \mathcal{H}$ to correspond to multiple points $p(T_i)$ for $T_i \in \mathcal{B}$. If there are two intervals $T_i, T_{i'} \in \mathcal{B}$ such that both $p(T_i)$ and $p(T_{i'})$ are contained in H , then H can be written as

$$H = H(T_i, \widehat{q}_i) = (T_{i'}, \widehat{q}_{i'}).$$

For each $H \in \mathcal{H}$, define the set $\widehat{P}(H)$ as

$$\widehat{P}(H) = \{p(T) : T \in \mathcal{B} \text{ for which there is some } \widehat{q} \in \mathcal{Q}^c(T) \text{ such that } p(T) + \widehat{q}\varepsilon \in H\}.$$

Set the number $\nu = \nu(H) = |\widehat{P}(H)|$ as the number of points $p(T_i)$ that have some shift index $q_i \in \mathcal{Q}^c$ such that $p(T_i) + q_i\varepsilon \in H$, that is, the number of unique pairs (T_i, \widehat{q}_i) that can be used to describe the same interval H .

4.2.3 Plug Stage

For each hole $H \in \mathcal{H}$, proceed as follows. Denote the endpoints of H as $H = (b_0, a_{\nu+1})$, where $\nu = \nu(H)$. By construction, the function ψ^k is linear on H ; denote its slope as m .

For each $p_i = p(T_i) \in \widehat{P}(H(T_i, \widehat{q}_i))$, construct a *plug* i.e. closed interval, denoted $\pi_i =$

$[a_i, b_i] \subset H$, with $1 \leq i \leq \nu$. For each i ,

$$p_i + \widehat{q}_i \varepsilon \in \pi_i.$$

Each plug will be constructed as to be disjoint, that is, for indices $i \neq i'$, that

$$\pi_i \cap \pi_{i'} = \emptyset.$$

Additionally, the values of ψ^{k+1} on each plug π_i will be assigned so that $\psi^{k+1}(\pi_i) = \psi^k(\widehat{p}_i)$.

Between plugs on H , ψ^{k+1} is forced to have slope $\widehat{m} = 1 - 2^{-k-1}$, with $m < \widehat{m}$.

These conditions on the plugs π_i and the function ψ^{k+1} are enforced by solving a linear system that specifies the length of the plugs. For simplicity, define $\widehat{p}_i = p_i + q_i \varepsilon$. Denote

$$\begin{aligned} f_0 &= \psi^k(b_0) \\ f_i &= \psi^k(\widehat{p}_i) \quad 1 \leq i \leq \nu \\ f_{\nu+1} &= \psi^k(a_{\nu+1}). \end{aligned} \tag{4.3}$$

The slope constraints provide $\nu + 1$ equations

$$\widehat{m}(a_i - b_{i-1}) = f_i - f_{i-1} \quad 1 \leq i \leq \nu + 1.$$

Since there are 2ν variables but $\nu + 1$ constraints, between each plug a symmetry constraint[†]

[†]Other constraints can be chosen.

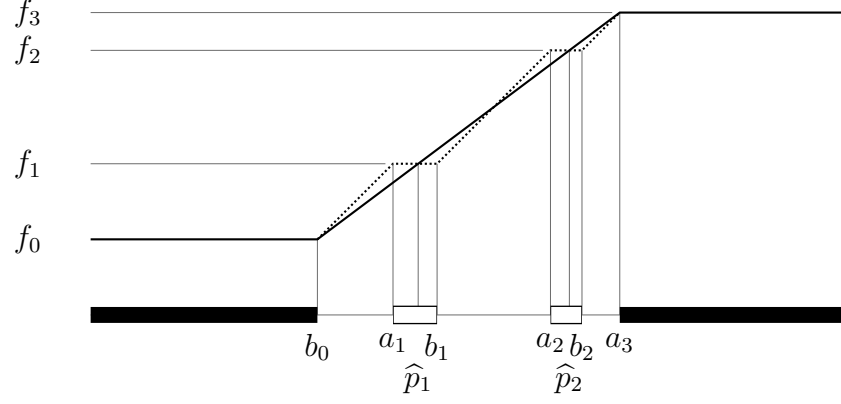


Figure 4.2 : Sketch of scenario for finding two plugs, with ψ^k solid and ψ^{k+1} dotted. Note the symmetry constraint $b_1 - \hat{p}_1 = \hat{p}_2 - a_2$ is enforced.

is enforced on ψ^{k+1} , that

$$b_i - \hat{p}_i = \hat{p}_{i+1} - a_{i+1} \quad 1 \leq i \leq \nu - 1.$$

Fig. 4.2 illustrates this setup in the case $\nu = 2$.

This provides the linear equation $Cx = z$, where

$$\left\{ \begin{array}{l} \nu + 1 \text{ slope equations} \\ \nu - 1 \text{ symmetry equations} \end{array} \right\} \left(\begin{array}{cccc} \widehat{m} & & & \\ & -\widehat{m} & \widehat{m} & \\ & & \ddots & \\ & & & -\widehat{m} & \widehat{m} \\ & & & & -\widehat{m} \\ \hline 0 & 1 & 1 & 0 \\ & & \ddots & \\ & & & 0 & 1 & 1 & 0 \end{array} \right) = C, \quad (4.4)$$

$$\left\{ \begin{array}{l} \nu + 1 \text{ slope equations} \\ \nu - 1 \text{ symmetry equations} \end{array} \right\} \left(\begin{array}{c} f_1 - f_0 \\ \vdots \\ \frac{f_{\nu+1} - f_{\nu}}{\widehat{p}_1 + \widehat{p}_2} \\ \vdots \\ \widehat{p}_{\nu-1} + \widehat{p}_{\nu} \end{array} \right) + \left(\begin{array}{c} \widehat{m}b_0 \\ \\ -\widehat{m}a_{\nu+1} \\ \\ \end{array} \right) = z \quad (4.5)$$

$$\begin{pmatrix} a_1 \\ b_1 \\ \vdots \\ a_i \\ b_i \\ \vdots \\ a_\nu \\ b_\nu \end{pmatrix} = x. \quad (4.6)$$

Permuting the rows of C creates a block diagonal matrix; since each block is invertible, C is invertible, so a unique solution exists. Given ψ^k is monotonic increasing, the plugs π_i are well-defined and do not overlap. On each plug, assign function values

$$\psi^{k+1}(\pi_i) = \psi^k(\widehat{p}_i).$$

For each H , add the plugs π_i to \mathcal{T}^k .

4.2.4 Break Stage

Recall the definition of the set \mathcal{P} , given as

$$\mathcal{P} = \{p \in T \ : \ T \in \mathcal{B}\}.$$

Also remember the definition of the set Υ ,

$$\Upsilon = \{T' \in \mathcal{T}^k : \exists q \in \{-2n, \dots, 2n\} \text{ such that } p + q\varepsilon \in T'\}.$$

At this point in the algorithm, for every point $p \in \mathcal{P}$, the shifted point $p + q\varepsilon$ belongs in some interval $T^q \in \mathcal{T}^k$ for every shift index $q \in \{-2n, \dots, 2n\}$ such that $p + q\varepsilon \in [-1, 1]$. As a result of the Plug Stage, the All But One condition is satisfied at points $p \in \mathcal{P}$ by ‘all’, not ‘all but one’, of the sets of shifted intervals. The *gap*, i.e. segment around p that is removed from T , must be chosen to be small enough so that it is contained within T^q for every shift index q . This restriction enforces the All But One Condition at the rest of points that are removed from T to break T in half.

Fix the constant scalar values $\alpha \in (0, 1)$ and $\beta \in (0, \frac{1}{2})$; without any loss of generality, assume $\alpha = 2/3$ and $\beta = 1/3$. Denote the interval T corresponding to the break point $p = p(T)$ as $T = [a, b]$. Define the values ρ_+ and ρ_- , which dictate the farthest one can go to the left or right of the point p while still being contained in T^q for each shift index q . Formally,

$$\rho_+ = \min_{T^q=[a^q, b^q] \in \Upsilon} (p + q\varepsilon) - a^q$$

$$\rho_- = \min_{T^q=[a^q, b^q] \in \Upsilon} b^q - (p + q\varepsilon)$$

To avoid creating gaps that overlap, the following distances are needed to constrain the

gap size:

$$\begin{aligned}
\delta_+ &= \min_{\substack{\tilde{p} \in \mathcal{P} \\ q \in \{-2n, \dots, 2n\} \\ \tilde{q} \in \{-2n, \dots, 2n\}}} \{(p - q\varepsilon) - (\tilde{p} - \tilde{q}\varepsilon) : p - q\varepsilon > \tilde{p} - \tilde{q}\varepsilon\} \\
\delta_- &= \min_{\substack{\tilde{p} \in \mathcal{P} \\ q \in \{-2n, \dots, 2n\} \\ \tilde{q} \in \{-2n, \dots, 2n\}}} \{(\tilde{p} - \tilde{q}\varepsilon) - (p - q\varepsilon) : p - q\varepsilon < \tilde{p} - \tilde{q}\varepsilon\}
\end{aligned} \tag{4.7}$$

Set the break radius ρ as

$$\rho = \min\{\alpha\rho_+, \alpha\rho_-, \beta\delta_+, \beta\delta_-\}. \tag{4.8}$$

The gap $G = G(p)$ is therefore given by

$$G = (p - \rho, p + \rho).$$

By the choice of ρ , for all shift indices q , shifted copies of G are always contained in T^q , that is,

$$G + q\varepsilon \subset T^q \text{ if } T^q - q\varepsilon \in [-1, 1].$$

Remove G from T ; this act breaks T into two new intervals T_- and T_+ , given by

$$T_- = [a, p - \rho] \quad T_+ = [p + \rho, b]. \tag{4.9}$$

Therefore,

$$T = T_- \cup G \cup T_+.$$

Let T_N be the next interval greater than T . Assign function values to the new iterate ψ^{k+1} as follows.

$$\begin{aligned}\psi^{k+1}(T_-) &= \psi^k(T) \\ \psi^{k+1}(T_+) &= \psi^k(T) + \eta\end{aligned}\tag{4.10}$$

where

$$\eta = \min \left\{ 2\widehat{m}\rho, \frac{1}{2}(\psi^k(T_N) - \psi^k(T)) \right\}.$$

This choice of η enforces monotonicity, since every new value assigned to ψ^{k+1} must lie between the previous value $\psi^k(T)$ and the next value $\psi^k(T_N)$ with η assigned as such.

By following this process to create gaps for every $p \in \mathcal{P}$, every interval $T \in \mathcal{B}$ is broken roughly in half. The set \mathcal{T}^{k+1} is therefore created from \mathcal{T}^k by replacing every interval $T \in \mathcal{B}$ with the intervals T_- and T_+ in its place.

These three stages are repeated iteratively for $k \in \mathbb{N}$. Practically, this can be continued until a finite machine tolerance has been reached. Therefore, this algorithm, which used the Fridman Strategy to create a function that satisfies Fridman's paper, is now complete. An implementation of this algorithm in Python is presented in Appendix D; the result through 11 iterations is displayed in Figure 4.3. It remains to be shown that the output of this algorithm satisfies the sufficient conditions for KST; this is proven in the next section.

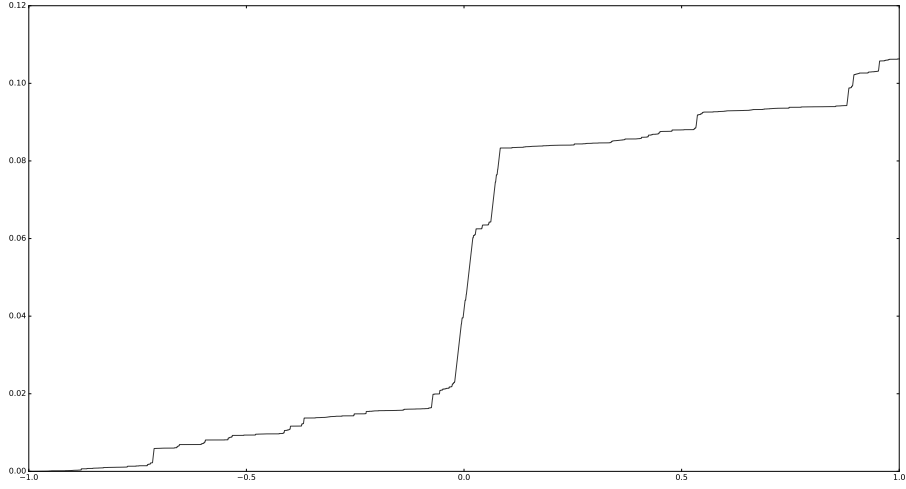


Figure 4.3 : Implementation of Fridman Strategy Algorithm for $k = 11$. Note that this function is has slope bounded above by one, and is monotonic increasing.

4.2.5 Proof of the Algorithm

This section proves that the function defined by Algorithm 1 satisfies the conditions in Section 4.1, including Fridman's Disjoint Image Condition, but not the Disjoint Image Condition itself. I systematically prove that the resulting functions ψ and ψ^k , and the sets \mathcal{T}^k , satisfy each of the conditions above, proceeding in the order the conditions were repeated in Section 4.1.

Claim 1. *For each refinement level $k \in \mathbb{N}$, the set of intervals \mathcal{T}^k satisfies the Refinement Condition.*

Proof. By construction, at refinement level k the size of the largest intervals are bounded by θ^k , and since $\theta \in (0, 1)$,

$$\lim_{k \rightarrow \infty} \theta^k = 0.$$

□

Claim 2. *For each refinement level $k \in \mathbb{N}$, the set of intervals \mathcal{T}^k satisfies the All But One Condition.*

Proof. This proof is done by induction on k .

- Base Case: For each shift index q , the set of intervals $\mathcal{T}^{0,q}$ consists of $\mathcal{T}^{0,q} = \{[-1 + q\varepsilon, 1 + q\varepsilon]\}$. Fix $x \in [0, 1]$. Since $\varepsilon \in (0, \frac{1}{2n}]$, for all shift indices $q \in \{0, \dots, 2n\}$,

$$x \in [-1 + q\varepsilon, 1 + q\varepsilon].$$

Therefore, $x \in \mathbb{I}$ for every q .

- Inductive Step: Fix $x \in [0, 1]$. Suppose this claim holds true at point x through refinement level $k \in \mathbb{N}$. Then, it still holds true after adding plugs at refinement level k , since nothing has been removed. Fix $p \in \mathcal{P}$, and let $G = G(p)$ be the gap containing point p . By construction, for each shift index $q \in \{0, \dots, 2n\}$, there is some interval $T^q \in \mathcal{T}^{k,q}$, possibly a newly added plug, such that $G \subset T^q$. As ρ was chosen to be less than δ_- and δ_+ , any possible shift of G will not overlap with any other possible gap $G' = G(p')$ being created at this refinement level. Therefore, every point in G is contained by some interval within the shifted sets $\mathcal{T}^{k+1,q}$ for all shift indices $q \neq 0$, and every point not in G is either unaffected or has been plugged sufficiently so that it is still covered by some interval (or plug) for at least $2n$ shift indices. Therefore, the All But One Condition is satisfied.

□

Claim 3. *At each refinement level $k \in \mathbb{N}$, the function ψ^k is monotonic increasing and satisfies the Bounded Slope Constraint.*

Proof. Clearly, ψ^k is continuous at refinement level $k \in \mathbb{N}$. It suffices to show the following two lemmas.

Lemma 1. *The function $\psi = \lim_{k \rightarrow \infty} \psi^k$ is well-defined, with convergence in the sup norm.*

Proof. Let $\mathcal{G}^k = \{G(p) : p \in \mathcal{P}\}$ be the set of all gaps and $\Pi^k = \{\pi : \pi \text{ plug added at step } k\}$ be the set of all plugs. Since the only places ψ^{k+1} differs from ψ^k are the gaps and plugs created at refinement level k ,

$$\begin{aligned} \|\psi^{k+1} - \psi^k\|_\infty &\leq \max \left\{ \sup_{G \in \mathcal{G}^{k+1}} \widehat{m}^k |G|, \sup_{\pi \in \Pi^{k+1}} \widehat{m}^k |\pi| \right\} \\ &< \sup_{G \in \mathcal{G}^{k+1}} |G| \\ &< \sup_{T \in \mathcal{T}^k} \text{Diam}(T) \\ &\leq \theta^k. \end{aligned}$$

Therefore, by the Weierstrass M-test, the function $\psi = \lim_{k \rightarrow \infty} \psi^k$ converges, and

$$\psi = \lim_{k \rightarrow \infty} \psi^k - \psi_0 = \lim_{k \rightarrow \infty} \sum_{j=1}^k \psi^j - \psi^{j-1}$$

is well-defined. □

Lemma 2. *Assume that ψ^k is monotonic increasing, constant on each interval $T \in \mathcal{T}^k$, and linear between such intervals with slope $\leq 1 - 2^{-k}$. Then, ψ^{k+1} is also monotonic increasing, constant on each interval $T \in \mathcal{T}^{k+1}$, and linear between such intervals, with slope $\leq 1 - 2^{-k-1}$.*

Proof. By construction, ψ^{k+1} is constant on each interval $T \in \mathcal{T}^{k+1}$; between such intervals, the value of ψ^{k+1} is given by linear interpolation. Let \mathcal{G}^k be given as in the proof of the previous lemma. For each gap $G \in \mathcal{G}^k$ formed between intervals from \mathcal{T}^{k+1} , let T be the interval to the left of G , and T_N be to the right. Exactly one of the following three cases must be true for each G :

1. The gap G existed between intervals T and T_N at refinement level k .
2. At least one of T or T_N is a plug created at this refinement level.
3. The gap G was created by breaking some interval in \mathcal{B}^k .

In each case, ψ^{k+1} maintains the desired properties:

1. ψ^{k+1} does not differ from ψ^k on G , and thus by the inductive hypothesis, ψ^{k+1} maintains the desired properties.
2. From the construction of the linear system, on interval T , $\psi^{k+1}(T) < \psi^{k+1}(T_N)$, and therefore ψ^{k+1} is still monotonic increasing. From the choice of the size of G , the slope of ψ^{k+1} is set to $m = 1 - 2^{-k-1}$ on G .
3. Since $\rho > 0$ and ψ^k is monotonic increasing, $\eta > 0$, so $\psi^{k+1}(T) < \psi^{k+1}(T_N)$. The value η was chosen so that the slope on g is

$$m = \frac{\eta}{2\rho} \leq \frac{1}{2} < 1 - 2^{-k-1}.$$

□

Thus, the proof of this claim is concluded. □

Corollary 1. *The function $\psi = \lim_{k \rightarrow \infty} \psi^k$ is Lipschitz continuous on $[-1, 1]$ with Lipschitz constant 1.*

Claim 4. *At refinement level $k \in \mathbb{N}$, the function ψ^k satisfies Fridman's Disjoint Image Condition.*

Proof. The constants $\lambda_1, \dots, \lambda_n$ are integrally independent, and the values of $\psi^k(T)$ are rational for each $T \in \mathcal{T}^k$. Therefore, for any two distinct multiindices $\mathbf{i} = (i_1, \dots, i_n)$ and $\mathbf{i}' = (i'_1, \dots, i'_n)$, the values of $\Psi^{k,q}(S_{\mathbf{i}}^k)$ and $\Psi^{k,q}(S_{\mathbf{i}'}^k)$ are distinct. As there are a finite number of squares $S_{\mathbf{i}}^k \in \mathcal{S}^k$, choose $\varepsilon_k > 0$ as the smallest distance between the values of $\Psi^{k,q}$ on squares at refinement level k . \square

While the function ψ provided by this algorithm satisfies Fridman's Disjoint Image Condition, it does not satisfy the Disjoint Image Condition. This can clearly be seen numerically. I implement this approach using the code in Appendix D through $k = 9$, and then I examine the action of function Ψ^7 on the set of squares \mathcal{S}^1 and \mathcal{S}^2 . The sets $\Psi^9(\mathcal{S}^1)$ and $\Psi^9(\mathcal{S}^2)$ are plotted in Figure 4.4 and Figure 4.5.

These images reveal that the subtle change from the Disjoint Image Condition to Fridman's Disjoint Image Condition relaxes the requirement of maintaining disjoint images of square under the function Ψ^q , to only examining the images of squares under the function $\Psi^{k,q}$. Fridman's condition is misleading: the function $\Psi^{k,q}$ is constant on squares, so the intervals $\Psi^{k,q}(S_1)$ and $\Psi^{k,q}(S_2)$ collapse to being points, which can be assumed to be rational values i.e. in \mathbb{Q} . Once Sprecher's reduction is used, this condition is guaranteed to be satisfied on the basis of Sprecher's scaling factors $\lambda_1, \dots, \lambda_n$ being integrally independent. Therefore, Fridman's Disjoint Image Condition does not sufficiently separate the images of

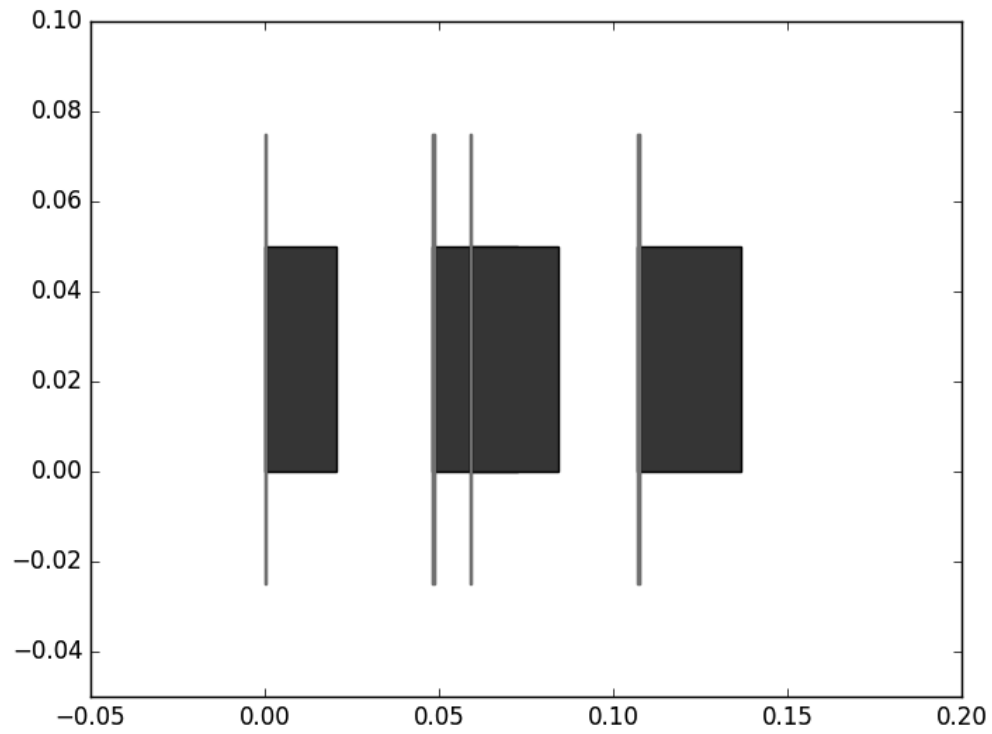


Figure 4.4 : Plot of $\Psi^9(\mathcal{S}^1)$. The tall lines demarcate the start of the image intervals; without them, the overlap of the image intervals would not necessarily be visible. Note that these image intervals clearly fail to be disjoint.

