Annals of Mathematics, **169** (2009), 315–346

Fitting a C^m-smooth function to data I

By Charles Fefferman* and Bo'az Klartag**

Abstract

Suppose we are given a finite subset $E \subset \mathbb{R}^n$ and a function $f : E \to \mathbb{R}$. How to extend f to a C^m function $F : \mathbb{R}^n \to \mathbb{R}$ with C^m norm of the smallest possible order of magnitude? In this paper and in [20] we tackle this question from the perspective of theoretical computer science. We exhibit algorithms for constructing such an extension function F, and for computing the order of magnitude of its C^m norm. The running time of our algorithms is never more than $CN \log N$, where N is the cardinality of E and C is a constant depending only on m and n.

1. Introduction

Let $m, n \geq 1$, and suppose we are given N points in $\mathbb{R}^{n+1}$. We regard m and n as fixed, but N as arbitrarily large. Among all functions $F : \mathbb{R}^n \to \mathbb{R}$ whose graphs pass through (or near to) the given points, we would like to find one whose norm in $C^m(\mathbb{R}^n)$ is of the smallest possible order of magnitude. Also, we would like to know the order of magnitude of the C^m norm of such an F. In this paper and [20], we give algorithms to solve these problems, and we estimate the resources required by an (idealized) computer to carry them out.

To state the above problems more precisely, we set up some notation. Let $E \subset \mathbb{R}^n$ be a finite set of cardinality

$$\#(E) = N \, , \tag{0}$$

and let $f : E \to \mathbb{R}$ and $\sigma : E \to [0, \infty)$ be given functions on E. Then we write $\|f\|_{C^m(E,\sigma)}$ to denote the infimum of all $M > 0$ for which there exists $F \in C^m(\mathbb{R}^n)$, such that

$$\|F\|_{C^m(\mathbb{R}^n)} \leq M, \text{ and } |F(x) - f(x)| \leq M\sigma(x) \text{ for all } x \in E. \tag{1}$$

*This work was partially supported by NSF grant #DMS-0245242.
** Supported by the Clay Mathematics Institute and by NSF grant #DMS-0456590.

The case $\sigma = 0$ is natural, as it corresponds to the problem of extending the function $f : E \to \mathbb{R}$ to a C^m function on the entire $\mathbb{R}^n$, with C^m norm of the smallest possible order of magnitude. Here, as usual, $C^m(\mathbb{R}^n)$ denotes the space of all m-times continuously differentiable $F : \mathbb{R}^n \to \mathbb{R}$, for which the norm

$$\|F\|_{C^m(\mathbb{R}^n)} = \max_{|\beta| \leq m} \sup_{x \in \mathbb{R}^n} \left| \partial^\beta F(x) \right|$$

is finite.

In this paper we solve

Problem 1. *Compute the order of magnitude of* $\|f\|_{C^m(E,\sigma)}$.

By "order of magnitude" we mean the following: Two numbers $X, Y \geq 0$ determined by E, f, σ, m, n are said to have "the same order of magnitude" provided we have $cX \leq Y \leq CX$, with constants c and C depending only on m and n. To "compute the order of magnitude of X" is to compute some Y such that X and Y have the same order of magnitude.

In [20] we will solve

Problem 2. *Compute a function* $F \in C^m(\mathbb{R}^n)$ *that satisfies* (1), *with* M *having the same order of magnitude as* $\|f\|_{C^m(E,\sigma)}$.

To "compute a function F" means the following: First, we enter the data E, f, σ into a computer. The computer works for a while, performing L_0 machine operations. It then signals that it is ready to accept further input. Whenever we enter a point $x \in \mathbb{R}^n$, the computer responds by producing an $m^{\underline{\text{th}}}$ degree polynomial P_x on $\mathbb{R}^n$, using L_1 machine operations to make the computation. We say that our algorithm "computes the function F" if, for each $x \in \mathbb{R}^n$, the polynomial P_x produced by that algorithm is precisely the $m^{\underline{\text{th}}}$ order Taylor polynomial of F at x.