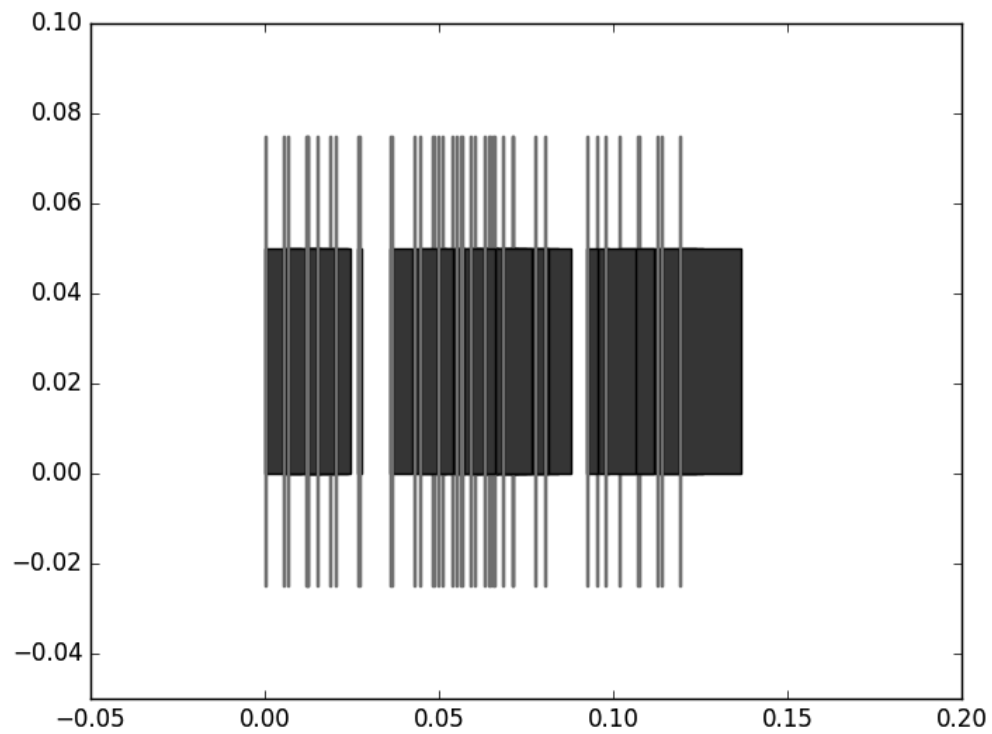


Figure 4.5 : Plot of $\Psi^9(\mathbb{S}^2)$. Again, the tall lines demarcate the start of the image intervals. Due to the significant overlap, even only for the second level of refinement, it is difficult to tell how many intervals overlap with each other; they are clearly not disjoint.

squares under Ψ^q .

As the Fridman Strategy is an integral component to Fridman’s proof, it is unclear whether Fridman’s result is valid. Fortunately, it is possible to prove that KST inner functions can be Lipschitz continuous using the proof supplied by Hedberg and Kahane that are mentioned in Section 2.2 (Hedberg [19]). This proof will be discussed in greater detail in Chapter 5; see [23] for a thorough explanation of this proof.

While Fridman’s weakened condition is not sufficient to show the Disjoint Image Condition, it is conceivable that the Fridman Strategy can still yield a Lipschitz continuous inner function if the Disjoint Image condition is enforced somehow at each refinement level of the construction. I do so in the next section by exploiting the Conservative Disjoint Image Condition at each refinement level.

4.3 The Fridman Strategy with the Conservative Disjoint Image Condition

In this section, I describe an alternative framework for the algorithmic computation of Lipschitz continuous inner functions for KST. This framework seeks to maintain the Fridman Strategy, but it replaces the weaker Fridman’s Disjoint Image Condition with the Conservative Disjoint Image Condition. The conservative image intervals are constructed using the Lipschitz bound of the final function Ψ , as it is the only information known before the construction begins. In Section 4.3.1, I describe the approach I take to ensure the conditions listed in Section 3.4 are met. I then describe the execution of each stage in the Fridman Strategy while meeting these conditions, taking care to specifically meet the Conservative

Disjoint Image Condition and the Bounded Slope Condition. This discussion ultimately establishes that for Fridman's strategy to satisfy the Conservative Disjoint Image Condition, more than half of each interval must be removed, yet doing so would violate the All But One Condition. This result therefore suggests that Fridman's approach may not be viable for computing a Lipschitz inner function. To rectify this concern, a fundamentally different approach is outlined in Chapter 5.

4.3.1 Motivation for Using the Conservative Disjoint Image Condition

Consider the Conservative Disjoint Image Condition, restated here. Recall that this condition enforces that the images of the squares $S_{\mathbf{i}}^k \in \mathcal{S}^k$ under Ψ^q remain disjoint, and that this condition does so by considering larger intervals $\Delta_{\mathbf{i}}^k$ such that $\Psi^q(S_{\mathbf{i}}^k) \subseteq \Delta_{\mathbf{i}}^k$, and then enforcing that these larger intervals $\Delta_{\mathbf{i}}^k$ are disjoint instead.

Condition 6 (Conservative Disjoint Image). *For any intervals $\Delta_{\mathbf{i}}^k, \Delta_{\mathbf{i}'}^k \in \square^k$, where $\Psi^q(S_{\mathbf{i}}^k) \subseteq \Delta_{\mathbf{i}}^k$ and $\Psi^q(S_{\mathbf{i}'}^k) \subseteq \Delta_{\mathbf{i}'}^k$,*

$$\Delta_{\mathbf{i}}^k \cap \Delta_{\mathbf{i}'}^k = \emptyset.$$

If the values of ψ^k are fixed across refinement levels at the left endpoints of each interval, and if the function ψ^k is maintained to be monotonic nondecreasing, the following relationship holds between refinement levels and function values for any square $S_{\mathbf{i}}^k \in \mathcal{S}^k$ at refinement level k :

$$\Psi^k(S_{\mathbf{i}}^k) \subseteq \Psi^{k+1}(S_{\mathbf{i}}^k) \subseteq \Psi^{k+2}(S_{\mathbf{i}}^k) \subseteq \dots \subseteq \Psi(S_{\mathbf{i}}^k). \quad (4.11)$$

In this relation, $\Psi^k(S_i^k)$ is a constant (fixed at the lower left corner of square S_i^k), but for any $k' > k$, $\Psi^{k'}(S_i^k)$ is possibly interval, since the square S_i^k may have been broken between successive refinement levels and $\Psi^{k'}$ would be linear on the induced gaps, not constant. Therefore, the Conservative Disjoint Image Condition demands finding intervals Δ_i^k such that

$$\Psi(S^k) \subset \Delta_i^k. \quad (4.12)$$

If the interval Δ_i^k is chosen minimally, so that

$$\Delta_i^k = \left[\Psi_i^k, \lim_{k' \rightarrow \infty} \max_{\mathbf{x} \in S_i^k} \Psi^{k'}(\mathbf{x}) \right],$$

then the Conservative Disjoint Image Condition and the Disjoint Image Condition are identical. The quantity $\lim_{k' \rightarrow \infty} \max_{\mathbf{x} \in S_i^k} \Psi^{k'}(\mathbf{x})$ exists, as the domain S_i^k is fixed independent of k' , and by definition $\lim_{k' \rightarrow \infty} \Psi^{k'} = \Psi$. However, at refinement level k , not much is known about the function Ψ , and even less about the quantity $\lim_{k' \rightarrow \infty} \max_{\mathbf{x} \in S_i^k} \Psi^{k'}(\mathbf{x})$. Therefore, such a minimal interval Δ_i^k is hard to compute, and currently there is no method known to find such intervals.

Instead, if the Bounded Slope Condition holds, then a Lipschitz constant for Ψ^k is known. This Lipschitz constant is exploited to define the (non-minimal) intervals Δ_i^k that allow for the Conservative Disjoint Image Condition to be satisfied. The length of these intervals Δ_i^k , the length of the break in T_i^k , and the values of ψ^k need to be chosen as to maintain all of the following conditions:

- Bounded Slope Condition;

- Monotonicity Condition;
- Conservative Disjoint Image Condition; and
- All But One Condition.

In the rest of this section, I outline how to construct intervals \mathcal{T}^k , functions ψ^k , and sets of intervals $\square^k = \{\Delta_i^k\}_i$ that satisfy these conditions, with particular focusing on meeting the Conservative Disjoint Image Condition and the Bounded Slope Condition.

Like the previous algorithm, this approach uses the Fridman Strategy as a framework for iterative refinement:

1. **Find Stage:** Find the intervals $T \in \mathcal{T}^k$ that need to be broken.
2. **Plug Stage:** Determine whether the break points $p \in T$ lies in all the shifted families $\mathcal{T}^{k,q}$, or all but one. If there is some shift \hat{q} such that for all $T \in \mathcal{T}^{k,\hat{q}}$, the shifted point $p + \hat{q}\varepsilon \notin T$, then add the necessary small *plug* segment to into the set $\mathcal{T}^{k,q}$.
3. **Break Stage:** Remove from T some small segment that includes the break point p , thereby breaking T in half.

While the previous algorithm crafted a linear system of equations to enforce conditions during the Plug Stage, this approach forms a linear program to enforce conditions during the Break Stage. Each of the three stages will now be described in greater detail.

Consider the constants $\lambda_1, \dots, \lambda_n$, introduced during Sprecher's reduction from multiple inner functions $\psi_{p,q}$ to one inner function ψ . Order these constants so that $\lambda_n > \dots > \lambda_2 > \lambda_1$. Set $\varepsilon = \frac{1}{2n}$. These constants will remain fixed throughout the construction. Begin with $\mathcal{T}^0 = \{[-1, 1]\}$ and $\psi^0 \equiv 0$.

4.3.2 Find and Plug Stages

The Find Stage is identical to its counterpart in the previous algorithm, given in Section 4.2.2. The Plug Stage is nearly the same to that in Section 4.2.3; however, the symmetry constraints are relaxed from the Linear System and the slope equations are treated as inequalities instead. These changes allow for the plugs to be created arbitrarily small. For small enough plugs, the Conservative Disjoint Image Condition is guaranteed to be met, since the plugs are being added precisely in areas of \mathbb{I}^n such that the image of the plug under Ψ^q is centered in a gap between Δ_i^k intervals.

4.3.3 Break Stage

Suppose that the necessary plugs have been found and added into the set \mathcal{T}^k , and the function ψ^k has been suitably updated, so that all of the sufficient conditions have been met.

I explicitly consider the case $n = 2$, which benefits from the ability to draw precise pictures. The case is similar for $n > 2$, but requires a significant amount of notation to be introduced, and is therefore omitted. Each interval $T_i^k \in \mathcal{B}^k$, will be broken into left and right intervals T_i^{k+1} and T_N^{k+1} , where the break point $p \in T_i^k$ yet $p \notin T_i^{k+1}$ nor $p \notin T_N^{k+1}$. The subscript i consecutively enumerates the intervals of \mathcal{T}^k (not \mathcal{B}^k), and the subscript N denotes a new interval is being added.

For the rest of this section, the q superscript of functions ψ^q and Ψ^q , and of the iterates $\psi^{k,q}$ and $\Psi^{k,q}$ will often be dropped, to declutter some of the notation.

Recall that for refinement levels $k' \geq k$, the values of $\psi^{k'}$ are fixed so that $\psi^{k'} = \psi^k$ on the left endpoints of intervals in \mathcal{T}^k . Therefore, it is expedient to have the left half of

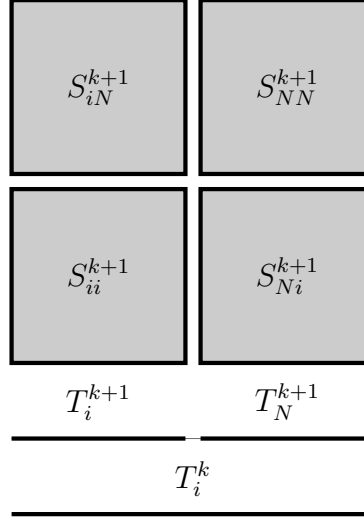


Figure 4.6 : Setup of squares after breaking step k , thereby creating squares $k + 1$. Note that the notation reflects which values are fixed to which squares during refinement: the squares that maintain index i share the function value (for that coordinate direction) as with its parent, whereas the squares with index N reflect a new value that has been assigned.

the interval T_i^k post-break maintain the same label, and to denote the new interval i.e. the right half with the subscript N . Figure 4.6 is in accordance with this notation. After the k th refinement level is complete, the intervals in \mathcal{T}^{k+1} are re-enumerated to accomodate the increase in number of intervals, as to correspond to the new ordering. Additionally, having the left endpoints fixed allows for the definition of the fixed constants ψ_i^k , given by

$$\psi_i^k = \psi^k(T_i^k). \quad (4.13)$$

These constants thereby define the constant values $\Psi_{\mathbf{i}}^k$, given by

$$\Psi_{\mathbf{i}}^k = \Psi^k(S_{\mathbf{i}}^k) = \sum_{p=1}^n \lambda_p \psi_{i_p}^k. \quad (4.14)$$

Let $\Theta : \mathcal{S}^k \rightarrow \mathbb{R}$ be the *variation* of a square $S_{\mathbf{i}}^k \in \mathcal{S}^k$, defined as

$$\Theta(S_{i_1, \dots, i_n}^k) = \sum_{p=1}^n \lambda_p |T_{i_p}^k|.$$

Informally, Θ describes the maximum Ψ can vary over a given square, assuming the inner function ψ is Lipschitz with constant 1. Note that for any two squares S and S' , if $S \subseteq S'$, then

$$\Theta(S) \leq \Theta(S'), \quad (4.15)$$

with strict inequality if S is strictly contained in S' .

Define $\Delta_{\mathbf{i}}^k \subset \mathbb{R}$ for a square $S_{\mathbf{i}}^k$ as

$$\Delta_{\mathbf{i}}^k = [\Psi_{\mathbf{i}}^k, \Psi_{\mathbf{i}}^k + \Theta(S_{\mathbf{i}}^k)].$$

With this definition,

$$\begin{aligned} \Psi(S_{\mathbf{i}}^k) &\subseteq \Delta_{\mathbf{i}}^k, \quad \text{and} \\ \lim_{k' \rightarrow \infty} \max_{\mathbf{x} \in S_{\mathbf{i}}^k} \Psi(x) &\leq \Psi_{\mathbf{i}}^k + \Theta(S_{\mathbf{i}}^k). \end{aligned} \quad (4.16)$$

Recall that the Conservative Disjoint Image Condition states the following:

Condition 6 (Conservative Disjoint Image). *For any intervals $\Delta_{\mathbf{i}}^k, \Delta_{\mathbf{i}'}^k \in \square^k$, where $\Psi^q(S_{\mathbf{i}}^k) \subseteq \Delta_{\mathbf{i}}^k$ and $\Psi^q(S_{\mathbf{i}'}^k) \subseteq \Delta_{\mathbf{i}'}^k$,*

$$\Delta_{\mathbf{i}}^k \cap \Delta_{\mathbf{i}'}^k = \emptyset.$$

There are two ways to satisfy this condition:

$$\Delta_{\mathbf{i}}^k < \Delta_{\mathbf{i}'}^k \quad \text{or} \quad \Delta_{\mathbf{i}}^k > \Delta_{\mathbf{i}'}^k. \quad (4.17)$$

At first glance, enforcing the Conservative Disjoint Image Condition would require enumerating every square that is broken while going from refinement level k to refinement level $k+1$, and checking that one of the two inequalities in Equation 4.17 is satisfied. For any one interval $T_i \in \mathcal{B}^k$, this could require checking that any square $S_{\mathbf{i}}^{k+1} \in \mathcal{S}^{k+1}$ whose multiindex \mathbf{i} includes the index i in any position. This formulation would require the solution of a large mixed-integer program for each break made, with linear constraints to describe bounds on the values ψ_N^{k+1} , and binary constraints to enforce that one of the two inequalities holds from Equation 4.17.

However, with care, it is not necessary to consider every square that shares the index i . Using the fact that $\lambda_1 < \lambda_2$, the binary constraints can be removed, as the values that Ψ^{k+1} can achieve on the squares $S_{\mathbf{i}}^{k+1}$ are ordered. This reduction is shown in next subsection. Moreover, assuming the Bounded Slope Condition and the Monotonicity Condition are satisfied, it suffices to enforce the Conservative Disjoint Image Condition only on squares that are split when due to breaking $T_i \in \mathcal{B}^k$, i.e. squares who are formed by a Cartesian product involving T_i . This result is shown in the subsection after next. These claims enable the construction of a small linear program to construct function values ψ^{k+1} and intervals \mathcal{T}^{k+1} that satisfy the Bounded Slope Condition, the Monotonicity Condition, and the Conservative Disjoint Image Condition.

Removing the Binary Constraints

Using the fact that $\lambda_1 < \lambda_2$, the binary constraints can be removed, as the values that Ψ^{k+1} can achieve on the squares S_i^{k+1} are ordered. This reduction enables Claim 5; The values ψ_i^{k+1} and ψ_N^{k+1} are sufficiently close if they allow for the Conservative Disjoint Image Condition to be satisfied.

Claim 5. *Let $\lambda_2 > \lambda_1$. Then, for ψ_N^{k+1} sufficiently close to ψ_i^{k+1} , we have*

$$\Delta_{iN}^{k+1} > \Delta_{Ni}^{k+1}.$$

Proof. By monotonicity, write $\psi_N^{k+1} = \psi_i^{k+1} + \theta$ for some $\theta > 0$. Then,

$$\begin{aligned} \lambda_1 \psi_N^{k+1} + \lambda_2 \psi_i^{k+1} &= \lambda_1 \psi_i^{k+1} + \lambda_1 \theta + \lambda_2 \psi_i^{k+1} \\ &< \lambda_1 \psi_i^{k+1} + \lambda_2 \theta + \lambda_2 \psi_i^{k+1} && \text{since } \lambda_2 > \lambda_1 \\ &= \lambda_1 \psi_i^{k+1} + \lambda_2 \psi_N^{k+1} && \text{by definition of } \psi_N^{k+1}. \end{aligned}$$

All of these functions are continuous, and the image of a connected domain (such as a square) under a continuous function is connected. Therefore, if ψ_N^{k+1} is chosen so that

$$\Psi^{k+1}(S_{Ni}^{k+1}) + \Theta(S_{Ni}^{k+1}) \leq \Psi^{k+1}(S_{iN}^{k+1}),$$

then

$$\Delta_{iN}^{k+1} > \Delta_{Ni}^{k+1}.$$

□

Reducing the Scope of Breaking Constraints

Let the square S^{k+1} be formed by a break going from refinement level k to level $k + 1$. It is necessary to show that the image of S^{k+1} under Ψ does not intersect with the image of any other square at that refinement level.

Claim 6. *Suppose the Conservative Disjoint Image Condition is satisfied through refinement level k , and suppose the interval $T_i^k \in \mathcal{B}^k$ was broken into T_i^{k+1} and T_N^{k+1} , thereby defining a new square $S_{\mathbf{i}'}^{k+1}$ and new conservative image interval $\Delta_{\mathbf{i}'}^{k+1}$ for some multiindex \mathbf{i}' . If $\Delta_{\mathbf{i}'}^{k+1} \subseteq \Delta_{\mathbf{i}}^k$, then for any other multiindex \mathbf{j} at refinement level k , with $\mathbf{j} \neq \mathbf{i}$, it holds that*

$$\Delta_{\mathbf{i}'}^{k+1} \cap \Delta_{\mathbf{j}}^k = \emptyset.$$

Proof. By the Conservative Disjoint Image Condition, $\Delta_{\mathbf{i}}^k \cap \Delta_{\mathbf{j}}^k = \emptyset$. As $\Delta_{\mathbf{i}'}^{k+1} \subseteq \Delta_{\mathbf{i}}^k$,

$$\Delta_{\mathbf{i}'}^{k+1} \cap \Delta_{\mathbf{j}}^k = \emptyset.$$

□

This claim implies the following corollary.

Corollary 2. *The Conservative Disjoint Image Condition holds if $\Delta_{\mathbf{i}'}^{k+1} \subseteq \Delta_{\mathbf{i}}^k$ and $\Delta_{\mathbf{i}'}^{k+1}$ does not intersect any other $\Delta_{\mathbf{i}''}^{k+1} \subset \Delta_{\mathbf{i}}^k$.*

Given this corollary, consider the following setup. Let \mathbf{i} be the multiindex at refinement level k all of whose entries are equal to i . Let \mathbf{i}' and \mathbf{i}'' be multiindices at refinement level $k + 1$ that describe squares formed by breaking $S_{\mathbf{i}}^k$. Additionally, let \mathbf{j} be a multiindex at

refinement level k with at least one entry equal to i , but not all of whose entries are equal to i . Let multiindices \mathbf{j}' and \mathbf{j}'' describe a squares at refinement level $k + 1$ formed from $S_{\mathbf{j}}^k$ by breaking T_i^k . Intuitively, the \mathbf{i} -related multiindices relate to squares formed by directly breaking a square $S_{\mathbf{i}}^k$ from refinement level k , one who was formed by a Cartesian product of intervals all of whom were each sufficiently large to need refinement. For example, if $T_i \in \mathcal{T}^k$ was large enough so that $T_i \in \mathcal{B}^k$, one square $S_{\mathbf{i}}^k$ is formed by the n -fold Cartesian product of T_i . In contrast, the \mathbf{j} -related multiindices are squares that are casualties of breaking $S_{\mathbf{i}}^k$. Theses squares are formed by a Cartesian product involving one of the intervals that was sufficiently large to require refinement. The breaks introduced when breaking $S_{\mathbf{i}}^k$ (that is, breaking T_i) mean that $S_{\mathbf{j}}^k$ was broken as well, and therefore these squares need to be considered as well.

As this section considers the case $n = 2$, there are only two entries per multiindex. Then, $\mathbf{i} = (i, i)$ and either $\mathbf{j} = (j, i)$ or $\mathbf{j} = (i, j)$ for some index $j \neq i$. In the setup of Figure 4.6, there are four cases for the multiindices \mathbf{i}' and \mathbf{i}'' : (i, i) , (i, N) , (N, i) , and (N, N) . Similarly, there are two cases for \mathbf{j}' and \mathbf{j}'' for each option for \mathbf{j} : (j, i) , (j, N) , (i, j) , and (N, j) .

To enforce the Conservative Disjoint Image Condition on all new squares by only considering what happens to square $S_{\mathbf{i}}^k$, the following properties must be shown:

1. $\Delta_{\mathbf{i}'}^{k+1} \subseteq \Delta_{\mathbf{i}}^k$.
2. $\Delta_{\mathbf{j}'}^{k+1} \subseteq \Delta_{\mathbf{j}}^k$.
3. $\Delta_{\mathbf{i}'}^{k+1} \cap \Delta_{\mathbf{i}''}^{k+1} = \emptyset$.
4. $\Delta_{\mathbf{j}'}^{k+1} \cap \Delta_{\mathbf{j}''}^{k+1} = \emptyset$.

The first two of these are shown summarily below.

Claim 7. *Suppose in the above setup, all conditions have been satisfied in construction through refinement level k , and the Bounded Slope and Monotonicity Conditions hold for refinement level $k + 1$. Then, for all multiindices \mathbf{i}' listed above,*

$$\Delta_{\mathbf{i}'}^{k+1} \subseteq \Delta_{\mathbf{i}}^k.$$

Proof. It suffices to show that $\Psi_{\mathbf{i}'}^{k+1} \geq \Psi_{\mathbf{i}}^k$ and that $\Psi_{\mathbf{i}'}^{k+1} + \Theta(S_{\mathbf{i}'}^{k+1}) \leq \Psi_{\mathbf{i}}^k + \Theta(S_{\mathbf{i}}^k)$.

By monotonicity of the inner functions, $\Psi_{\mathbf{i}'}^{k+1} \geq \Psi_{\mathbf{i}}^k$ for all squares $S_{\mathbf{i}'}^{k+1} \subseteq S_{\mathbf{i}}^k$. For the other side of the interval, the multiindex (N, N) is considered first; in this case, by the definition of the variation Θ and the function Ψ^{k+1} ,

$$\Psi_{NN}^{k+1} + \Theta(S_{NN}^{k+1}) = (\lambda_1 + \lambda_2) (\psi_N^{k+1} + |T_N^{k+1}|). \quad (4.18)$$

Having assumed the Bounded Slope Condition holds for ψ^{k+1} , the following inequality holds:

$$\begin{aligned} \psi_N^{k+1} &\leq \psi_i^k + (1 - 2^{-(k+1)}) [|T_i^k| - |T_i^{k+1}| - |T_N^{k+1}|] \\ &= |T_i^k| - (2^{-(k+1)}) [|T_i^k| - |T_N^{k+1}|] - (1 - 2^{-(k+1)}) |T_i^{k+1}|. \end{aligned} \quad (4.19)$$

Therefore,

$$\begin{aligned}
\Psi_{NN}^{k+1} + \Theta(S_{NN}^{k+1}) &= (\lambda_1 + \lambda_2) (\psi_N^{k+1} + |T_N^{k+1}|) \\
&\leq (\lambda_1 + \lambda_2) (\psi_i^k + (1 - 2^{-(k+1)}) [|T_i^k| - |T_i^{k+1}| - |T_N^{k+1}|] + |T_N^{k+1}|) \\
&= (\lambda_1 + \lambda_2) (\psi_i^k + |T_i^k|) \\
&\quad - (\lambda_1 + \lambda_2)(2^{-(k+1)}) [|T_i^k| - |T_N^{k+1}|] \\
&\quad - (\lambda_1 + \lambda_2)(1 - 2^{-(k+1)}) |T_i^{k+1}| \\
&\leq (\lambda_1 + \lambda_2) (\psi_i^k + |T_i^k|) \\
&= \Psi_{ii}^k + \Theta(S_{ii}^k).
\end{aligned} \tag{4.20}$$

The same proof can be completed for the other three multiindices, using the steps above and invoking monotonicity of ψ^{k+1} where appropriate. \square

A similar result can be shown for the second claim.

Claim 8. *Suppose in the above setup, all conditions have been satisfied in construction through refinement level k , and the Bounded Slope and Monotonicity Conditions hold for refinement level $k + 1$. Then, for multiindices \mathbf{j} and \mathbf{j}' listed above,*

$$\Delta_{\mathbf{j}'}^{k+1} \subseteq \Delta_{\mathbf{j}}^k.$$

Proof. By symmetry, it suffices to show $\Delta_{ji}^{k+1}, \Delta_{jN}^{k+1} \subseteq \Delta_{ji}^k$.

First consider Δ_{ji}^{k+1} ; it is wanted to show $\Delta_{ji}^{k+1} \subseteq \Delta_{ji}^k$. Note that $\Psi_{ji}^k = \Psi_{ji}^{k+1}$ since the values of Ψ^k are fixed at the left endpoints of squares for all future $k' > k$, and $k + 1 > k$.

Additionally, recall that by Equation 4.15, since $T_i^{k+1} \subset T_i^k$, it holds that

$$\Theta(S_{ji}^{k+1}) < \Theta(S_{ji}^k).$$

Therefore,

$$\begin{aligned} \Psi_{ji}^{k+1} + \Theta(S_{ji}^{k+1}) &= \Psi_{ji}^k + \Theta(S_{ji}^{k+1}) \\ &\leq \Psi_{ji}^k + \Theta(S_{ji}^k). \end{aligned} \tag{4.21}$$

Thus, it holds that

$$\begin{aligned} \Delta_{ji}^{k+1} &= \left[\Psi_{ji}^{k+1}, \quad \Psi_{ji}^{k+1} + \Theta(S_{ji}^{k+1}) \right] \\ &\subset \left[\Psi_{ji}^k, \quad \Psi_{ji}^k + \Theta(S_{ji}^k) \right] \\ &= \Delta_{ji}^k. \end{aligned}$$

Next consider Δ_{jN}^{k+1} ; it is wanted to show $\Delta_{jN}^{k+1} \subseteq \Delta_{jN}^k$. By assumption, ψ^{k+1} satisfies the Bounded Slope constraint, and since the value of ψ^k are fixed at the left corners, the following hold:

$$\begin{aligned} \psi_i^{k+1} &= \psi_i^k \\ \psi_j^{k+1} &= \psi_j^k \\ \psi_N^{k+1} &\leq \psi_i^k + (1 - 2^{-(k+1)}) \left[|T_i^k| - |T_i^{k+1}| - |T_N^{k+1}| \right]. \end{aligned} \tag{4.22}$$

Observe that by monotonicity, $\psi_N^{k+1} > \psi_i^{k+1}$; thus,

$$\begin{aligned}
\Psi_{jN}^{k+1} &= \lambda_1 \psi_j^{k+1} + \lambda_2 \psi_N^{k+1} \\
&> \lambda_1 \psi_j^{k+1} + \lambda_2 \psi_i^{k+1} \\
&= \lambda_1 \psi_j^k + \lambda_2 \psi_i^k \quad \text{by keeping values fixed on the left corners} \\
&= \Psi_{jN}^k.
\end{aligned} \tag{4.23}$$

By the definition of Θ ,

$$\begin{aligned}
\Psi_{jN}^{k+1} + \Theta(S_{jN}^{k+1}) &= \lambda_1 \psi_j^{k+1} + \lambda_2 \psi_N^{k+1} + \lambda_1 |T_j^{k+1}| + \lambda_2 |T_N^{k+1}| \\
&= \lambda_1 (\psi_j^{k+1} + |T_j^{k+1}|) + \lambda_2 (\psi_N^{k+1} + |T_N^{k+1}|).
\end{aligned} \tag{4.24}$$

Again, having assumed the Bounded Slope Condition holds,

$$\psi_N^{k+1} \leq \psi_i^k + (1 - 2^{-(k+1)}) [|T_i^k| - |T_i^{k+1}| - |T_N^{k+1}|]. \tag{4.25}$$

Therefore,

$$\begin{aligned}
\psi_N^{k+1} + |T_N^{k+1}| &\leq \psi_i^k + (1 - 2^{-(k+1)}) [|T_i^k| - |T_i^{k+1}| - |T_N^{k+1}|] + |T_N^{k+1}| \\
&= \psi_i^k + |T_i^k| - (2^{-(k+1)}) (|T_i^k| - |T_N^{k+1}|) - (1 - 2^{-(k+1)}) |T_i^{k+1}| \\
&< \psi_i^k + |T_i^k|.
\end{aligned} \tag{4.26}$$

Combining Equation 4.24 and Equation 4.26,

$$\begin{aligned}
 \Psi_{jN}^{k+1} + \Theta(S_{jN}^{k+1}) &< \lambda_1(\psi_j^k + |T_j^k|) + \lambda_2(\psi_i^k + |T_i^k|) \\
 &= \Psi_{ji}^k + \Theta(S_{ji}^k).
 \end{aligned} \tag{4.27}$$

Therefore,

$$\begin{aligned}
 \Delta_{jN}^{k+1} &= \left[\Psi_{jN}^{k+1}, \quad \Psi_{jN}^{k+1} + \Theta(S_{jN}^{k+1}) \right] \\
 &\subset \left[\Psi_{ji}^k, \quad \Psi_{jN}^{k+1} + \Theta(S_{ji}^k) \right] \\
 &\subset \left[\Psi_{ji}^k, \quad \Psi_{ji}^k + \Theta(S_{ji}^k) \right] \\
 &= \Delta_{ji}^k.
 \end{aligned}$$

□

These two claims highlight that this breaking process creates a type fixed point iteration: as the squares are refined, the image of each subsquare remains inside the image of its parent square. This pattern is notable, since without these claims it could only be assumed that the squares themselves were nested, not their images; it would otherwise have been possible for the image sets of $\Delta_{\mathbf{i}}^{k+1}$ to spill beyond the image set of $\Delta_{\mathbf{i}}^k$.

Of the four tasks listed on page 74 above, the third and fourth tasks are still outstanding. These claims are resolved with the following statement:

Claim 9. *Suppose in the above setup, all conditions have been satisfied in construction through refinement level k , and the Bounded Slope and Monotonicity Conditions hold for refinement level $k+1$. Then, for multiindices \mathbf{i}' , \mathbf{i}'' , \mathbf{j}' , and \mathbf{j}'' as described above, the following*

must be enforced directly:

$$\begin{aligned}\Delta_{\mathbf{i}'}^{k+1} \cap \Delta_{\mathbf{i}''}^{k+1} &= \emptyset, \quad \text{and} \\ \Delta_{\mathbf{j}'}^{k+1} \cap \Delta_{\mathbf{j}''}^{k+1} &= \emptyset.\end{aligned}\tag{4.28}$$

This claim can only be proven if the new image intervals at refinement level $k+1$ are chosen correctly. To make these choices that will directly enforce the statements in Equation 4.28, I construct a linear program. I focus first on the constraints related to \mathbf{i} and \mathbf{i}'' . Due to the ordering presented in Subsection 4.3.3, for $\lambda_2 > \lambda_1$, the following need to be enforced:

$$\begin{aligned}\Psi_{ii}^{k+1} + \Theta(S_{ii}^{k+1}) &\leq \Psi_{Ni}^{k+1} \\ \Psi_{Ni}^{k+1} + \Theta(S_{Ni}^{k+1}) &\leq \Psi_{iN}^{k+1} \\ \Psi_{iN}^{k+1} + \Theta(S_{iN}^{k+1}) &\leq \Psi_{NN}^{k+1}.\end{aligned}\tag{4.29}$$

These are equivalent to the following:

$$\begin{aligned}\lambda_1 \psi_i^{k+1} + \lambda_2 \psi_i^{k+1} + \lambda_1 |T_i^{k+1}| + \lambda_2 |T_i^{k+1}| &\leq \lambda_1 \psi_N^{k+1} + \lambda_2 \psi_i^{k+1} \\ \lambda_1 \psi_N^{k+1} + \lambda_2 \psi_i^{k+1} + \lambda_1 |T_N^{k+1}| + \lambda_2 |T_i^{k+1}| &\leq \lambda_1 \psi_i^{k+1} + \lambda_2 \psi_N^{k+1} \\ \lambda_1 \psi_i^{k+1} + \lambda_2 \psi_N^{k+1} + \lambda_1 |T_i^{k+1}| + \lambda_2 |T_N^{k+1}| &\leq \lambda_1 \psi_N^{k+1} + \lambda_2 \psi_N^{k+1}.\end{aligned}\tag{4.30}$$

After some cancellation and rearranging, and adding in the Bounded Slope Condition, I pose the following linear program.

Claim 10. *Let λ_1 , λ_2 , $|T_i^k|$, and ψ_i^k be fixed. Let \widehat{m}_{k+1} be the maximal slope of ψ^{k+1} under the Bounded Slope Condition, that is, $\widehat{m}_{k+1} \leq 1 - 2^{-(k+1)}$. Let $\mathbf{x} = (\rho, |T_i^{k+1}|, |T_N^{k+1}|, \psi_N^{k+1})$,*

where ρ is the size of the gap between T_i^{k+1} and T_N^{k+1} . Consider the following linear program:

$$\begin{aligned}
 & \min \quad x_1 \\
 & \text{subject to} \quad \begin{bmatrix} 0 & \lambda_1 + \lambda_2 & 0 & -\lambda_1 \\ 0 & \lambda_2 & \lambda_1 & \lambda_1 - \lambda_2 \\ 0 & \lambda_1 & \lambda_2 & -\lambda_1 \\ -\widehat{m}_{k+1} & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ -1 & -1 & -1 & 0 \end{bmatrix} \mathbf{x} \leq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ |T_i^k| \\ -|T_i^k| \end{bmatrix} \quad (4.31)
 \end{aligned}$$

Any feasible solution to this linear program satisfies the Bounded Slope Condition and the Monotonicity Condition, and $\Delta_{\mathbf{i}'}^{k+1} \cap \Delta_{\mathbf{i}''}^{k+1} = \emptyset$ for the multiindices \mathbf{i}' and \mathbf{i}'' described previously.

A similar linear program can be constructed for the corresponding statement involving the multiindices \mathbf{j} and \mathbf{j}' . Other objective functions can be chosen as well; what is important is that these linear programs are feasible.

Once these linear programs have been solved, the break size ρ is determined. The set \mathcal{T}^{k+1} is created by repeating this process for every $T \in \mathcal{B}$, thereby breaking every interval as to enforce the Refinement Condition. Doing so determines the function values for ψ_i^{k+1} on the intervals $T_i \in \mathcal{T}^{k+1}$, and interpolating linearly between the intervals at this refinement level provides the definition of ψ^{k+1} on the entire interval $[-1, 1]$. Therefore, the solution of this linear program effectively completes the construction at this refinement level.

4.3.4 Implications

While the linear program given in Equation 4.31 characterizes the smallest break size, it is possible to derive bounds on the minimal break size by considering the Conservative Disjoint Image Condition and the Bounded Slope Condition. Independent of the solution of this linear program, the size of ρ is bounded below by $\frac{1}{2}|T_i^k|$.

Claim 11. *The smallest possible break size ρ^* is bounded below by $\frac{1}{2}|T_i^k|$.*

Proof. By the definition of Δ_i^k , the following statements hold:

$$\begin{aligned}
 |\Delta_{ii}^k| &= (\lambda_1 + \lambda_2)|T_i^k| \\
 |\Delta_{ii}^{k+1}| &= (\lambda_1 + \lambda_2)|T_i^{k+1}| \\
 |\Delta_{Ni}^{k+1}| &= \lambda_1|T_N^{k+1}| + \lambda_2|T_i^{k+1}| \\
 |\Delta_{iN}^{k+1}| &= \lambda_1|T_i^{k+1}| + \lambda_2|T_N^{k+1}| \\
 |\Delta_{NN}^{k+1}| &= (\lambda_1 + \lambda_2)|T_N^{k+1}|.
 \end{aligned} \tag{4.32}$$

By Claim 7, the last four of the above Δ_i^{k+1} are contained in Δ_i^k . If they do not intersect, then

$$|\Delta_{ii}^{k+1}| + |\Delta_{Ni}^{k+1}| + |\Delta_{iN}^{k+1}| + |\Delta_{NN}^{k+1}| \leq |\Delta_{ii}^k|. \tag{4.33}$$

Substituting in the above definitions yields

$$2(\lambda_1 + \lambda_2) (|T_i^{k+1}| + |T_N^{k+1}|) \leq (\lambda_1 + \lambda_2)|T_i^k|. \tag{4.34}$$

Divide by $\lambda_1 + \lambda_2$. Using the definition of the gap size, that $\rho = |T_i^k| - |T_i^{k+1}| + |T_N^{k+1}|$,

$$2(|T_i^k| - \rho) \leq |T_i^k|. \quad (4.35)$$

Therefore,

$$\rho \geq \frac{1}{2}|T_i^k|. \quad (4.36)$$

□

Removing more than half of the largest intervals at each refinement level guarantees that the All But One condition will be violated. The size of the segment removed from T_i^k needs to be comparatively large to satisfy the Conservative Disjoint Image Condition, and it needs to be sufficiently large to keep the increase in the slope of ψ^{k+1} controlled, yet if too much of T_i^k is removed during the breaking process, there will be some point no longer contained in all but one of the shifted families. In essence, the Conservative Disjoint Image Condition is too conservative: this condition necessitates the creation of breaks that are large enough so that all the conservative image intervals Δ_i^{k+1} can fit inside, but as a result the space allocated for each of these conservative image intervals is too large for the All But One condition to be satisfied.

This result suggests that when using Fridman's general scheme, it is not possible to enforce all three of the Conservative Disjoint Image Condition, the Bounded Slope Condition, and the All But One Condition, while still resulting in a Lipschitz continuous inner function. Fridman's initial algorithm demonstrates it is possible to enforce the Bounded Slope Condition and the All But One Condition, and even a weak version of Fridman's Disjoint

Image Condition, but not simultaneously the Disjoint Image Condition. This next attempt satisfies the Disjoint Image Condition and the Bounded Slope Condition, but leaves no possibility to maintain the All But One Condition. The Disjoint Image Condition and the All But One condition together, without the Bounded Slope Condition, results in the successful construction of an inner function, but not a Lipschitz inner function. Fridman’s weakened Disjoint Image Condition is too relaxed, while the Conservative Disjoint Image Condition is too restrictive. The exact Disjoint Image Condition from Kolmogorov’s original paper is in a ‘goldilocks’ zone: it allows for enough space between image intervals $\Psi^q(S_{ij}^k)$ to ensure the images of disjoint squares remain disjoint, while it restricts the amount of space they can occupy so that the image intervals do not spread as to overlap as $k \rightarrow \infty$. The Disjoint Image Condition, and no condition stronger, nor any condition weaker, is needed for computing KST inner functions. However, a dynamic construction of the intervals \mathcal{T}^k provides no information that can be exploited to guarantee disjoint images, whereas Sprecher’s approaches to construct Hölder continuous inner functions can exploit the self-similarity in the definition of \mathcal{T}^k to define function values for the function ψ^k that obey the Disjoint Image Condition itself (as exemplified in Appendix C).

I therefore conclude that Fridman’s paper needs to be reexamined, to determine whether it is indeed possible to algorithmically construct a function using the method he outlines. Yet, I emphasize that the conditions listed in Chapter 3 are sufficient conditions for computational KST, and not necessary conditions; if it is somehow possible to use the Disjoint Image Condition directly in the Fridman Strategy, then Fridman’s results may hold.

In the next chapter, I turn to Kahane and Hedberg’s proof of KST, and of their existence proofs that KST is possible with Lipschitz inner functions. The Kahane and Hedberg formu-

lation of KST does not use the Fridman Strategy; an adaptation of this approach provides a new foundation for computing Lipschitz continuous inner functions for KST.

Chapter 5

A Reparameterization Argument

The failure of approaches using the Fridman Strategy to construct Lipschitz continuous inner functions suggested that Fridman's original paper, which relied on the weakened Fridman's Disjoint Image Condition, may be flawed. As this paper supplied the initial proof that KST is possible using Lipschitz continuous inner functions, it is reasonable to suspect that this result is unattainable. Fortunately, Hedberg reformulated KST into a statement on quasi-all $2n + 1$ -tuples of monotonic increasing continuous functions, and Kahane showed that under this restatement of KST, the possibility of using Lipschitz continuous inner functions follows automatically, simply by reparameterizing non-Lipschitz inner functions (Hedberg [19], Kahane [22]). As a result, I reparameterize the Hölder continuous inner function suggested by Sprecher, corrected by Köppen, and analyzed by Braun and Griebel, to create a Lipschitz inner function (Sprecher [42], [43]; Köppen [26]; Braun and Griebel [7]).

In this chapter, I revisit the work of Hedberg and Kahane to prove the existence of Lipschitz continuous inner functions. In Section 5.1, I summarize Hedberg's approach to restate KST, and I describe Kahane's arguments of how this reformulation guarantees that Lipschitz inner functions can be used for KST. In Section 5.2, I describe Köppen's inner function in detail. This function is reparameterized in Section 5.3 to obtain a Lipschitz inner function. Section 5.4 verify that the Hölder continuous inner function remains an inner function even after reparameterization.

5.1 Hedberg and Kahane's Reformulation

Hedberg was the first to reformulate KST relying primarily on the Baire Category Theorem (Hedberg [19]). Kahane later improved on this argument to provide a geometric interpretation of KST (Kahane [22]). In this geometric framework, the existence of Lipschitz continuous inner functions follows naturally. In this section, I motivate Hedberg and Kahane's geometric approach, and I then demonstrate how Lipschitz inner functions are guaranteed in this formulation.

5.1.1 A Geometric Interpretation of KST

In this subsection, I first repeat Kolmogorov's statement of KST, and I then provide Hedberg and Kahane's reformulation of KST. As I compare these statements, I highlight how Hedberg and Kahane enable the interpretation of the inner functions as defining a geometric curve in $2n+1$ -space. Much of this analysis was clearly communicated by Khavinson in his monograph on approximate nomography (Khavinson [23]).