We call L_0 the "one-time work" and L_1 the "work to answer a query". When we "compute a function F", the response P_x to a query x is not allowed to depend on any queries x' that may have been previously addressed to the computer.

Our algorithms will run on an idealized computer with standard von-Neumann architecture, able to work with exact real numbers. That is, we assume a model of computation in which the registers and the memory cells are able to store a real number to perfect accuracy, and in which the processor is capable of performing basic arithmetic operations on real numbers. We refer the reader to, e.g., [33] and [25] for a thorough discussion of the von-Neumann architecture, and to, e.g., [28] for the concept of a computer that works with exact real numbers (i.e., the "real RAM" model). We assume that a real number consumes a single register or memory cell, and we assume that it takes one machine operation to add, subtract, multiply or divide two given real numbers x and y, or to compare them (i.e., decide whether $x < y$, $x > y$ or $x = y$). These operations are performed with perfect accuracy. Thus, we ignore here roundoff, overflow and underflow errors, although it is possible to modify our discussion to take account of these issues (see, e.g., the appendix of [20]). We also ignore here issues of parallel computation, although we believe that our algorithms are well-suited to parallelism (see Callahan-Kosaraju [11]).

The "work" or "running time" of an algorithm is the number of machine operations needed to carry it out, and the "storage" of an algorithm is the number of random access memory addresses required.

Our results for Problems 1 and 2 are as follows. Recall that N is the number of points in E.

Theorem 1. *The algorithm to be explained below computes the order of magnitude of* $\|f\|_{C^m(E,\sigma)}$ *using work at most* $CN \log N$ *and storage at most* CN, *where* C *depends only on* m *and* n.

Theorem 2. *The algorithm to be described in* [20] *computes a function* $F \in C^m(\mathbb{R}^n)$ *that satisfies* (1), *with* M *having the same order of magnitude as* $\|f\|_{C^m(E,\sigma)}$. *The one-time work of our algorithm is at most* $CN \log N$, *the storage is at most* CN, *and the work to answer a query is at most* $C \log N$. *Here,* C *depends only on* m *and* n.

Formally, the model of computation we work with in Theorem 2, is slightly different from what was described above. We require, in addition to comparisons and arithmetic operations on exact real numbers, the operations of logarithm, powers of two, and rounding a real number to the closest integer. As will be explained in [20], we do not "abuse" these operations, which are standard computer operations in any programming language known to the authors. For instance, suppose we switch to the common digital computer model, and consider an instance of Problem 2 in which the values of f and σ, and the coordinates of the points of E, are R-bit numbers. Then the algorithm announced in Theorem 2 returns the answer to an accuracy of R bits, and never uses numbers - neither integers nor reals - in an accuracy that requires more than CR bits, for C depending only on m and n. Details are in [20].

The proof of Theorem 1, and a key idea in the proof of Theorem 2, will be given here. The full details of the proof of Theorem 2, including the description of the algorithm, will appear in [20]. In [20] we also deal with some variants of Problem 2. For instance, the function F in Problem 2 may actually be taken to depend linearly on f. We will discuss in [20] a version of the algorithm that computes the coefficients of this linear dependence. An additional feature is related the fact that F in Problem 2 is not uniquely determined. Given

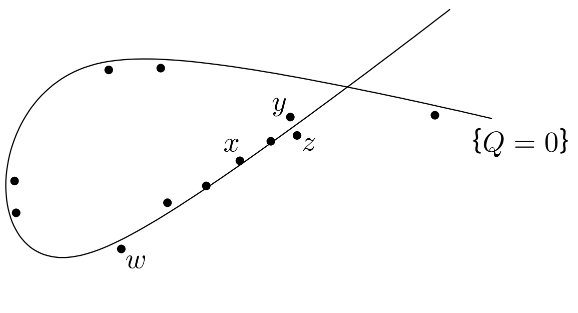

Figure 1

$x \in \mathbb{R}^n$, consider the set of all Taylor polynomials at x of functions that satisfy (1), with M of the same order of magnitude as $\|f\|_{C^m(E,\sigma)}$. We show in [20] that given $x \in \mathbb{R}^n$, an approximation to this set can be computed in $C \log N$ operations, after one-time-work of $CN \log N$ operations and CN storage. We also consider initial data of a different kind from E, f, σ as above.