Neither Kolmogorov, Hedberg, nor Kahane utilize Sprecher's reduction. For the rest of this section, I revert to fully indexing the inner and outer functions with both the p and q subscripts.

Recall Kolmogorov's original statement of KST:

Theorem 1.1.1 (KST). *Let $f \in C(\mathbb{I}^n) : \mathbb{I}^n \rightarrow \mathbb{R}$ be a continuous multivariate function.*

There exist univariate continuous functions $\chi_q : \mathbb{I} \rightarrow \mathbb{R}$ and $\psi_{p,q} : \mathbb{R} \rightarrow \mathbb{R}$, where $p = 1, \dots, n$

and $q = 0, \dots, 2n$, such that for all $x = (x_1, \dots, x_p) \in \mathbb{I}^n$,

$$f(x_1, \dots, x_n) = \sum_{q=0}^{2n} \chi_q \left(\sum_{p=1}^n \psi_{p,q}(x_p) \right). \quad (1.1)$$

The functions $\psi_{p,q}$ do not depend on the function f (Kolmogorov [25]).

Also recall from Chapter 2 the original reformulation by Hedberg, also used by Kahane (Hedberg [19]):

Theorem 2.2.2 (Hedberg and Kahane KST). *Let $\Phi \subset C(\mathbb{I})$ be the set of continuous non-decreasing functions on \mathbb{I} such that $\forall \psi \in \Phi$,*

$$\psi(0) = 0 \quad \psi(1) = 1.$$

Let $\lambda_1, \dots, \lambda_n$ be rationally independent positive numbers, such that $\sum_{p=1}^n \lambda_p = 1$. Then, for quasi-all tuples $(\psi_1, \dots, \psi_{2n+1}) \in \Phi^{2n+1}$, it follows that for any $f \in C(\mathbb{I}^n) : \mathbb{I}^n \rightarrow \mathbb{R}$ multivariate continuous function, there exists some continuous univariate function $\chi : \mathbb{I} \rightarrow \mathbb{R}$ such that

$$f(x_1, \dots, x_n) = \sum_{q=0}^{2n} \chi \left(\sum_{p=1}^n \lambda_p \psi_q(x_p) \right).$$

As this statement is about quasi-all functions, I repeat the definition of this term here.

Definition. *Let X be a complete metric space. A property holds for **quasi-all** points in X if it holds on a countable intersection of open dense sets in X .*

I also repeat the definition of integral independence.

Definition. A tuple $\lambda = (\lambda_1, \dots, \lambda_n) \subset \mathbb{R}^n$ is **integrally independent** (rationally independent) if for any rational $x \in \mathbb{Q}^n$ with $x \neq 0$,

$$\langle \lambda, x \rangle \neq 0,$$

where $\langle \cdot, \cdot \rangle$ denotes the ℓ_2 inner product.

Two examples of quasi-all sets are relevant to this discussion. First, quasi-all points on the surface of a unit ball in \mathbb{R}^n have integrally independent coordinates. Second, note that the set Φ in Kahane and Hedberg's reformulation, i.e. the closed subset of $C[0, 1]$ of continuous, monotonic nondecreasing functions, is a complete metric space; quasi-all functions in Φ are strictly monotonic increasing.

Khavinson demonstrates how these two examples combine to construct tuples $(\phi_0, \dots, \phi_{2n}) \in \Phi^{2n+1}$ that uniformly separate regular Borel measures on the unit cube $[0, 1]^n$; for more details, see [23]. These tuples characterize a subspace $Y \subset C[0, 1]$ of outer functions that freely interpolate minimally separated, nowhere dense set of points in $[0, 1]^n$; the definition of freely interpolating functions is given below.

Definition. Let $Y \subset C[0, 1]$ and let $E \subseteq [0, 1]$ be a closed subset. The subspace Y **interpolates freely** on E if for each $h \in C(E)$, there exists some $\chi \in Y$ such that for all $x \in E$,

$$\chi(x) = h(x).$$

Additionally, Y **interpolates freely with constant c** if for each $h \in C(E)$ there is some

$\chi \in Y$ such that for all $x \in E$,

$$\chi(x) = h(x), \quad \text{and} \quad \|\chi\|_{C[0,1]} \leq c \|h\|_{C(E)}.$$

Kahane and Hedberg observed that this property of free interpolation exactly characterizes the outer functions for KST. They prove (and Khavinson restates) the following theorem (Khavinson [23]).

Theorem 5.1.1. *Let $\lambda_1, \dots, \lambda_n$ be integrally independent; ordered, so that $\lambda_1 > \lambda_2 > \dots > \lambda_n$; positive; and sum to one, i.e. $\sum_{p=1}^n \lambda_p = 1$. Let $Y \subset C([0, 1])$ freely interpolate a nowhere dense closed subset $E \subset (0, 1)$ with constant $c < \frac{2n+2}{2n+1}$, and also freely interpolate with the same constant all sets that are given as E with a finite number of points added from $[0, 1]$ to E . Then, quasi-all $2n+1$ -tuples of inner functions $(\varphi_0, \dots, \varphi_{2n}) \in \Phi^{2n+1}$ allow for KST: for any $f \in C([0, 1]^n)$ and any $h \in C(E)$, there exists some $\chi \in Y$ such that*

$$f(x_1, \dots, x_n) = \sum_{q=0}^{2n+1} \chi \left(\sum_{p=1}^n \lambda_p \varphi_q(x_p) \right), \quad (5.1)$$

and for any $x \in E$,

$$\chi(x) = h(x).$$

This theorem readily admits a geometric interpretation. Let $\lambda = (\lambda_1, \dots, \lambda_n)$ be an n -tuple of coefficients from Theorem 5.1.1, and let $\varphi = (\varphi_0, \dots, \varphi_{2n}) \in \Phi^{2n+1}$ be a tuple that can be used for KST. Define a continuous embedding $X : \mathbb{I}^n \rightarrow \mathbb{R}^{2n+1}$ via the $2n+1$

continuous coordinate embeddings $X_q : \mathbb{I}^n \rightarrow \mathbb{R}$ given as

$$X_q(x_p) = \sum_{p=1}^n \lambda_p \varphi_q(x_p).$$

That is,

$$X = (X_0, \dots, X_{2n}).$$

Note that the image of $[0, 1]^n$ under X is compact, i.e. $X([0, 1]^n) = \Gamma \subseteq [0, 1]^{2n+1}$. It can be shown that X is a homeomorphism. Under Theorem 5.1.1, any $f \in C([0, 1]^n)$ can be written as

$$f(x_1, \dots, x_n) = \sum_{q=0}^{2n} \chi(X_q),$$

and the set of outer functions freely interpolate the compact set Γ . This geometric result, i.e. the construction of the homeomorphism X that enables Theorem 5.1.1, improves upon the Menger-Nöbeling Theorem, which implies any compact set in $[0, 1]^n$ can be homeomorphically embedded in $[0, 1]^{2n+1}$, by constructing an embedding that enables this free interpolation property.

Kahane observed that because this homeomorphism acts on quasi-all $2n + 1$ -tuples in Φ^{2n+1} , it suffices to consider the quasi-all set tuples in Φ^{2n+1} that are strictly increasing functions (Kahane [22]). This strict monotonicity guarantees that the functions ϕ_q are rectifiable, thereby providing the possibility of Lipschitz reparameterization. This reparameterization is described in the next subsection.

5.1.2 Guarantee of Lipschitz Inner Functions

According to Kahane, Lipschitz inner functions are guaranteed to exist because any non-Lipschitz KST inner function is rectifiable, and therefore can be reparameterized using a Lipschitz reparameterization. This guarantee builds upon the geometric interpretation of KST, by looking at the inner functions as describing curves in the set Γ , and then providing a Lipschitz reparameterization of these curves.

Recall that quasi-all tuples $(\phi_0, \dots, \phi_{2n}) \in \Phi^{2n+1}$ are strictly monotonic increasing. Khavinson asserts that without loss of generality, a tuple of inner functions for KST is strictly increasing. Fix $q \in \{0, \dots, 2n\}$. Define κ as the parameterized curve of the image of X_q , that is, for $t \in [0, 1]$, define

$$\kappa(t) = X_q = \phi_q(t).$$

The curve κ is a rectifiable curve; the definition of such curves is given below.

Definition (Rectifiable Curve). *Let $\zeta : [a, b] \rightarrow \mathbb{C}$ be a curve. Let \mathbf{P} be the set of all partitions of $[a, b]$; for any $P \in \mathbf{P}$ denote the partition elements as*

$$a = t_0 < t_1 < \dots < t_{\ell-1} < t_\ell = b.$$

The arc length of ζ , denoted by $|\zeta|$ is defined as

$$|\zeta| = \sup_{P \in \mathbf{P}} \sum_{i=1}^{\ell} |\zeta(t_i) - \zeta(t_{i-1})|.$$

*The curve ζ is **rectifiable** if $|\zeta|$ is finite, i.e. the sum above converges.*

A function is rectifiable if its image can be parameterized as a rectifiable curve. In this case, view $\zeta : [a, b] \rightarrow \mathbb{R}$; ζ is rectifiable if

$$|\zeta| = \sup_{P \in \mathbf{P}} \sum_{i=1}^{\ell} \sqrt{(t_i - t_{i-1})^2 + (\zeta(t_i) - \zeta(t_{i-1}))^2} < \infty.$$

The following theorem is the foundation of Kahane's argument.

Theorem 5.1.2. *A curve is rectifiable if and only if it admits a Lipschitz reparameterization.*

Proof. A comprehensive proof is given in Sullivan [46]. □

Kahane makes the following statement about κ , and then concludes the following corollary:

Claim 12. *The curve κ is rectifiable.*

Proof. Since ϕ_q is strictly monotonic increasing, it is of bounded variation, and therefore its image κ is a rectifiable curve. □

Corollary 3. *The curve κ has a Lipschitz reparameterization.*

Define σ as the parameterization of the arc length of κ , rescaled so that $\sigma : [0, 1] \rightarrow [0, 1]$ by dividing by the total arc length of ϕ_q on the interval $[0, 1]$. Khavinson suggests that reparameterization of κ by σ is the Lipschitz reparameterization required to convert the inner function ϕ_q into a Lipschitz inner function; the function $\phi_q \circ \sigma$ describes the same curve κ as given by ϕ_q alone (Khavinson [23]). For a given $f \in C([0, 1]^n)$, it is thus possible to instead consider a corresponding function $F \in C(\sigma([0, 1])^n)$ and represent F using KST

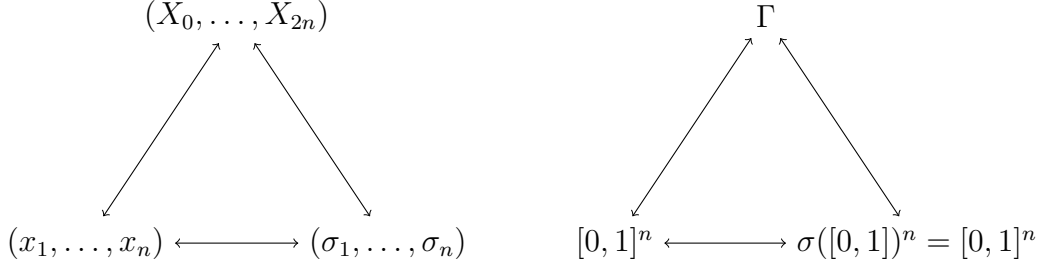


Figure 5.1 : Relationship of homeomorphisms that enable a Lipschitz reparameterization of arbitrary KST inner functions. On the left are the coordinates as they are mapped between each other, and on the right are the spaces characterized by the homeomorphisms in question.

via

$$F(\sigma_1, \dots, \sigma_n) = \sum_{q=0}^{2n} \chi \left(\sum_{p=1}^n \lambda_p \phi_q(\sigma_p) \right).$$

Therefore, given a non-Lipschitz inner function $\hat{\psi}$ that satisfies Sprecher's reduction, it is possible to construct a Lipschitz inner function ψ from $\hat{\psi}$ by considering the inverse map of this reparameterization:

$$\psi = \hat{\psi} \circ \sigma^{-1}.$$

The relationship of the homeomorphisms that enable this representation are summarized in Figure 5.1 (Khavinson [23]).

Approaching this problem computationally requires knowing any (possibly non-Lipschitz) inner function, and I only need to parameterize its arclength to construct a Lipschitz inner function. In Section 5.2, I provide a Hölder continuous inner KST function, and then in Section 5.4 I verify that the Lipschitz reparameterization of this function is indeed an inner function sufficient for KST, in accordance with the theory outlined in this section.

5.2 A Hölder Continuous Inner Function

In this section, I introduce a Hölder continuous inner function for KST. The following function $\widehat{\psi}$ was first proposed by Sprecher in [42] and [43]. It was later edited by Köppen in [26] and analyzed by Braun and Griebel in [7]. After introducing this function, I highlight a handful of properties that will be relevant for the reparameterization argument given by Kahane ([22]).

Fix a radix $\gamma \in \mathbb{N}$. Let $\mathcal{D}^k \subset [0, 1]$ be the numbers whose expansion in base γ terminate at or before the k^{th} digit. Any number $d \in \mathcal{D}^k$ can be expressed as

$$d = i_0.i_1i_2, \dots, i_k.$$

Similarly, set \mathcal{D}^k can be written as

$$\mathcal{D}^k = \left\{ \frac{i}{\gamma^k} : i = 0, \dots, \gamma^k \right\}.$$

Define $\varepsilon = \frac{1}{\gamma(\gamma-1)}$.

Let the function $\beta : \mathbb{N} \rightarrow \mathbb{N}$ be

$$\beta(k) = \frac{n^k - 1}{n - 1}. \tag{5.2}$$

Define the set \mathcal{R}^k as the set of the following intervals:

$$\mathcal{R}^k = \left\{ \left[\frac{i}{\gamma^k}, \frac{i}{\gamma^k} + \frac{\gamma-1}{\gamma^{k+1}} \right] : i = 0, \dots, \gamma^k \right\}.$$

With this definition, the intervals in the set \mathcal{R}^k are disjoint.

The function $\widehat{\psi}$ is defined as the limit of a sequence of functions $\widehat{\psi}^k$. For each $k \in \mathbb{N}$, the function $\widehat{\psi}^k$ is the piecewise linear interpolant of assigned values at points in the set \mathcal{D}^k . For $d_k = i_0.i_1, \dots, i_k \in \mathcal{D}^k$, the function $\widehat{\psi}^k$ is defined as follows:

$$\widehat{\psi}^k(d_k) = \begin{cases} d_k & k = 1 \\ \widehat{\psi}^{k-1}\left(d_k - \frac{i_k}{\gamma^k}\right) + \frac{i_k}{\gamma^{\beta(k)}} & k > 1 \text{ and } i_k \neq \gamma - 1 \\ \frac{1}{2}\left(\widehat{\psi}^k\left(d_k - \frac{1}{\gamma^k}\right) + \widehat{\psi}^{k-1}\left(d_k + \frac{1}{\gamma^k}\right)\right) & k > 1 \text{ and } i_k = \gamma - 1 \end{cases} \quad (5.3)$$

This function $\widehat{\psi}^k$ can be extended from a domain of $[0, 1]$ to all of \mathbb{R} by assigning

$$\widehat{\psi}^k(x) \mapsto \widehat{\psi}^k(x - \lfloor x \rfloor) + \lfloor x \rfloor.$$

With this extension, the function $\widehat{\psi} : [0, 2] \rightarrow \mathbb{R}$ is a Hölder continuous inner function for KST (Braun and Griebel [7]). The proof of this statement is not given here; see Braun and Griebel's paper for a thorough analysis. This function is plotted in Figure 5.2, using code from Appendix E.

Kahane's reparameterization observation will make use of a few properties: that $\widehat{\psi}^k \rightarrow \widehat{\psi}$, that $\widehat{\psi}$ is strictly monotonic increasing, and that $\widehat{\psi}^k$ and $\widehat{\psi}$ are rectifiable. I show each these below.

Claim 13. *The function $\widehat{\psi} = \lim_{k \rightarrow \infty} \widehat{\psi}^k$ is well-defined, where convergence regards the supremum norm.*

Proof. This can be shown by a similar argument to the sketch previously given in Section

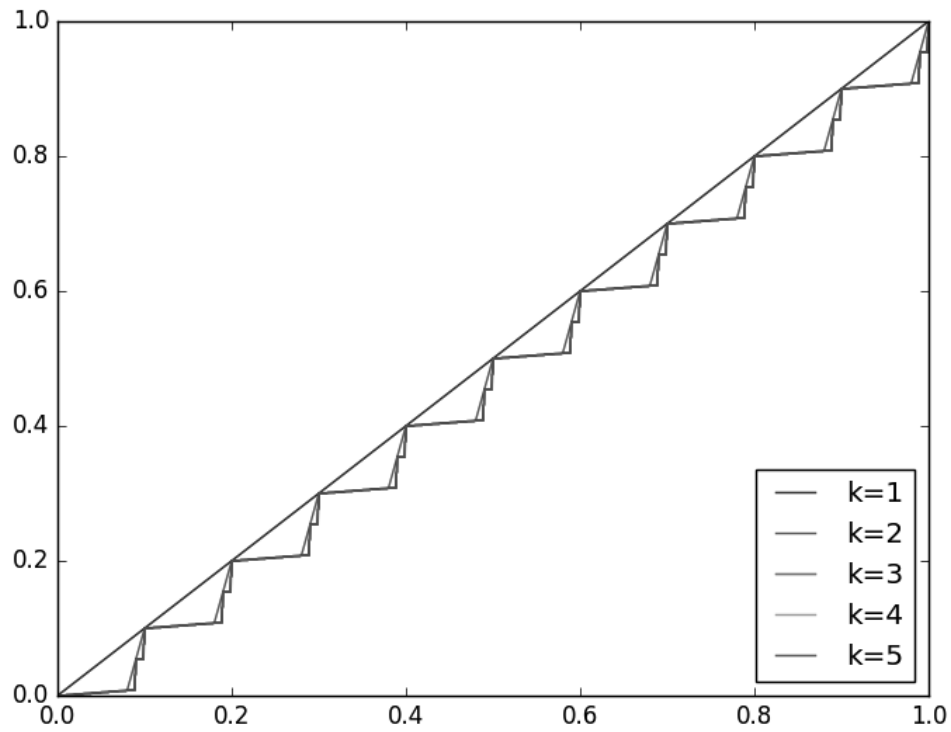


Figure 5.2 : Plots of Köppen's $\hat{\psi}^k$ for first few values of k . Note that the The functions $\hat{\psi}^k$ are self-similar at two alternating scales. For a more detailed analysis of this self-similarity, see [26] and [7].

2.2, describing how Sprecher's inner functions are Hölder continuous even with his reduction from $2n^2 + n$ inner functions to one. A more detailed version of Sprecher's argument is in Appendix C. The complete proof of this statement can be found in Braun and Griebel's analysis of Köppen's inner function (Braun and Griebel [7]); compare to Sprecher's analysis in [38]. \square

I next show the monotonicity of the functions in question.

Claim 14. *The functions $\widehat{\psi}^k$ and $\widehat{\psi}$ are strictly monotonic increasing.*

Proof. As the values of $\widehat{\psi}^k$ strictly increase with $d_k \in \mathcal{D}^k$, and the definition of $\widehat{\psi}^k$ is completed using linear interpolation, the function $\widehat{\psi}^k$ is strictly monotonic increasing.

Since $\widehat{\psi}^k \rightarrow \widehat{\psi}$, the function $\widehat{\psi}$ is guaranteed to be monotonic increasing. It is necessary to show that $\widehat{\psi}$ is strictly monotonic increasing. Fix $x_1, x_2 \in [0, 1]$ such that $x_1 < x_2$. Then, there exists some $k \in \mathbb{N}$ such that there are two numbers $d_{k_1}, d_{k_2} \in \mathcal{D}^k$ where

$$x_1 < d_{k_1} < d_{k_2} < x_2.$$

Then,

$$\widehat{\psi}(x_1) \leq \widehat{\psi}(d_{k_1}) \leq \widehat{\psi}(d_{k_2}) \leq \widehat{\psi}(x_2).$$

Yet, by definition of $\widehat{\psi}$ and $\widehat{\psi}^k$,

$$\begin{aligned} \widehat{\psi}(d_{k_1}) &= \widehat{\psi}^k(d_{k_1}) \\ \widehat{\psi}(d_{k_2}) &= \widehat{\psi}^k(d_{k_2}). \end{aligned} \tag{5.4}$$

Therefore $\widehat{\psi}(d_{k_1}) < \widehat{\psi}(d_{k_2})$, which implies

$$\widehat{\psi}(x_1) < \widehat{\psi}(x_2).$$

□

I finally show that these functions are rectifiable.

Claim 15. *The functions $\widehat{\psi}$ and $\widehat{\psi}^k$ are rectifiable.*

Proof. Since $\widehat{\psi}$ and $\widehat{\psi}^k$ are strictly monotonic increasing, they are of bounded variation, and therefore their images are rectifiable curves. □

Having defined a Hölder continuous inner function, and having highlighted a few of its properties, I turn to discussing the Lipschitz reparameterization of this inner function.

5.3 Reparameterization

Define the function $\widehat{\sigma} : [0, 1] \rightarrow \mathbb{R}$ as to measure the arclength of the function $\widehat{\psi}$. Formally, Let $\mathbf{P}(x)$ be the set of all partitions of the interval $[0, x]$ for $x \leq 1$; any partition $P \in \mathbf{P}(x)$ can be written as $P = \{0 = t_0 < t_1 < \dots < t_{\ell-1} < t_\ell = x\}$. Then, for $\widehat{\psi} : [0, 1] \rightarrow [0, 1]$, define $\widehat{\sigma} : [0, 1] \rightarrow \mathbb{R}$ as

$$\widehat{\sigma}(x) = \sup_{P \in \mathbf{P}(x)} \sum_{i=1}^{\ell} \sqrt{(t_i - t_{i-1})^2 + (\widehat{\psi}(t_i) - \widehat{\psi}(t_{i-1}))^2}.$$

That $\widehat{\psi}$ is rectifiable guarantees that $\widehat{\sigma}$ is well-defined. However, it is of concern that $\widehat{\psi}$ is defined as the limit of functions $\widehat{\psi}^k$. Define $\widehat{\sigma}^k$ as the similarly-defined arclength function for

the function $\widehat{\psi}^k$. It would be convenient if $\widehat{\sigma}^k \rightarrow \widehat{\sigma}$ as $k \rightarrow \infty$; this is stated in the following lemma.

Lemma 3. *Suppose $\widehat{\psi}^k \rightarrow \widehat{\psi}$ uniformly as $k \rightarrow \infty$, and suppose that $\widehat{\psi}^k$ and $\widehat{\psi}$ are continuous and rectifiable. Moreover, assume $\widehat{\psi}^k(d) = \widehat{\psi}(d)$ for $d \in \mathcal{D}^k$. Let $\widehat{\sigma}^k$ measure the accumulated arclength of $\widehat{\psi}^k$, and $\widehat{\sigma}$ measure the accumulated arclength of $\widehat{\psi}$. Then, $\widehat{\sigma}^k \rightarrow \widehat{\sigma}$ pointwise as $k \rightarrow \infty$.*

Proof. For fixed $k \in \mathbb{N}$, note that $\widehat{\sigma}^k$ is exactly equal to a polygonal approximation of the curve $\widehat{\psi}$, using the points $d \in \mathcal{D}^k$ as the vertices of the approximating polygon. By Proposition 1.4 in Sullivan [46], as the maximal spacing between vertices of successive polygonal approximations approaches zero, the length of the polygon approaches the length of the curve it approximates. Therefore, $\widehat{\sigma}^k(x) \rightarrow \widehat{\sigma}(x)$ as $k \rightarrow \infty$. \square

This lemma implies that for the sake of computation, it suffices to consider each computed version of $\widehat{\psi}^k$ and compute its arclength, and use the limit of the arclength functions for $\widehat{\psi}^k$ as the limit of the arclength of $\widehat{\psi}$ itself.

Next, rescale $\widehat{\sigma}$ to create a new function σ , so that

$$\sigma(x) = \frac{\widehat{\sigma}(x)}{\widehat{\sigma}(1)}.$$

Define a new inner function $\psi : [0, 1] \rightarrow [0, 1]$ given by

$$\psi = \widehat{\psi} \circ \sigma^{-1}.$$

With this definition, I make the following claim, as does Khavinson in [23]:

Claim 16. *The reparameterization given by σ is the Lipschitz reparameterization.*

That ψ is Lipschitz is suggested via the following computational argument. The function ψ is computed by taking $\psi^k = \widehat{\psi}^k \circ (\sigma^k)^{-1}$ as the refinement level $k \rightarrow \infty$. The functions ψ^k for $k = 1, \dots, 5$ are plotted in Figure 5.3 below, using code from Appendix E.

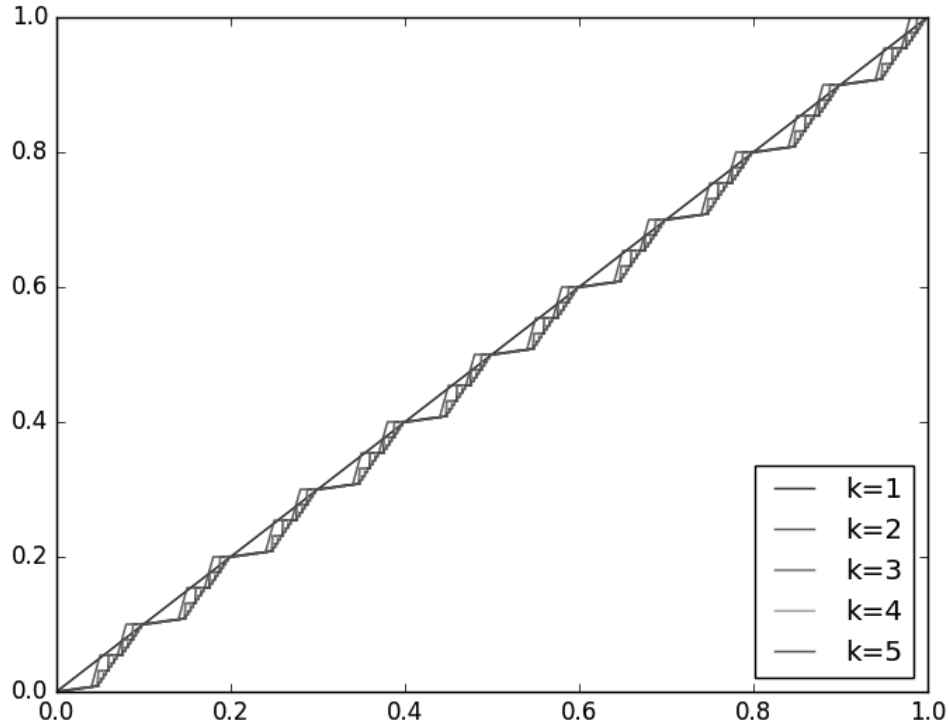


Figure 5.3 : Plots of ψ^k for first few values of k . Note that the slopes appear to all be bounded, meaning that ψ^k is Lipschitz, and that they appear to converge uniformly as $k \rightarrow \infty$.

Figure 5.4 compares the reparameterized function ψ with the original function $\widehat{\psi}$, both constructed through 6 levels of refinement.

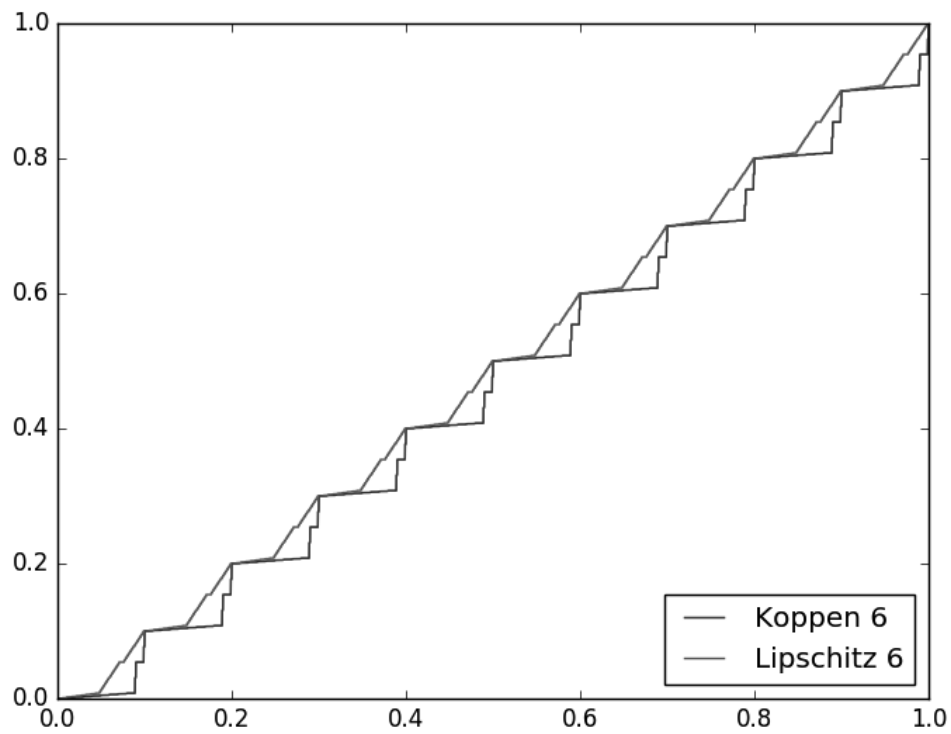


Figure 5.4 : Plot of $\hat{\psi}^6$ and ψ^6 . Note that ψ^6 has a much more controlled slope than $\hat{\psi}^6$. This makes sense, given ψ is Lipschitz continuous whereas $\hat{\psi}$ is not.

Transform the set of intervals \mathcal{R}^k into the set \mathcal{T}^k , given as

$$\mathcal{T}^k = \{\sigma(R) : R \in \mathcal{R}^k\}.$$

With this reparameterization, I claim ψ is Lipschitz inner function for KST, corresponding to the set of intervals \mathcal{T}^k . I verify that ψ , the corresponding functions $\psi^k = \widehat{\psi}^k \circ \sigma^{-1}$, and the sets of intervals \mathcal{T}^k satisfy the Sufficient KST conditions in the next section.

5.4 Verification of Sufficient Conditions for KST

Hedberg and Kahane's proofs, which confirm KST can be done using Lipschitz inner functions, do not make a statement on how the outer function is constructed. However, if the reparameterized, Lipschitz inner function meets the sufficient conditions from Chapter 3, then Kolmogorov's proof of KST can be used for the construction of the outer functions. Combined with a method that implements Kolmogorov's outer function construction, this reparameterization would constitute a complete approach to computing KST representations of arbitrary multivariate continuous functions. In this section, I restate the conditions that this reparameterization needs to meet, and I demonstrate that this function satisfies these conditions. Regarding the Refinement Condition and the All But One Condition, the proofs of these statements remain as future work; computational verification of these conditions is presented instead.

Condition 8 (Refinement). *With successive refinement, the length of intervals $T \in \mathcal{T}^k$ goes to zero uniformly, i.e.*

$$\lim_{k \rightarrow \infty} \max_{T \in \mathcal{T}^k} |T| = 0.$$

Condition 3 (All But One). *For any point $x \in \mathbb{I}$, there are $2n$ values of q such that $x \in T$ for some $T \in \mathcal{T}^{k,q}$.*

Condition 7 (Bounded Slope). *For any two points $x_1, x_2 \in [0, 2]$,*

$$|\psi^k(x_1) - \psi^k(x_2)| \leq (1 - 2^{-k})|x_1 - x_2|.$$

Condition 4 (Monotonicity). *The functions $\psi^{k,q}$ are monotonic increasing. The functions ψ^q are strictly monotonic increasing.*

Condition 2 (Disjoint Image). *For any $S_1, S_2 \in \mathcal{S}^{k,q}$,*

$$\Psi^q(S_1) \cap \Psi^q(S_2) = \emptyset.$$

Claim 17. *The set of intervals \mathcal{T}^k meets the Refinement condition.*

Proof. The completion of the proof remains as future work. An idea for this proof is to use the fact that σ is of bounded variation (as it is strictly monotonic increasing and bounded) in order to show that the size of any interval $T = \sigma(R) \in \mathcal{T}^k$ must get smaller as $k \rightarrow \infty$. The primary concern this proof must address is that the mapping σ is not ‘too bad’, in the sense that σ does not distort sizes of specific intervals too greatly.

This claim can be verified computationally. Using the code presented in Appendix E to compute ψ and σ in the case of $n = 2$, I track the length of the largest interval $\sigma(R)$ for $R \in \mathcal{R}^k$, revealing the maximum sizes of intervals for levels $k = 1, \dots, 8$ in Figure 5.5

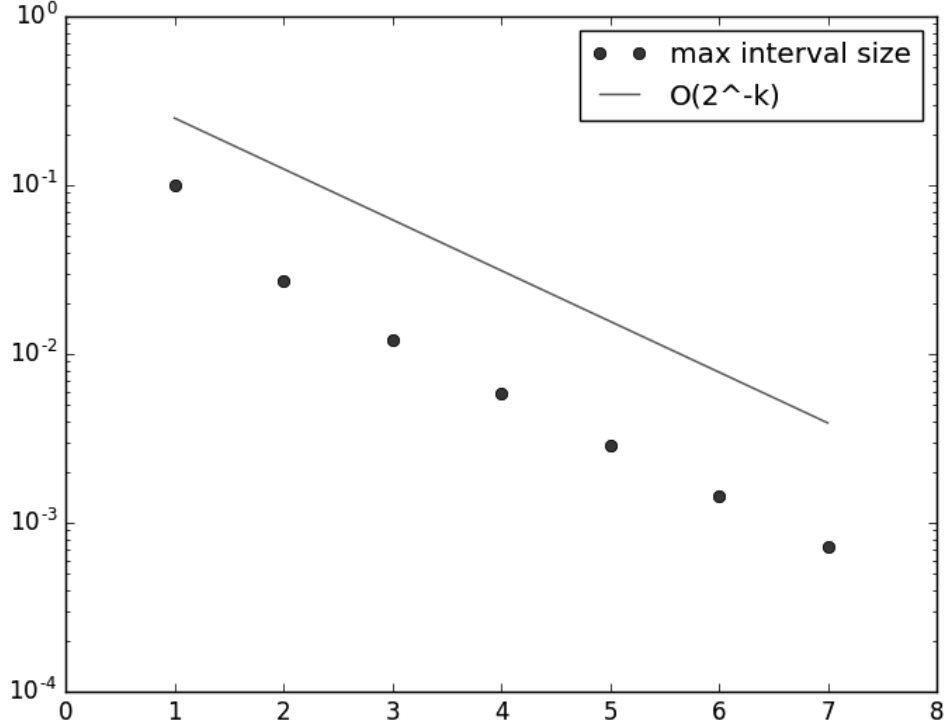


Figure 5.5 : Semilog plot of the largest interval sizes in $\mathcal{T}^k = \sigma(\mathcal{R}^k)$ for the first few values of k . Note that the largest interval size decreases at $O(2^{-k})$; this trend suggests that the Refinement Condition is met.

Figure 5.5 suggests that the size of the largest intervals in \mathcal{T}^k roughly decrease by half at each level of refinement. This refinement rate makes sense in the context of Fridman's paper, where the largest intervals are dynamically broken at each step, thereby creating intervals roughly half the size of at the previous level. This trend suggests that the Fridman Strategy has merit, if enough is known to enforce the Disjoint Image Condition instead of a conservative, stronger condition. More work should be done to determine if a firm comparison can be created between Fridman's approach and Lipschitz reparameterization.

□

Claim 18. *The set of intervals \mathcal{T}^k meets the All But One condition.*

Proof. The completion of this proof remains as future work. This claim should hold true: since σ is of bounded variation, the size of the gaps between intervals in \mathcal{R} cannot be altered too greatly by σ . An ironclad proof of this claim could use the closed form of $\widehat{\psi}$ as given by Braun and Griebel in [7] to define σ in a closed form, and then examine what σ does to the points \mathcal{D}^k for successive refinement levels k .

Again, this claim can be verified computationally. Using the code in Appendix E, I map the shifted families $\mathcal{T}^{k,q}$ in the case $n = 2$ for the first few refinement levels, specifically $k = 1, 2$. Higher values of k would be shown, but the lines are drawn to close together to be distinguished in an image. These intervals are depicted in Figure 5.6 and Figure 5.7.

These figures seem to verify that, at least at the first two levels, the All But One Condition is met. More work should be done to prove this result, using the known action of ψ on the points \mathcal{D}^k to determine the behavior of $\sigma(d)$ for $d \in \mathcal{D}^k$.

□

Claim 19. *The functions ψ^k and $\psi = \lim_{k \rightarrow \infty} \psi^k$ are monotonic increasing.*

Proof. By the proof in Section 5.2, the functions $\widehat{\psi}^k$ and $\widehat{\psi}$ are strictly monotonic increasing. As σ is a continuous homeomorphism of $[0, 1]$ onto itself, it is strictly monotonic increasing, and therefore σ^{-1} is strictly monotonic increasing as well. As the composition of strictly monotonic increasing functions is also strictly monotonic increasing, the functions ψ^k and ψ must satisfy this claim.

□

Claim 20. *The function ψ is Lipschitz Continuous.*

Proof. ψ is precisely the Lipschitz continuous reparameterization of $\widehat{\psi}$.

□

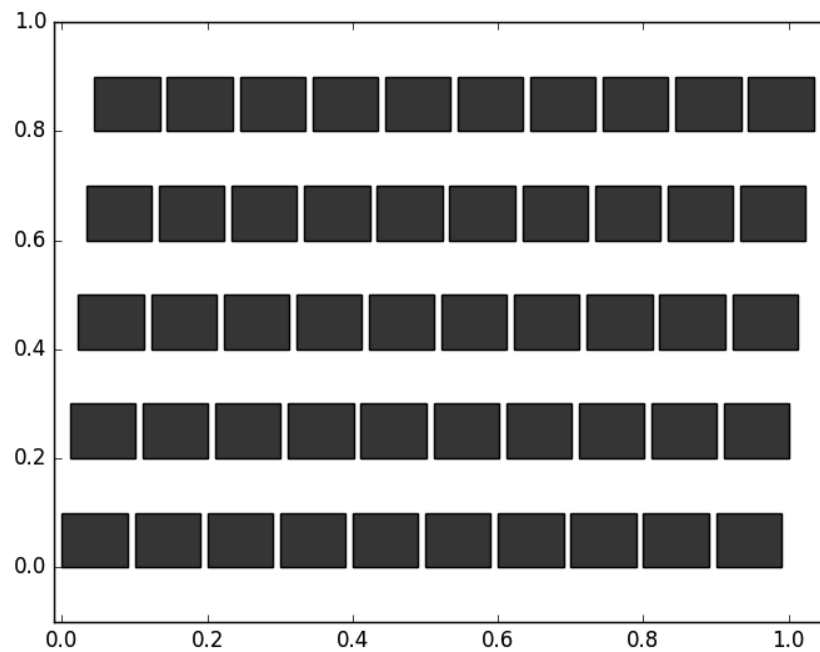
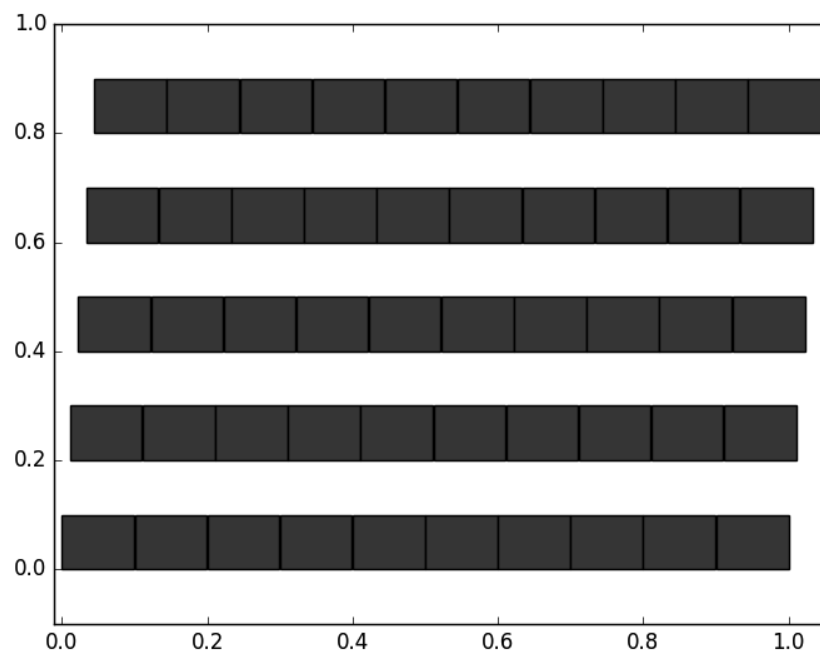
(a) Shifted families of $\mathcal{R}^{1,q}$.(b) Shifted families of $\mathcal{T}^{1,q} = \sigma(\mathcal{R}^{1,q})$.

Figure 5.6 : Shifted families of $\mathcal{T}^k = \sigma(\mathcal{R}^k)$ for $k = 1$. Note that the breaks are located in the same position for each set of intervals, with the breaks of $\sigma(\mathcal{R}^1)$ smaller than that of \mathcal{R}^1 alone; thus, the All But One property is preserved.

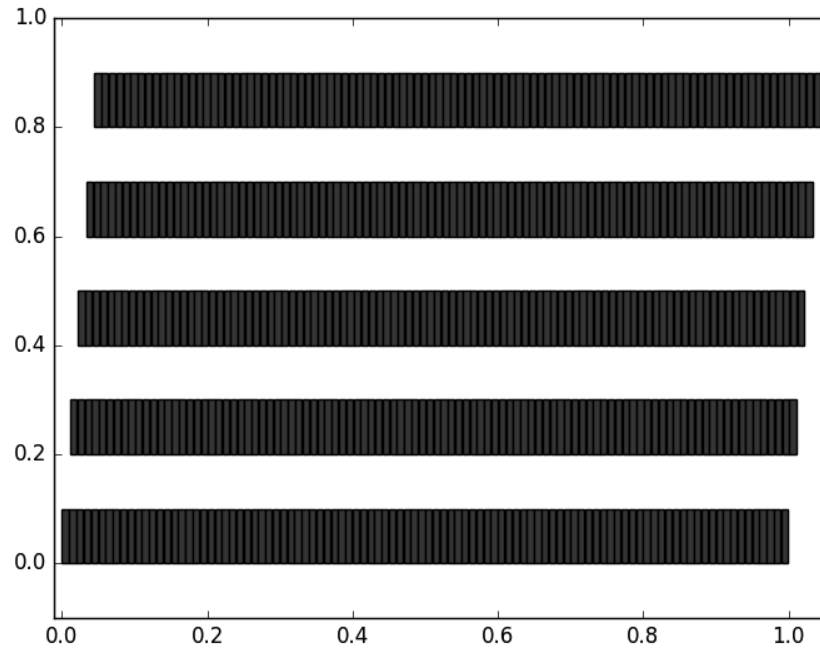
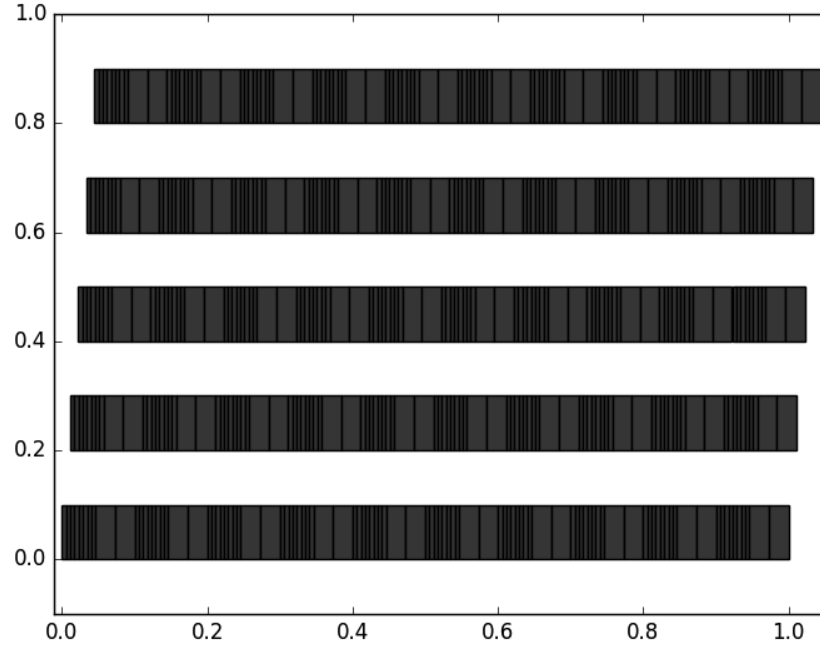
(a) Shifted families of $\mathcal{R}^{2,q}$.(b) Shifted families of $\mathcal{T}^{2,q} = \sigma(\mathcal{R}^{2,q})$.

Figure 5.7 : Shifted families of $\mathcal{T}^k = \sigma(\mathcal{R}^k)$ for $k = 2$. Note that while the breaks are no longer located in the same position for each set of intervals, the sizes of the breaks in \mathcal{T}^2 are smaller than that of \mathcal{R}^2 ; thus, the All But One property is preserved. Additionally, note that the largest intervals are only roughly half the size of the those at the previous level of refinement, demonstrating the comparative sharpness of the $O(2^k)$ bound presented in Figure 5.5.

Claim 21. *The function ψ satisfies the Disjoint Image Condition on \mathcal{T}^k for each $k \in \mathbb{N}$.*

Proof. Fix $k \in \mathbb{N}$. Braun and Griebel's function satisfies the Disjoint Image Condition on \mathcal{R}^k . Therefore, for any $R_1, R_2 \in \mathcal{R}^k$,

$$\widehat{\psi}(R_1) \cap \widehat{\psi}(R_2) = \emptyset.$$

Fix $T_1, T_2 \in \mathcal{T}^k$. There exist $R_1, R_2 \in \mathcal{R}^k$ such that

$$\begin{aligned} T_1 &= \sigma(R_1) \\ T_2 &= \sigma(R_2). \end{aligned} \tag{5.5}$$

Then,

$$\begin{aligned} \psi(T_1) &= \widehat{\psi}(\sigma^{-1}(T_1)) \\ &= \widehat{\psi}(\sigma^{-1}(\sigma(R_1))) \\ &= \widehat{\psi}(R_1), \quad \text{and} \\ \psi(T_2) &= \widehat{\psi}(\sigma^{-1}(T_2)) \\ &= \widehat{\psi}(\sigma^{-1}(\sigma(R_2))) \\ &= \widehat{\psi}(R_2). \end{aligned} \tag{5.6}$$

Therefore,

$$\psi(T_1) \cap \psi(T_2) = \emptyset.$$

□

Appendix A

Statement of Hilbert's 13th Problem

Hilbert, at the second International Congress of Mathematicians in Paris, presented this problem as the 13th of his 23 famous problems. These problems were later published by the American Mathematical Society. The complete text of the problem is below (Hilbert [20]).

13. Impossibility of the solution of the general equations of the 7th degree by means of functions of only two arguments.

Nomography deals with the problem: to solve equations by means of drawings of families of curves depending on an arbitrary parameter. It is seen at once that every root of an equation whose coefficients depend upon only two parameters, that is, every function of two independent variables, can be represented in manifold ways according to the principle lying at the foundation of nomography. Further, a large class of functions of three or more variables can evidently be represented by this principle alone without the use of variable elements, namely all those which can be generated by forming first a function of two arguments, then equating each of these arguments to a function of two arguments, next replacing each of those arguments in their turn by a function of two arguments, and so on, regarding as admissible any finite number of insertions of functions of two arguments. So, for example, every rational function of any number of argu-

ments belongs to this class of functions constructed by nomographic tables; for it can be generated by the processes of addition, subtraction, multiplication and division and each of these processes produces a function of only two arguments. One sees easily that the roots of all equations which are solvable by radicals in the natural realm of rationality belong to this class of functions; for here the extraction of roots is adjoined to the four arithmetical operations and this, indeed, presents a function of one argument only. Likewise the general equations of the 5th and 6th degrees are solvable by suitable nomographic tables; for, by means of Tschirnhausen transformations, which require only extraction of roots, they can be reduced to a form where the coefficients depend upon two parameters only.

Now it is probable that the root of the equation of the seventh degree is a function of its coefficients which does not belong to this class of functions capable of nomographic construction, i. e., that it cannot be constructed by a finite number of insertions of functions of two arguments. In order to prove this, the proof would be necessary that the equation of the seventh degree $x^7 + xf^6 + yf^2 + zf^3 + 1 = 0$ is not solvable with the help of any continuous functions of only two arguments. I may be allowed to add that I have satisfied myself by a rigorous process that there exist analytical functions of three arguments x, y, z which cannot be obtained by a finite chain of functions of only two arguments. By employing auxiliary movable elements, nomography succeeds in constructing functions of more than two arguments, as d'Ocagne has recently proved in the case of the equation of the 7th degree.

Appendix B

Proof of the Kolmogorov Superposition Theorem

The Kolmogorov Superposition Theorem (KST) was first given by Kolmogorov in 1957 [25].

Theorem 1.1.1 (KST). *Let $f \in C(\mathbb{I}^n) : \mathbb{I}^n \rightarrow \mathbb{R}$ be a continuous multivariate function. There exist univariate continuous functions $\chi_q : \mathbb{I} \rightarrow \mathbb{R}$ and $\psi_{p,q} : \mathbb{R} \rightarrow \mathbb{R}$, where $p = 1, \dots, n$ and $q = 0, \dots, 2n$, such that for all $x = (x_1, \dots, x_n) \in \mathbb{I}^n$,*

$$f(x_1, \dots, x_n) = \sum_{q=0}^{2n} \chi_q \left(\sum_{p=1}^n \psi_{p,q}(x_p) \right). \quad (1.1)$$

The functions $\psi_{p,q}$ do not depend on the function f (Kolmogorov [25]).

The proof of this theorem requires the construction of squares $\mathcal{S}^{k,q}$ and functions $\psi_{p,q}$ that meet the following two conditions:

Condition 1 (More Than Half). *For any $x \in \mathbb{I}^n$, there are $n + 1$ values for q such that $x \in S$ for some square $S \in \mathcal{S}^{k,q}$.*

Condition 2 (Disjoint Image). *For any $S_1, S_2 \in \mathcal{S}^{k,q}$,*

$$\Psi^q(S_1) \cap \Psi^q(S_2) = \emptyset.$$

These conditions are explained more extensively in Section 3.1. Suppose systems of squares $\mathcal{S}^{k,q}$ and functions $\psi_{p,q}$ meet the More Than Half Condition and the Disjoint Image Condition. With these conditions met, Kolmogorov completes an iterative construction of the outer functions χ_q using approximates χ_q^k . These functions χ_q^k approximate f on the image of each square $S \in \mathcal{S}^{k,q}$ under the map Ψ_q . As $k \rightarrow \infty$, the functions $\chi_q^k \rightarrow \chi_q$ uniformly.

Proof. Let the shift index q range from $q = 0, \dots, 2n$. At each refinement level k , let the index i range from $i = 1, \dots, (9n)^k + 1$. For each index i , define the interval $T_i^{k,q} \subset \mathbb{I}$ as

$$T_i^{k,q} = \left[\frac{1}{(9n)^k} \left(i - 1 - \frac{q}{3n} \right), \frac{1}{(9n)^k} \left(i - \frac{1}{3n} - \frac{q}{3n} \right) \right]. \quad (\text{B.1})$$

Define the set $\mathcal{T}^{k,q}$ as

$$\mathcal{T}^{k,q} = \{T_i^{k,q} : i = 1, \dots, (9n)^k + 1\}. \quad (\text{B.2})$$

This definition is analogous to that in Equation 3.2, where $\varepsilon = \frac{1}{3n}$. This set $\mathcal{T}^{k,q}$ satisfies the All But One Condition, which implies the More Than Half Condition by the Pigeon Hole Principle.

Condition 3 (All But One). *For any point $x \in \mathbb{I}$, there are $2n$ values of q such that $x \in T$ for some $T \in \mathcal{T}^{k,q}$.*

As the set \mathcal{T}^k is constructed (and thus $\mathcal{T}^{k,q}$ and $\mathcal{S}^{k,q}$), the inner functions $\psi_{p,q}$ are defined concurrently, using Lemma 4 and Lemma 5.

Lemma 4. *Let k range the natural numbers \mathbb{N} . Let i enumerate the intervals in the set $\mathcal{T}^{k,q}$, and let $\mathbf{i} = (i_1, \dots, i_n)$ be a multiindex that enumerates the squares in the set $\mathcal{S}^{k,q} = \prod_{p=1}^n \mathcal{T}^{k,q}$.*

There exist constants ϵ_k and $\psi_i^{k,p,q}$ such that the following conditions hold:

1. $\psi_i^{k,p,q} < \psi_{i+1}^{k,p,q} \leq \psi_i^{k,p,q} + \frac{1}{2^k}$.
2. *Choose $T_i^{k,q} \in \mathcal{T}^{k,q}$ and $T_{i'}^{k+1,q} \in \mathcal{T}^{k+1,q}$. If $T_i^{k,q}$ and $T_{i'}^{k+1,q}$ do not intersect, then*

$$\psi_i^{k,p,q} \leq \psi_{i'}^{k+1,p,q} \leq \psi_i^{k,p,q} + \epsilon_k - \epsilon_{k+1}.$$
3. *For fixed q and k , the closed intervals $\Delta_{\mathbf{i}}^{k,q} = \left[\sum_{p=1}^n \psi_{i_p}^{k,p,q}, \sum_{p=1}^n \psi_{i_p}^{k,p,q} + n\epsilon_k \right]$ are pairwise disjoint.*
4. $\epsilon_k \leq \frac{1}{2^k}$.

The values assigned in Lemma 4 directly define functions $\psi_{p,q}$:

Lemma 5. *Let values $\psi_i^{k,p,q}$ and ϵ_k satisfy Lemma 4. For each $T_i^{k,q} \in \mathcal{T}^{k,q}$ and for each $x \in T_i^{k,q}$, define $\psi_{p,q}(x)$ pointwise so that for all $k \in \mathbb{N}$,*

$$\psi_i^{k,p,q} \leq \psi_{p,q}(x) \leq \psi_i^{k,p,q} + \epsilon_k.$$

The function $\psi_{p,q}$ is well-defined, continuous, and monotonic increasing.

Lemmas 4 and Lemma 5 together prove that Kolmogorov's functions $\psi_{p,q}$ meet the Disjoint Image Condition. Kolmogorov does not define the constants $\psi_i^{k,p,q}$, nor does he give any direction towards the systematic choice these values, but he does state that the assignment of values in accordance with Lemma 4 can be accomplished by induction. Arnold completes

the first few steps of this process for $n = 2$, as does Tikhomirov (Arnold [2], Tikhomirov [49]).

The proof continues as follows. Suppose systems of squares $\mathcal{S}^{k,q}$ and functions $\psi_{p,q}$ are constructed as to meet the More Than Half Condition and the Disjoint Image Condition.

The functions χ_q are constructed as

$$\chi_q = \lim_{r \rightarrow \infty} \chi_q^r,$$

where $\chi_q^0 \equiv 0$ and χ_q^r is defined by induction on $r \in \mathbb{N}$. This induction index r relates to refinement level $k_r \in \mathbb{N}$ from the construction of the system of squares $\mathcal{S}^{k,q}$. Define

$$f_r(x) = \sum_{q=0}^{2n} \chi_q^r(\Psi^q(x)), \quad M_r = \|f - f_r\|_\infty.$$

For the base case $r = 0$, set $f_0 \equiv 0$ and $M_0 = \|f\|_\infty$.

For the inductive step, suppose a continuous function χ_q^{r-1} has been constructed, and in the inductive process an index $k_{r-1} \in \mathbb{N}$ and a continuous function f_{r-1} have been defined. Let $\omega(S)$ equal the oscillation of $f - f_{r-1}$ on square $S \in \mathcal{S}^{k_r,q}$. Choose $k_r \in \mathbb{N}$ such that for all $S \in \mathcal{S}^{k_r,q}$,

$$\omega(S) \leq \frac{1}{2n+2} M_{r-1}. \quad (\text{B.3})$$

Let $\xi \in S$ denote the lower left corner of S . For $y \in \Psi^q(S) \subset \mathbb{R}$, define

$$\chi_q^r(y) = \chi_q^{r-1}(y) + \frac{1}{n+1} [f(\xi) - f_{r-1}(\xi)]. \quad (\text{B.4})$$

Rearrange Equation B.4 and take the norm of each side; then, for χ_q^r, χ_q^{r-1} restricted to $\Psi^q(S)$,

$$\|\chi_q^r - \chi_q^{r-1}\|_\infty \leq \frac{1}{n+1} M_{r-1}. \quad (\text{B.5})$$

Outside of $\Psi^q(S)$, define χ_q^r as to maintain continuity and so that $\|\chi_q^r - \chi_q^{r-1}\|_\infty \leq \frac{1}{n+1} M_{r-1}$.

Such a definition of χ_q^r is guaranteed to exist by the Tietze Extension Theorem.

Fix $x \in \mathbb{I}^n$. Recall from the definition of f_r ,

$$\begin{aligned} f_{r-1}(x) &= \sum_{q=0}^{2n} \chi_q^{r-1}(\Psi^q(x)) \\ f_r(x) &= \sum_{q=0}^{2n} \chi_q^r(\Psi^q(x)). \end{aligned} \quad (\text{B.6})$$

Adding and subtracting f_{r-1} to $f - f_r$ yields the relation

$$\begin{aligned} f(x) - f_r(x) &= f(x) - f_r(x) + \left(f_{r-1}(x) - \sum_{q=0}^{2n} \chi_q^{r-1}(\Psi^q(x)) \right) \\ &= f(x) - f_{r-1}(x) - \sum_{q=0}^{2n} \chi_q^r(\Psi^q(x)) - \chi_q^{r-1}(\Psi^q(x)). \end{aligned} \quad (\text{B.7})$$

The sum in Equation B.7 is split in relation to More Than Half Condition. Denote $\mathcal{Q} = \{0, \dots, 2n\}$, and define the sets

$$\begin{aligned} \mathcal{Q}_1 &= \{q \in \mathcal{Q} \mid x \in S^q \text{ for some } S^q \in \mathcal{S}^{k_r, q}\} \\ \mathcal{Q}_2 &= \mathcal{Q} \setminus \mathcal{Q}_1. \end{aligned} \quad (\text{B.8})$$

By More Than Half Condition, $|\mathcal{Q}_1| = n+1$, $|\mathcal{Q}_2| = n$. There are different sup norm bounds for $\chi_q^{k_r}$ depending on whether $q \in \mathcal{Q}_1$ or $q \in \mathcal{Q}_2$. For $q \in \mathcal{Q}_1$, let S^q be the square in $\mathcal{S}^{k_r, q}$

that contains the point x . Then,

$$\begin{aligned}\chi_q^r(\Psi^q(x)) - \chi_q^{r-1}(\Psi^q(x)) &= \frac{1}{n+1} [f(\xi) - f_{r-1}(\xi)] \\ &= \frac{1}{n+1} [f(x) - f_{r-1}(x)] + \frac{\omega}{n+1},\end{aligned}\tag{B.9}$$

where ω is given by the expression

$$\omega = (f(\xi) - f_{r-1}(\xi)) - (f(x) - f_{r-1}(x)).\tag{B.10}$$

By the above Equation B.3,

$$|\omega| \leq \omega(S^q) = \frac{1}{2n+2} M_{r-1}.$$

For $q \in \mathcal{Q}_2$, recall the prior estimate in Equation B.5,

$$\|\chi_q^r - \chi_q^{r-1}\|_\infty \leq \frac{1}{n+1} M_{r-1}.$$

Split the sum in Equation B.7, and substitute in Equation B.9 in the sum over \mathcal{Q}_1 . Then,

$$\begin{aligned}
f(x) - f_r(x) &= f(x) - f_{r-1}(x) - \sum_{q=0}^{2n} \chi_q^r(\Psi^q(x)) - \chi_q^{r-1}(\Psi^q(x)) \\
&= f(x) - f_{r-1}(x) - \sum_{q \in \mathcal{Q}_1} \left[\chi_q^r(\Psi^q(x)) - \chi_q^{r-1}(\Psi^q(x)) \right] \\
&\quad - \sum_{q \in \mathcal{Q}_2} \left[\chi_q^r(\Psi^q(x)) - \chi_q^{r-1}(\Psi^q(x)) \right] \\
&= f(x) - f_r(x) - \sum_{q \in \mathcal{Q}_1} \left[\frac{1}{n+1} [f(x) - f_{r-1}(x)] + \frac{\omega}{n+1} \right] \\
&\quad - \sum_{q \in \mathcal{Q}_2} \left[\chi_q^r(\Psi^q(x)) - \chi_q^{r-1}(\Psi^q(x)) \right] \\
&= - \sum_{q \in \mathcal{Q}_1} \frac{\omega}{n+1} - \sum_{q \in \mathcal{Q}_2} \left[\chi_q^r(\Psi^q(x)) - \chi_q^{r-1}(\Psi^q(x)) \right].
\end{aligned}$$

Take the sup norm of the above equation:

$$\begin{aligned}
\|f - f_r\|_\infty &\leq \frac{1}{n+1} \sum_{q \in \mathcal{Q}_1} |\omega| + \sum_{q \in \mathcal{Q}_2} \|\chi_q^r - \chi_q^{r-1}\|_\infty \\
&\leq \frac{1}{2n+2} M_{r-1} + \frac{n}{n+1} M_{r-1} \\
&= \left(\frac{2n+1}{2n+2} \right) M_{r-1}.
\end{aligned}$$

Therefore,

$$\begin{aligned}
M_r &\leq \left(\frac{2n+1}{2n+2} \right) M_{r-1} \\
M_r &\leq \left(\frac{2n+1}{2n+2} \right)^r M_0.
\end{aligned} \tag{B.11}$$

Since for all $0 \leq q \leq 2n$,

$$\|\chi_q^r - \chi_q^{r-1}\|_\infty \leq \frac{1}{n+1} M_{r-1},$$

the sequence $\{\chi_q^r\}_{r \in \mathbb{N}}$ is a Cauchy sequence and therefore converges. Therefore, the function $\chi_q = \lim_{r \rightarrow \infty} \chi_q^r$ is well-defined and continuous. Since $M_r \rightarrow 0$ as $r \rightarrow \infty$, $f_r \rightarrow f$. Therefore,

$$f(x) = \lim_{r \rightarrow \infty} f_r(x) = \lim_{r \rightarrow \infty} \sum_{q=0}^{2n} \chi_q^r(\Psi^q(x)) = \sum_{q=0}^{2n} \chi_q(\Psi^q(x)). \quad (\text{B.12})$$

Thus the proof of KST is complete. □

Appendix C

Three Aspects of Sprecher's KST Reformulation

Sprecher stated the following improvement of the Kolmogorov Superposition Theorem in 1965 [38]:

Theorem C.0.1 (KST-Sprecher). *Let $f \in C(\mathbb{I}^n) : \mathbb{I}^n \rightarrow \mathbb{R}$ be a continuous multivariate function. Let $\lambda_1, \dots, \lambda_n$ be integrally independent positive real numbers. Fix $\widehat{\varepsilon} \in (0, \frac{1}{2n}]$. There exist a rational number $\varepsilon \leq \widehat{\varepsilon}$ and univariate continuous functions $\chi_q : \mathbb{I} \rightarrow \mathbb{R}$ and $\psi : [0, 2] \rightarrow \mathbb{R}$, where $q = 0, \dots, 2n$, such that for all $x = (x_1, \dots, x_p) \in \mathbb{I}^n$,*

$$f(x_1, \dots, x_n) = \sum_{q=0}^{2n} \chi_q \left(\sum_{p=1}^n \lambda_p \psi(x_p + q\varepsilon) \right). \quad (\text{C.1})$$

The function ψ , scalars $\lambda_1, \dots, \lambda_n$, and constant ε do not depend on the function f in question. Additionally, the function ψ can be constructed to be Hölder continuous with exponent $\log_{2n+2}(2)$ (Sprecher [38]).

As the proof of Sprecher's refinement requires a non-trivial introduction of notation and bookkeeping, the proof of Theorem C.0.1 is not repeated here. Instead, the following aspects of this proof will be highlighted here in substantial detail:

1. Construction of ψ and $\mathcal{T}^{k,q}$;
2. Verification that ψ satisfies the Disjoint Image Condition;

3. Justification of ψ being Hölder continuous.

For the curious reader, the complete proof can be found in [38].

C.1 Construction of ψ and $\mathcal{T}^{k,q}$

First is the construction of $\mathcal{T}^{k,q}$ for each refinement level $k \in \mathbb{N}$ and shift index $q \in \{0, \dots, 2n\}$. Fix an integer $\gamma > 2n + 1$; WLOG choose $\gamma = 2n + 2$. This constant γ is the radix of this construction, in sense that each refinement level concerns itself with intervals with left endpoints whose decimal expansions terminate in base γ . Recall the designation of the ‘prototype set’ of intervals \mathcal{T}^k , where the shift index q is dropped. Each interval $T_i^k \in \mathcal{T}^k$ is given as

$$T_i^k = \left[\frac{i}{\gamma^k}, \frac{i}{\gamma^k} + \delta_k \right], \quad (\text{C.2})$$

where

$$\delta_k = \left(\frac{\gamma - 2}{\gamma - 1} \right) \frac{1}{\gamma^k}. \quad (\text{C.3})$$

In this way, the intervals in \mathcal{T}^k are refined self-similarly by a factor of γ between each refinement level. This compares to the factor of $\frac{1}{9n}$ in Kolmogorov’s construction. The left endpoints of the intervals are exactly the rational numbers whose decimal expansions in base γ terminate by the k^{th} decimal place. It is only a moderate amount of bookkeeping to show that, with a correctly chosen ε value, the sets $\mathcal{T}^{k,q}$ satisfy the More Than Half Condition.

Values of ψ are determined concurrently while \mathcal{T}^k is constructed. At refinement level k , the values that ψ obtains are fixed at the left endpoints for each $T_i^k \in \mathcal{T}^k$. As k increases towards infinity, a dense set of values is ascribed to ψ , and the definition of ψ is completed

by continuously connecting between the points that have already received values. While the intervals refine following γ^k , the function values scale according to a factor of γ^{β_k} , where the sequence $\{\beta_k\}_{k \in \mathbb{N}}$ is chosen inductively so that given β_k , the value of $\gamma^{-\beta_{k+1}}$ is sufficiently small:

$$\gamma^{-\beta_{k+1}} < \gamma^{-\beta_k-1} \min_{\substack{h \in \{-\gamma^{\beta_k}, \dots, -1, 0, 1, \dots, \gamma^{\beta_k}\}^n \\ h \neq 0}} \left| \sum_{p=1}^n h_p \lambda_p \right|. \quad (\text{C.4})$$

How these function values are chosen with respect to this scaling rate is discussed in the next subsection.

C.2 Verification of the Disjoint Image Condition

Once the values for β_k have been defined, it is possible to fix the range of ψ on T_i^k to stay within certain bounds. Similar to how the left endpoints of T_i^k were expanded in base γ , denote the values for the corresponding possible image intervals as follows:

$$H_j^k = \left[\frac{j}{\gamma^{\beta_k}}, \frac{j}{\gamma^{\beta_k}} + (\gamma - 2) \sum_{\ell \in \mathbb{N}} \gamma^{-\beta_k + \ell} \right]. \quad (\text{C.5})$$

These intervals are only *possible* image intervals, as for a given refinement level k , there are many more intervals H_j^k than intervals T_i^k . Sprecher describes an injection $i \mapsto j(i)$ that matches indices $j = j(i)$ for the image intervals to the indices i of the domain decomposition intervals. With this correspondance, for any $x \in T_i^k$,

$$\psi(x) \in H_{j(i)}^k.$$

Let the index $\mathbf{i} = (i_1, \dots, i_n)$ be the n -fold Cartesian product of indices of intervals in \mathcal{T}^k , and define the index $J(\mathbf{i}) = (j(i_1), \dots, j(i_n))$ denote the vector of corresponding image intervals.

For each $J(\mathbf{i})$, define the intervals $\hat{H}_{J(\mathbf{i})}^k$ as

$$\hat{H}_{J(\mathbf{i})}^k = \left[\sum_{p=1}^n \lambda_p \frac{j(i_p)}{\gamma^{\beta_k}} + \lambda_p \frac{q}{n}, \sum_{p=1}^n \lambda_p \frac{j(i_p)}{\gamma^{\beta_k}} + \lambda_p \frac{q}{n} + \left((\gamma - 2) \sum_{\ell \in \mathbb{N}} \gamma^{-\beta_k + \ell} \right) \sum_{p=1}^n \lambda_p \right]. \quad (\text{C.6})$$

These intervals relate the square $S_{\mathbf{i}}^k = \prod_{p=1}^n T_{i_p}^k$ to the function $\Psi^q = \sum_{p=1}^n \lambda_p \psi(x_p + q\varepsilon)$ similar to how the intervals $H_{j(i)}^k$ relate the individual intervals T_i^k to ψ . For $x \in S_{\mathbf{i}}^k$,

$$\Psi^q(x) \in \hat{H}_{J(\mathbf{i})}^k.$$

This is the basis of verification that the constructed ψ function satisfies the Disjoint Image Condition, although slightly more work is needed to show that the $\hat{H}_{J(\mathbf{i})}^k$ intervals are disjoint. Verification that these images are disjoint uses the assumption that the coefficients $\lambda_1, \dots, \lambda_p$ are integrally independent. More accurately, it uses the guarantee that for any scalar $\alpha \neq 0$, the value $\alpha \lambda_p$ does not terminate in a base γ expansion. This property is necessary for this specific proof of the Disjoint Image Condition; if another proof were supplied that did not make use of the property, then the integral independence restriction on the λ_p values could be relaxed.

In comparison with Kolmogorov's proof, Sprecher's intervals $\hat{H}_{J(\mathbf{i})}^k$ correspond to Kolmogorov's intervals $\Delta_{\mathbf{i}}^{k,q}$ as seen in Appendix B, Lemma 4.

C.3 Justification of Hölder Continuity

Let the inner function ψ and intervals \mathcal{T}^k be defined by the process sketched above. Consider the following lemma.

Lemma 6. *For any two points $x_1, x_2 \in \mathbb{I}$, with $x_1 \neq x_2$, there exists some refinement level k such that the two points are separated by exactly one gap between intervals belonging to \mathcal{T}^k , or by at least one interval and no more than $\gamma - 1$ intervals belonging to \mathcal{T}^k .*

This lemma follows from the regular distribution of intervals at refinement level k and by the fact that the diameter of the intervals is the same for all intervals and decreases uniformly in k . For a detailed proof of this lemma, see [38]. Sprecher uses this lemma to bound the distance between the two points x_1 and x_2 . For any such two points x_1 and x_2 , there is some $k \in \mathbb{N}$ such that the following bounds hold:

$$\begin{aligned} |x_1 - x_2| &\geq 2^{\gamma-k-1} \\ |\psi(x_1) - \psi(x_2)| &\leq (\gamma + 3)2^{-k-1}. \end{aligned} \tag{C.7}$$

Set $c = (\gamma + 3)2^{-\alpha}$ and $\alpha = \log_\gamma(2)$. Then,

$$\begin{aligned}
 |\psi(x_1) - \psi(x_2)| &\leq (\gamma + 3)2^{-(k+1)} \\
 &= (c2^\alpha)2^{-(k+1)} \\
 &= (c2^\alpha)(\gamma^\alpha)^{-(k+1)} && \text{by definition of } \alpha \\
 &= c2^\alpha(\gamma^{-(k+1)})^\alpha && \text{(C.8)} \\
 &\leq c(2^{\gamma-(k+1)})^\alpha && \text{since } \gamma \geq 2n + 2 \\
 &\leq c|x_1 - x_2|^\alpha.
 \end{aligned}$$

Therefore, ψ is Hölder continuous with exponent $\log_\gamma(2)$.

Appendix D

Code: The Fridman Strategy

The following code implements the Fridman Strategy, as described in Chapter 4. The code was written in Python, and beyond the standard packages, it makes use of the `IntervalTree` and `mpmath` packages as well.

```

1  import random
2  import matplotlib.pyplot as plt
3  import matplotlib.patches as patches
4  import mpmath as mp
5  import argparse
6  import sys
7  from operator import attrgetter
8  import numpy as np
9  from mpl_toolkits.mplot3d import Axes3D
10 from intervaltree import Interval, IntervalTree
11 import time
12 import itertools as itt
13
14 # Code for implementing Lipschitz Inner Function Algorithm, using IntervalTree
   structure to represent system of Towns
15 # Breaks all of one level at once, first by solving for the holes and plugs
   and then breaking to create gaps
16
17 parser = argparse.ArgumentParser(
18     description = 'Sprecher Lipschitz Functions ',
19     epilog = 'For more information, see Sprecher (1972), An Improvement in the
   Superposition Theorem of Kolmogorov ',
20     formatter_class = argparse.ArgumentDefaultsHelpFormatter)
21 parser.add_argument( '--dim', type=int, default=2,
22                     help='The spatial dimension')
23 parser.add_argument( '--J', type=int, default=8,
24                     help='Number of iterations')
25 parser.add_argument( '--break-ratio', type=float, default=0.5,
26                     help='Reduction in size of largest town, at each step')
27 parser.add_argument( '--draw', type=int, default=0,

```



```

28             help='Use --draw1 to draw the intervals after every
                iteration, and --draw2 to draw only the end result')
29 parser.add_argument('--plot', type=int, default=0,
30                     help='Use --plot1 to plot the resulting function')
31 parser.add_argument('--Phi', type=int, default=0,
32                     help="Full Psi plot")
33 parser.add_argument('--prec', type=int, default = 2**8,
34                     help='mp.mpf precision')
35 parser.add_argument('--verbose', type=int, default=0,
36                     help='Prints diagnostic output to screen')
37 parser.add_argument('--summary', type=int, default = 1,
38                     help='Summary of town and function construction displayed
                after each iteration')
39 args = parser.parse_args()
40
41 class Town(object):
42     def __init__(self, start, end, val, nv, birth, parent):
43         self.end = end
44         self.start = start
45         self.length = mp.mpf(str(end-start))
46         self.center = start + self.length/2.0
47         self.val = val
48         self.vleft = val
49         self.vright = val
50         self.nextval = nv
51         self.birth = birth
52         self.parent = parent
53         self.children = []
54     def __lt__(self, other):
55         return self.start < other.start
56     def __eq__(self, other):
57         return (type(self) == type(other)) and (self.start == other.start) and
            (self.end == other.end)
58     def __hash__(self):
59         return hash((self.start, self.length, self.val))
60     def __repr__(self):
61         return str(self.start) + "~~~" + str(self.end)
62     def __str__(self):
63         return str(self.start) + "~~~" + str(self.end)
64
65 class Gap(object):
66     def __init__(self, start, end, parent, val_l, val_r):
67         self.start = start
68         self.end = end
69         self.length = end - start
70         self.parent = parent
71         self.children = []
72         self.vleft = val_l
73         self.vright = val_r
74     def __hash__(self):
75         return hash((self.start, self.end))
76     def __repr__(self):
77         return str(self.start) + "~~~" + str(self.end)
78     def __str__(self):

```

```

79         return str(self.start) + "~~~~" + str(self.end)
80
81     class Hole(object):
82         def __init__(self, start, end, val_left, val_right, pts, shifts, tl):
83             self.start = start
84             self.end = end
85             self.length = end - start
86             self.val_left = val_left
87             self.val_right = val_right
88             self.town_left = tl
89             try:
90                 _ = (e for e in pts)
91                 self.pts = pts
92                 self.shifts = shifts
93             except TypeError:
94                 self.pts = [pts]
95                 self.shifts = [shifts]
96         def __hash__(self):
97             return hash((self.start, self.end))
98         def __repr__(self):
99             return str(self.start) + "~~~~" + str(self.pts) + "~~~~" + str(self.
            end)
100        def __str__(self):
101            return str(self.start) + "~~~~" + str(self.pts) + "~~~~" + str(self.
            end)
102
103    class BT(object):
104        def __init__(self, treetop):
105            self.levels = mp.mpf('1')
106            self.top = treetop
107            self.leaves = IntervalTree()
108            self.leafTowns = IntervalTree()
109            self.leafGaps = IntervalTree()
110
111
112    mp.prec = args.prec
113    n = args.dim
114    J = args.J
115    Q = mp.mpf(2*n + 1)
116    Q_int = 2*n + 1
117    P = mp.mpf(n)
118    theta = args.break_ratio # \theta \in (1/2, 1)
119    epsilon = mp.mpf('1')/(Q_int-1)
120    draw = args.draw
121    reftown = Town(0,0,0,None,0,None)
122    verbose = args.verbose
123    summary = args.summary
124    plotfxn = args.plot
125    plotPhi = args.Phi
126
127    def evaluate(B, pt, q):
128        if pt >= 1.0: # safeguard trying to evaluate at endpoint
129            pt = mp.mpf('1') - mp.mpf('2')*(-J - 2)
130            pt = pt - q*epsilon

```

```

131     bb = B.leaves[pt]
132     [b] = bb
133     bl = b.data.vleft
134     br = b.data.vright
135     bs = b.data.start
136     be = b.data.end
137     return bl + (br-bl)/(be-bs)*(pt-bs)
138
139 def Phi(B, x, q):
140     return sum([(mp.mpf(1.0)/((p+2)*mp.mpf(0.5)))*evaluate(B, x[p], q) for p
141                  in range(len(x))])
142
143 def plot_fxn(towntree, shift):
144     points = []
145     vals = []
146     # get coord values for plot
147     st = sorted(list(towntree))
148     for twn in st:
149         t = twn.data
150         if t.start + shift >= 0.0 and t.end + shift <= 1.0:
151             points.append(t.start + shift)
152             points.append(t.end + shift)
153             vals.append(t.val)
154             vals.append(t.val)
155         elif t.start + shift < 1.0 and t.end + shift > 1.0:
156             points.append(t.start + shift)
157             points.append(1.0)
158             vals.append(t.val)
159             vals.append(t.val)
160         elif t.start + shift < 0.0 and t.end + shift > 0.0:
161             points.append(0.0)
162             points.append(t.end + shift)
163             vals.append(t.val)
164             vals.append(t.val)
165     plt.plot(points, vals, '-')
166
167 def plot_towns(towntree):
168     fig1 = plt.figure()
169     ax1 = fig1.add_subplot(111)
170
171     y_shift = 0.05
172     for q in range(Q_int):
173         for twn in towntree:
174             t = twn.data
175             ax1.add_patch(
176                 patches.Rectangle(
177                     (t.start + q*epsilon, y_shift + q*0.2),      # (x,y)
178                     t.length,                                     # width
179                     0.1,                                          # height
180                 )
181             )
182     plt.xlim([-1,2])
183     plt.show()

```

```

184
185 def draw_BT_Node(node, vshift, yloc, ax1):
186     if type(node) == type(reftown):
187         ax1.add_patch(
188             patches.Rectangle(
189                 (node.start, yloc),          # (x,y)
190                 node.end - node.start,      # width
191                 vshift,                      # height
192             )
193         )
194
195 def visualize_BT_Tree(node, level, vshift, ax1):
196     draw_BT_Node(node, vshift, vshift*(2*level + 1), ax1)
197     for c in node.children:
198         visualize_BT_Tree(c, level+1, vshift, ax1)
199
200 def visualize_BT(B):
201     BT = B.top
202     num_levels = B.levels
203     fig1 = plt.figure()
204     ax1 = fig1.add_subplot(111)
205     vshift = 1.0/(2.0*num_levels + 1.0)
206     visualize_BT_Tree(BT, 0, vshift, ax1)
207     plt.xlim([-1,1])
208     plt.show()
209
210 # Function get_towns_above_thres: IntervalTree(data=Town), num —> list(Town)
211 # Gets all towns whose length is above the threshold
212 # Input:
213 #   towntree: IntervalTree of Towns
214 #   thres: float, threshold to determine those to break
215 # Output:
216 #   list of Towns
217 def get_towns_above_thresh(towntree, thresh):
218     return [t.data for t in towntree if t.data.length > thresh]
219
220 # Function get_my_town: IntervalTree(data=Town), pt —> Town
221 # Returns the town a point lies in; else returns None
222 # Input:
223 #   towntree: IntervalTree of Towns
224 #   pt: float
225 # Output:
226 #   Town, if pt is in the Town
227 #   None, if there is no Town that includes pt
228 def get_my_town(towntree, pt):
229     town_ret = towntree[pt]
230     emptyset = set()
231     if town_ret != emptyset:
232         [tr] = town_ret
233         if tr.data.start == pt:
234             return None
235         elif tr.data.end == pt:
236             return None
237     else:

```

```

238         return tr.data
239     else:
240         return None
241
242 # Function is_pt_in_town: IntervalTree(Towns), num, num → Town() OR str
243 # Checks to see whether a point falls within a shifted set of Towns
244 # Input:
245 #   towntree: IntervalTree of Town()'s
246 #   pt: point to check
247 #   shift: shift for the towntree
248 # Output:
249 #   If pt is in the shifted towntree: the shifted town that includes the point
250 #   If pt is before the towns start: string 'before'
251 #   If pt is after the towns end: string 'after'
252 #   If pt is in gap of shifted towntree: string 'gap'
253 def is_pt_in_town(towntree, pt, shift):
254     if pt - shift <= -1:
255         return "before"
256     elif pt - shift >= 1:
257         return "after"
258     else:
259         town_ret = get_my_town(towntree, pt - shift)
260         if town_ret == None:
261             return 'hole'
262         else:
263             return town_ret
264
265 # Function get_gap: IntervalTree(Towns), list(num), num → num, num
266 # Determines size of gap, taking into account both other Towns and other
breakpts/gaps as well
267 # Input:
268 #   towntree: IntervalTree of all Towns
269 #   breakpts: list of all points to break at this level of refinement
270 #   pt: the break point
271 # Output:
272 #   break_start: start of break
273 #   break_end: end of break
274 def get_gap(towntree, breakpts, pt):
275     rho_l = np.inf
276     rho_r = np.inf
277     if verbose:
278         print "\nGetting_break_for_point", pt, "\n"
279     for q in range(-Q_int+1, Q_int):
280         my_town = get_my_town(towntree, pt + q*epsilon)
281         if my_town is not None:
282             if verbose:
283                 print pt + q*epsilon, ":\t", my_town, "\t\t\rho_l:\t", pt + q*
                    epsilon - my_town.start, "\t\t\rho_r:\t", my_town.end - (pt
                    + q*epsilon)
284                 d_l = (mp.mpf(2)/3)*(pt + q*epsilon - my_town.start) #const < 1
285                 if d_l < rho_l:
286                     rho_l = d_l
287                 d_r = (mp.mpf(2)/3)*(my_town.end - (pt + q*epsilon))
288                 if d_r < rho_r:

```

```

289         rho_r = d_r
290     else:
291         if verbose:
292             print pt + q*epsilon, ":\t", my_town
293     for q in range(-(2*Q_int)+1,2*Q_int): #Avoid creating overlapping gaps on
        this refinement level
294         rho_pts_l = [pt - (bp+q*epsilon) for bp in breakpts if pt > bp + q*
            epsilon]
295         rho_pts_r = [(bp+q*epsilon) - pt for bp in breakpts if pt < bp + q*
            epsilon]
296         if len(rho_pts_l) > 0:
297             d_l = (mp.mpf(1)/3)*min(rho_pts_l) #const < 1/2
298             if d_l < rho_l:
299                 rho_l = d_l
300         if len(rho_pts_r) > 0:
301             d_r = (mp.mpf(1)/3)*min(rho_pts_r)
302             if d_r < rho_r:
303                 rho_r = d_r
304     assert rho_l > 0
305     assert rho_r > 0
306     break_start = pt - min(rho_l, (mp.mpf(1)/3)*epsilon)
307     break_end = pt + min(rho_r, (mp.mpf(1)/3)*epsilon)
308     if verbose:
309         print '\nbreak:_', break_start, "_~~~_", pt, '_~~~_', break_end, "\n"
310     return break_start, break_end
311
312 # Function get_border_towns: IntervalTree(Town), num, list(Town) —> list(Town
    ), list(Town), list(num)
313 # Given points and their shifts that align into holes, determines the towns to
    the left and right of the point
314 # Inputs:
315 #   towntree: IntervalTree of all Trees
316 #   pt: point to check
317 #   townlist: list of Town's or key string 'hole' corresponding to where each
    pt falls
318 # Output:
319 #   t_left: list of towns to the left of each pt for each in townlist
320 #   t_right: list of towns to the right of each pt for each in townlist
321 #   shifts: list of shifts for the pts to align in the holes
322 def get_border_towns(towntree, pt, townlist):
323     QQ = range(-Q_int+1,Q_int)
324     qq = [q for q, t in enumerate(townlist) if t=='hole']
325     t_left = []
326     t_right = []
327     shifts = []
328     for p in range(len(qq)):
329         q_k = -QQ[qq[p]]
330         shifts.append(q_k*epsilon)
331         t_left.append( max([t.data for t in towntree if t.data.end < pt +
            shifts[p]], key=attrgetter('end')) )
332         t_right.append( min([t.data for t in towntree if t.data.start > pt +
            shifts[p]], key=attrgetter('start')) )
333     return t_left, t_right, shifts
334

```

```

335 def refine(j, B):
336     time_start = time.time()
337     newLeaves = IntervalTree()
338     newLeafTowns = IntervalTree()
339     newLeafGaps = IntervalTree()
340     towntree = B.leafTowns
341
342     towns_to_break = get_towns_above_thresh(B.leafTowns, theta**j)
343     ntb = len(towns_to_break)
344     if verbose:
345         print 'ntb', ntb
346         print
347     if ntb < 1:
348         if verbose:
349             print('No towns to break at this iteration')
350         return
351
352 #
353 #
354     #####
355     ###      ###
356     ### PLUGS ###
357     ###      ###
358     #####
359
360 # Find Holes and Solve for Plugs
361 holetree = IntervalTree()
362 for tb in towns_to_break:
363     pt = tb.center
364     if verbose:
365         print
366         print 'tb', tb
367         print 'pt', pt
368     pt_towns = [None] * (2*Q_int - 1)
369     for q in range(-Q_int+1, Q_int):
370         pt_towns[q+Q_int-1] = is_pt_in_town(towntree, pt, q*epsilon)
371     num_h = sum(t=='hole' for t in pt_towns)
372     if verbose:
373         print 'pt_towns', pt_towns
374         print 'nh', num_h
375     if num_h > 2:
376         print "NOT RIGHT NUMBER OF BREAKS!"
377         return
378     if num_h > 0:
379         gap_left, gap_right, shifts = get_border_towns(towntree, pt,
380             pt_towns)
381         for i in range(num_h):
382             already_present = holetree[gap_left[i].end] # Check to see
383                 if this hole is already present in the holetree
384             if len(already_present) == 0: # Hole is not
385                 present —> add hole
386                 h = Hole(gap_left[i].end, gap_right[i].start, gap_left[i].
387                     val, gap_right[i].val, pt, shifts[i], gap_left[i])
388                 holetree.addi(gap_left[i].end, gap_right[i].start, h)
389             if verbose:

```

```

385         print "Hole:\t", h
386     else:
387         # Hole is
388         present —> update list of points that fall in that hole
389         h = already_present.pop().data
390         h.pts.append(pt)
391         h.shifts.append(shifts[i])
392         holetree.addi(h.start, h.end, h)
393         if verbose:
394             print "Hole:\t", h
395
396 for g in B.leafGaps:
397     gb = g.data
398     ht_intersect = holetree[gb.start:gb.end]
399     if ht_intersect == set():
400         gb_dup = Gap(gb.start, gb.end, gb, gb.vleft, gb.vright)
401         gb.children = [gb_dup]
402         newLeaves.addi(gb_dup.start, gb_dup.end, gb_dup)
403         newLeafGaps.addi(gb_dup.start, gb_dup.end, gb_dup)
404     else:
405         assert len(ht_intersect) == 1
406         [w] = ht_intersect
407         h = w.data
408         num_pts = len(h.pts)
409         if verbose:
410             print '\nHole:\t', h
411             old_slope = (h.val_right - h.val_left)/h.length
412             new_slope = mp.mpf(1) - mp.mpf(2)**(-j-1)
413
414         p_shifted = [h.pts[p] + h.shifts[p] for p in range(num_pts)]
415         p_shifted.sort()
416         f = [h.val_left + old_slope*(p_shifted[p] - h.start) for p in
417              range(num_pts)]
418         f.insert(0, h.val_left)
419         f.append(h.val_right)
420
421         # Construct System
422         C = mp.matrix(2*num_pts)
423         z = mp.matrix(2*num_pts, 1)
424         for i in range(num_pts + 1):
425             if i == 0:
426                 C[0,0] = new_slope
427                 z[0] = f[1] - f[0] + new_slope*h.start
428             elif i == num_pts:
429                 C[num_pts, 2*num_pts-1] = -new_slope
430                 z[num_pts] = f[-1] - f[-2] - new_slope*h.end
431             else:
432                 C[i, 2*i-1] = -new_slope
433                 C[i, 2*i] = new_slope
434                 z[i] = f[i+1] - f[i]
435         if num_pts > 1:
436             for i in range(num_pts+1, 2*num_pts):
437                 C[i, 2*(i-num_pts)-1] = mp.mpf(1)
438                 C[i, 2*(i-num_pts)] = mp.mpf(1)
439                 z[i] = p_shifted[i-num_pts-1] + p_shifted[i-num_pts]

```



```

437
438     if verbose:
439         print 'System: Cx=z '
440         print 'C:\n', C
441         print 'z:\n', z, '\n'
442
443     x = mp.lu_solve(C, z)
444
445     for i in range(num_pts):
446         plug_l = x[2*i]
447         plug_r = x[2*i + 1]
448         if verbose:
449             print 'Point:\t', h.pts[i]
450             print "hole_start:\t", h.start
451             print "plug_l\t\t\t\t\t", plug_l
452             print "pt_shifted:\t", p_shifted[i]
453             print "plug_r\t\t\t\t\t", plug_r
454             print "hole_end\t\t\t\t\t", h.end
455         plug = Town(plug_l, plug_r, f[i+1], f[i+2], j, gb)
456         towntree.addi(plug_l, plug_r, plug)
457         newLeaves.addi(plug_l, plug_r, plug)
458         newLeafTowns.addi(plug_l, plug_r, plug)
459
460         if i == 0:
461             lg_l = h.start
462         else:
463             lg_l = x[2*i - 1]
464         leftgap = Gap(lg_l, plug_l, gb, f[i], f[i+1])
465         newLeaves.addi(leftgap.start, leftgap.end, leftgap)
466         newLeafGaps.addi(leftgap.start, leftgap.end, leftgap)
467
468         gb.children = gb.children + [plug, leftgap]
469
470         rightgap = Gap(plug_r, h.end, gb, f[-2], f[-1])
471         newLeaves.addi(rightgap.start, rightgap.end, rightgap)
472         newLeafGaps.addi(rightgap.start, rightgap.end, rightgap)
473         gb.children = gb.children + [rightgap]
474
475         h.town_left.nextval = f[1]
476
477     if verbose:
478         print "\nPlugged_Towntree:"
479         print [t.data for t in towntree]
480         print
481
482     #
483     # #####
484     ###      ###
485     ### GAPS ###
486     ###      ###
487     #####
488
489     breakpts = [tb.center for tb in towns_to_break]
490     for t in B.leafTowns:

```

```

491     tb = t.data
492     if tb.length > theta**j:
493         pt = tb.center
494         break_start, break_end = get_gap(towntree, breakpts, pt)
495         if (tb.nextval == None) or ((tb.nextval - tb.val) > (break_end -
            break_start)*mp.mpf('0.5')):
496             new_val = tb.val + (break_end - break_start)*mp.mpf('0.5') #
                Starting slope value
497         else: #Not enough room to go up a full starting-slope
498             new_val = (tb.val + tb.nextval)*mp.mpf('0.5') #average
499
500         left = Town(tb.start, break_start, tb.val, new_val, tb.birth, tb)
501         right = Town(break_end, tb.end, new_val, tb.nextval, tb.birth, tb)
502         middle = Gap(break_start, break_end, tb, tb.val, new_val)
503
504         tb.children = [left, right, middle]
505         newLeaves.addi(left.start, left.end, left)
506         newLeaves.addi(right.start, right.end, right)
507         newLeaves.addi(middle.start, middle.end, middle)
508         newLeafTowns.addi(left.start, left.end, left)
509         newLeafTowns.addi(right.start, right.end, right)
510         newLeafGaps.addi(middle.start, middle.end, middle)
511     else:
512         tb_dup = Town(tb.start, tb.end, tb.val, tb.nextval, tb.birth+1, tb
            )
513         tb.children = [tb_dup]
514         newLeaves.addi(tb_dup.start, tb_dup.end, tb_dup)
515         newLeafTowns.addi(tb_dup.start, tb_dup.end, tb_dup)
516
517
518 #
519 #
520 #####
521 ###          ###
522 ### UPDATE B ###
523 ###          ###
524 #####
525
526 B.levels = B.levels + 1
527 B.leaves = newLeaves
528 B.leafTowns = newLeafTowns
529 B.leafGaps = newLeafGaps
530
531 time_end = time.time()
532 if summary or verbose:
533     print '\ntheta:\t\t\t', theta**j
534     print 'smallest_town_size:\t', min([t.data.length for t in B.leafTowns
        ])
535     print 'largest_town_size:\t', max([t.data.length for t in B.leafTowns
        ])
536     print 'number_of_towns:\t', len(B.leafTowns)
537     print 'number_of_towns_broken:\t', ntb
538     print 'total_length:\t\t', sum([t.data.length for t in B.leafTowns])
539     print 'time_elapsed:\t\t', time_end - time_start

```

```

540         print
541     if draw == 1:
542         plot_towns(B.leafTowns)
543
544 def fullRefine(J):
545     print
546     print( "Setting up a Sprecher Town System" )
547
548     domain = IntervalTree()
549     start = mp.mpf(-1)
550     end = mp.mpf(1)
551     I = Town(start, end, mp.mpf(0), None, 0, None)
552     domain.addi(start, end, I)
553
554     B = BT(I)
555     B.leaves = domain
556     B.leafTowns = domain
557
558     print( 'Beginning refinement ... ' )
559     print
560
561     j = mp.mpf('0')
562     while j < J:
563         if summary or verbose:
564             print 'Beginning level ', j
565             refine(j, B)
566             j = j+1
567
568     return B
569
570 def plot_Phi(B):
571     assert P == 2
572     for q in range(Q):
573         Psi_q = lambda x, y: Phi(B, (x,y), mp.mpf(str(q)))
574         mp.splot(Psi_q, [0.0,0.99999], [0.0,0.99999], points = 100)
575         plt.show()
576
577
578 def constructInnerFunction_plotting(B):
579     if draw == 1 or draw == 2:
580         plot_towns(B.leafTowns)
581     if plotfxn:
582         fig2 = plt.figure()
583         plot_fxn(B.leafTowns, 0)
584         plt.show()
585     if plotPhi:
586         plot_Phi(B)
587
588 # Driver for Inner Function Construction
589 # Visualizes evolution of Towns as they undergo refinement
590 def constructInnerFunction():
591     B = fullRefine(J)
592     constructInnerFunction_plotting(B)
593     if draw == 1 or draw == 2:

```

```
594         visualize_BT(B)
595     return B
```

Appendix E

Code: Köppen's Inner Function

The following Python code implements Köppen's inner function [26], as described in Chapter 5.

```

1  from numpy import *
2  import sys, argparse, copy, time
3  import matplotlib.pyplot as plt
4  import matplotlib.patches as patches
5  from mpl_toolkits.mplot3d import Axes3D
6  from collections import Counter
7  from bisect import *
8  from intervaltree import Interval, IntervalTree
9
10
11  parser = argparse.ArgumentParser(
12  parser.add_argument('--dim', type=int, default=2,
13                      help='The spatial dimension')
14  parser.add_argument('--kmin', type=int, default=1,
15                      help='The initial resolution')
16  parser.add_argument('--kmax', type=int, default=5,
17                      help='The final resolution')
18  parser.add_argument('--gamma', type=int, default=10,
19                      help='Base for expansion')
20  args = parser.parse_args()
21
22  n = args.dim
23  m = 2*n + 1
24  gamma = args.gamma
25  a = 1./(gamma * (gamma - 1.0))
26
27
28  def beta(r):
29      return (pow(n, r) - 1.)/(n - 1.)
30
31  def lambda_p(p):
32      if p == 1:
33          return 1
34      else:

```

```

35     eps = 1.
36     tol = sys.float_info.epsilon
37     total = 0.0
38     r = 1.
39     while eps > tol:
40         eps = pow(gamma,(1-p)*beta(r))
41         r += 1.
42         total += eps
43     return total
44
45 def extractDigits(d, k, gamma):
46     '''Return a list of the first k base gamma digits for an integer, with the
47         most significant digit first'''
48     arr_d = [];
49     for i in range(k):
50         arr_d.insert(0, d % gamma)
51         d = d // gamma
52     return arr_d
53
54 def koppenPhiDigits(d, gamma):
55     '''Koppen algorithm applied to a base gamma digit string'''
56     k = len(d)
57     if k == 1:
58         return d[0]/gamma
59     elif d[-1] < (gamma - 1):
60         return koppenPhiDigits(d[:-1], gamma) + d[-1]*pow(gamma, -beta(k))
61     else:
62         d_prev = d[:-1]
63         d_next = copy.deepcopy(d)
64         while d_next != [] and d_next[-1] == gamma-1:
65             d_next = d_next[:-1]
66         if d_next == []:
67             d_next = [gamma]
68         else:
69             d_next[-1] += 1
70         return 0.5 * (koppenPhiDigits(d_prev, gamma) + koppenPhiDigits(d_next,
71             gamma) + (gamma-2)*pow(gamma, -beta(k)))
72
73 def koppenPhi(dk, gamma, k):
74     '''Koppen algorithm for evaluating an inner function at base gamma decimal d
75         to accuracy level k'''
76     if k == 1:
77         return dk
78     else:
79         dk = dk*pow(gamma, k) # Shift decimal to be a whole
80                                number
81         d = extractDigits(round(dk), k, gamma) # Turn into a list of digits
82         return koppenPhiDigits(d, gamma)
83
84 def innerTabulation(gamma, k):
85     dkrange = linspace(0., 1., num = pow(gamma,k) + 1)
86     kP = array(dkrange)
87     for i, dk in enumerate(dkrange[:-1]):
88         value = koppenPhi(dk, gamma, k)

```

```

85     kP[i] = value
86     return dkrange, kP
87
88 def computeArcLength(kP):
89     rises = [ current - prev for prev, current in zip(kP, kP[1:])]
90     slopes = [ rises[j]/pow(gamma, -k) for j in range(len(rises))]
91     piece_lengths = [sqrt( pow(gamma, -2*k) + rises[j]**2) for j in range(len(
        rises))]
92     curve_lengths = cumsum(piece_lengths)
93     total = curve_lengths[-1]
94     curve_lengths = insert(curve_lengths, 0, 0)
95     return rises, curve_lengths, total
96
97 def array_to_fxn(xs, ys):
98     def arr_func(t):
99         if t <= xs[0]:
100             return ys[0]
101         elif t >= xs[-1]:
102             return ys[-1]
103         else:
104             i = bisect_left(xs, t)
105             if i == len(xs) - 1:
106                 return ys[-1]
107             else:
108                 xl = xs[i]
109                 xr = xs[i+1]
110                 return ys[i] + (t - xl)*(ys[i+1] - ys[i])
111     return arr_func
112
113
114 def KoppenIntervals(gamma, k):
115     dkrange = linspace(0., 1., num = pow(gamma,k) + 1)
116     starts = dkrange[:-1]
117     ends = [s + (gamma-1)*pow(gamma, -(k+1)) for s in starts]
118     return [(s, e) for s, e in zip(starts, ends)]
119
120 def plot_intervals(list_intervals):
121     fig1 = plt.figure()
122     ax1 = fig1.add_subplot(111)
123
124     y_shift = 0.2
125     q = -1
126     for I in list_intervals:
127         q += 1
128         for i in I:
129             ax1.add_patch(
130                 patches.Rectangle(
131                     (i[0], q*y_shift),      # (x,y)
132                     i[1] - i[0],           # width
133                     0.1,                   # height
134                     )
135             )
136     plt.xlim([0, 1])
137     plt.ylim([-0.1, (q+1)*y_shift])

```

```

138     plt.show()
139
140 def plot_interval_shifted(I, shift, num_shifts):
141     fig1 = plt.figure()
142     ax1 = fig1.add_subplot(111)
143
144     y_shift = 0.2
145     for q in range(num_shifts):
146         for i in I:
147             ax1.add_patch(
148                 patches.Rectangle(
149                     (i[0] + q*shift, q*y_shift),      # (x,y)
150                     i[1] - i[0],                      # width
151                     0.1,                              # height
152                 )
153             )
154     plt.ylim([-0.1, (q+1)*y_shift])
155     plt.show()
156
157
158 for k in range(args.kmin, args.kmax):
159     dkrange, kP = innerTabulation(gamma, k)
160     rises, curve_lengths, total_length = computeArcLength(kP)
161     print 'length_of_curve_', k, ':\t', total_length
162     plt.plot(dkrange, [c / total_length for c in curve_lengths], label=str(k))
163     plt.plot([c/total_length for c in curve_lengths], dkrange, label='inv_' +
164              str(k))
165     plt.legend(loc='upper_left')
166     plt.show()
167     scaled_curve = [c / total_length for c in curve_lengths]
168
169     sigma = array_to_fxn(dkrange, scaled_curve)
170     KI = KoppenIntervals(gamma, k)
171     sigmaKI = [(sigma(i[0]), sigma(i[1])) for i in KI]
172     print 'max_interval_length_for_KI_', k, ':\t', max([s[1]-s[0] for s in
173              sigmaKI])
174     plot_intervals([KI, sigmaKI])
175     plot_interval_shifted(KI, 1./(gamma*(gamma-1)), 5)
176     plot_interval_shifted(sigmaKI, 1./(gamma*(gamma-1)), 5)
177
178     sigma_inv = array_to_fxn(scaled_curve, dkrange)
179     psi = array_to_fxn(dkrange, kP)
180     mapped_psi = [psi(sigma_inv(dkrange[d])) for d in range(len(dkrange))]
181     plt.plot(dkrange, kP)
182     plt.plot(dkrange, mapped_psi)
183     plt.show()

```


Bibliography

- [1] Vladimir I. Arnol'd. On Functions of Three Variables. *Dokl. Akad. Nauk SSSR*, 114:679–681, 1957.
- [2] Vladimir I. Arnol'd. On the Representation of Functions of Several Variables as a Superposition of Functions of a Smaller Number of Variables. *Mat. Prosveshchenie*, 3:4161, 1958.
- [3] Aida Kh. Asgarova and Vugar E. Ismailov. Diliberto–Straus Algorithm for the Uniform Approximation by a Sum of Two Algebras. *Proceedings-Mathematical Sciences*, 127(2):361–374, 2017.
- [4] Robert Bellman. Curse of Dimensionality. *Adaptive Control Processes: A Guided Tour*. Princeton, NJ, 1961.
- [5] Valeriu Bern and Artur Zawadzki. On Kolmogorov's Superpositions: Novel Gates and Circuits for Nanoelectronics? In *IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on Neural Networks*, volume 1, pages 651–656. IEEE, 2005.
- [6] Lorenz T. Biegler, Omar Ghattas, Matthias Heinkenschloss, and Bart van Bloemen Waanders. Large-Scale PDE-Constrained Optimization: An Introduction. In *Large-Scale PDE-Constrained Optimization*, pages 3–13. Springer, 2003.

- [7] Jurgen Braun and Michael Griebel. On a Constructive Proof of Kolmogorov's Superposition Theorem. *Constr. Approx.*, 30:653–675, 2009.
- [8] Mark Coppejans. On Kolmogorov's Representation of Functions of Several Variables by Functions of One Variable. *Journal of Econometrics*, 123(1):1–31, 2004.
- [9] Jérôme Darbon and Stanley Osher. Algorithms for Overcoming the Curse of Dimensionality for Certain Hamilton–Jacobi Equations Arising in Control Theory and Elsewhere. *Research in the Mathematical Sciences*, 3(1):19, 2016.
- [10] Rui de Figueiredo. Implications and Applications of Kolmogorov's Superposition Theorem. *IEEE Transactions on Automatic Control*, 25(6):1227–1231, 1980.
- [11] Stephen Demko. A Superposition Theorem for Bounded Continuous Functions. *Proceedings of the American Mathematical Society*, 66(1):75–78, 1977.
- [12] Persi Diaconis and Mehrdad Shahshahani. On Nonlinear Functions of Linear Combinations. *SIAM Journal on Scientific and Statistical Computing*, 5(1):175–191, 1984.
- [13] Stephen P. Diliberto and Ernst G. Straus. On the Approximation of a Function of Several Variables by the Sum of Functions of Fewer Variables. *Pacific J. Math*, 1(2):195–210, 1951.
- [14] Raouf Doss. A Superposition Theorem for Unbounded Continuous Functions. *Transactions of the American Mathematical Society*, 233:197–203, 1977.
- [15] Buma L. Fridman. Improvement in the Smoothness of Functions in the Kolmogorov Superposition Theorem. *Dokl. Akad. Nauk SSR*, 177:5:1019–1022, 1967. English transl.

- Soviet Math. Dokl. 8, 6 (1967), 1550-1553.
- [16] Jerome H. Friedman and Werner Stuetzle. Projection Pursuit Regression. *Journal of the American statistical Association*, 76(376):817–823, 1981.
 - [17] Federico Girosi and Tomaso Poggio. Representation Properties of Networks: Kolmogorov’s Theorem is Irrelevant. *Neural Computation*, 1(4):465–469, 1989.
 - [18] Robert Hecht-Nielsen. Kolmogorov’s Mapping Neural Network Existence Theorem. In *Proceedings of the international conference on Neural Networks*, pages 11–14. IEEE Press, 1987.
 - [19] Torbjörn Hedberg. The Kolmogorov Superposition Theorem, Appendix II in Topics in Approximation Theory, HS Shapiro. *Lecture Notes*, 187, 1971.
 - [20] David Hilbert. Mathematical Problems. *Bulletin of the American Mathematical Society*, 8(10):437–479, 1902.
 - [21] Boris Igelnik and Neel Parikh. Kolmogorov’s Spline Network. *IEEE transactions on neural networks*, 14(4):725–733, 2003.
 - [22] Jean-Pierre Kahane. Sur le Theoreme de Superposition de Kolmogorov. *Journal of Approximation Theory*, 13(3):229–234, 1975.
 - [23] Semen Ya. Khavinson. *Best Approximation by Linear Superpositions (Approximate Nomography)*, volume 159. American Mathematical Soc., 1997.
 - [24] Andrei N. Kolmogorov. The Representation of Continuous Functions of Several Variables by Superpositions of Continuous Functions of a Smaller Number of Variables.

- Doklady Akademii Nauk SSSR*, 108(2):179–182, 1956.
- [25] Andrei N. Kolmogorov. On the Representation of Continuous Functions of Several Variables as Superpositions of Continuous Functions of One Variable and Addition. *Dokl. Akad. Nauk SSSR*, 114:5:953–956, 1957. English transl. Amer. Math. Soc. Transl. (2) 28 (1963), 55.
- [26] Mario Köppen. *On the Training of a Kolmogorov Network*, pages 474–9. ICANN 2002, LNCS 2415, 2002.
- [27] Věra Kůrková. Kolmogorov’s Theorem and Multilayer Neural Networks. *Neural Networks*, 5(3):501–506, 1992.
- [28] Pierre-Emmanuel Leni, Yohan D. Fougerolle, and Frédéric Truchetet. Kolmogorov Superposition Theorem and Wavelets for Image Compression. *Wavelet Applications in Industrial Processing VII, Proceedings of the SPIE*, 7535(1):753502–753510, 2010.
- [29] William A. Light. The DilibertoStraus Algorithm in $L_1(X \times Y)$. *Journal of Approximation Theory*, 38(1):1–8, 1983.
- [30] Benjamin F. Logan. The Uncertainty Principle in Reconstructing Functions from Projections. *Duke Mathematical Journal*, 42(4):661–706, 1975.
- [31] Benjamin F. Logan and Larry A. Shepp. Optimal Reconstruction of a Function from its Projections. *Duke Mathematical Journal*, 42(4):645–659, 1975.
- [32] George G. Lorentz. *Approximation of Functions*. Holt, Rinehart and Winston New York, 1966.

- [33] Phillip A. Ostrand. Dimension of Metric Spaces and Hilberts Problem 13. *Bulletin of the American Mathematical Society*, 71(4):619–622, 1965.
- [34] Alexander Ostrowski. Über Dirichletsche Reihen und Algebraische Differentialgleichungen. *Mathematische Zeitschrift*, 8(3-4):241–298, 1920.
- [35] Allan Pinkus. *Ridge Functions*, volume 205. Cambridge University Press, 2015.
- [36] George Pólya and Gabor Szegő. *Problems and Theorems in Analysis I: Series, Integral Calculus, Theory of Functions*. Springer, 1978. Transl. D. Aepli.
- [37] Marcello Sanguineti. Universal Approximation by Ridge Computational Models and Neural Networks: A Survey. *The Open Applied Mathematics Journal*, 2(1):31–58, 2008.
- [38] David A. Sprecher. On the Structure of Continuous Functions of Several Variables. *Transactions of the American Mathematical Society*, 115:340–355, 1965.
- [39] David A. Sprecher. On the Structure of Representations of Continuous Functions of Several Variables as Finite Sums of Continuous Functions of One Variable. *Proceedings of the American Mathematical Society*, 17(1):98–105, 1966.
- [40] David A. Sprecher. On Similarity in Functions of Several Variables. *The American Mathematical Monthly*, 76(6):627–632, 1969.
- [41] David A. Sprecher. An Improvement in The Superposition Theorem of Kolmogorov. *Journal of Mathematical Analysis and Applications*, 38:208–213, 1972.
- [42] David A. Sprecher. A Numerical Implementation of Kolmogorov’s Superpositions. *Neural Networks*, 9(5):765–772, 1996.

- [43] David A. Sprecher. A Numerical Implementation of Kolmogorov's Superpositions II. *Neural Networks*, 10(3):447–457, 1997.
- [44] David A. Sprecher. *From Algebra to Computational Algorithms: Kolmogorov and Hilbert's Problem 13*. Docent Press, 2017.
- [45] Yaki Sternfeld. Hilbert's 13th Problem and Dimension. In *Geometric Aspects of Functional Analysis*, pages 1–49. Springer, 1989.
- [46] John M. Sullivan. Curves of Finite Total Curvature. In *Discrete Differential Geometry*, pages 137–161. Springer, 2008.
- [47] Anatoli G. Vitushkin. On the 13th Problem of Hilbert. *DAN*, 95:701–704, 1954.
- [48] Anatoli G. Vitushkin. Some Properties of Linear Superpositions of Smooth Functions. In *Dokl. Akad. Nauk SSSR*, volume 156, pages 1003–1006, 1964.
- [49] Vladimir M. Tikhomirov. Kolmogorov's Work on ϵ -Entropy of Functional Classes and the Superposition of Functions. *Russian Mathematical Surveys*, 18:5:51–87, 1963.