A significant feature of our algorithms is that they work for arbitrary (finite) $E \subset \mathbb{R}^n$, $f : E \to \mathbb{R}, \sigma : E \to [0, \infty)$. Under simplifying assumptions on the geometry of the set E, it is easy to give fast algorithms for Problems 1 and 2. A delicate case is indicated in Figure 1, where $E \subset \mathbb{R}^2$ lies near the curve $\{Q = 0\}$ for a low-degree polynomial Q. We take $\sigma = 0$, and we take E to consist of the dots in Figure 1. To solve Problem 2, we must be able to determine P_x, the Taylor polynomial of our desired C^m function F at the point x in Figure 1, given the values of F at the points of E. Since all points of E lie close to $\{Q = 0\}$, a crude algorithm may fail to distinguish between the hypotheses $P_x = P$ and $P_x = P + Q$ for a given polynomial P. However, we can make progress by means of the following idea. The line segment joining the two nearby points y and z in Figure 1 meets $\{Q = 0\}$ at a not-so-small angle. We write ν for the unit vector in the direction $y - z$, and we approximate $\nu \cdot \nabla F(x)$ by $\nu \cdot \nabla F(z)$, which in turn is approximated by $\frac{F(y)-F(z)}{|y-z|}$.

Hence, hopefully, $\nu \cdot \nabla F(x)$ is well-approximated by something we can calculate from the values of F at points in E. This allows us to distinguish between the two hypotheses $P_x = P$ and $P_x = P + Q$. However, the same idea works less effectively at the point w in Figure 1, because w lies too far away from y and z. This gives some idea of the issues relevant to Problems 1 and 2 in a delicate case. We invite the reader to trace what our algorithms do in the example sketched in Figure 1. We return now to the general case.

This paper is part of a literature on the problem of extending a given function $f : E \to \mathbb{R}$, defined on an arbitrary subset $E \subset \mathbb{R}^n$, to a function $F \in C^m(\mathbb{R}^n)$. The question goes back to Whitney [35], [36], [37], with contributions by Glaeser [22], Brudnyi-Shvartsman [5]–[10] and [29], [30], [31], Zobin [38], [39], Bierstone-Milman-Pawłucki [2], [3], Fefferman [13]–[19] and A. and Y. Brudnyi [4]. Here, we take E finite, and we pose the question from the viewpoint of theoretical computer science.

The following result from [14] will play a crucial rôle in our treatment of Problems 1 and 2. (See also Brudnyi-Shvartsman [8], whose earlier results and conjectures overlap with those of [14].) We write $\mathcal{P}$ for the vector space of real $(m-1)^{\underline{\text{rst}}}$ degree polynomials on $\mathbb{R}^n$.

Theorem 3. *Given m, $n \geq 1$, there exists k, depending only on m and n, for which the following holds.*

Let $E \subset \mathbb{R}^n$ be finite, let $f : E \to \mathbb{R}$ and $\sigma : E \to [0, \infty)$ be functions on E, and let $M \in (0, \infty)$. Assume that, given any $S \subseteq E$ with $\#(S) \leq k$, there exists a map $y \mapsto P^y$, from S into $\mathcal{P}$, such that:

(a) $|\partial^\alpha P^y(y)| \leq M$ *for* $|\alpha| \leq m-1, y \in S$;

(b) $|\partial^\alpha (P^y - P^{y'})(y)| \leq M|y-y'|^{m-|\alpha|}$ *for* $|\alpha| \leq m-1, y, y' \in S$; *and*

(c) $|P^y(y) - f(y)| \leq M\sigma(y)$ *for all* $y \in S$.

Then $\|f\|_{C^m(E,\sigma)} \leq CM$, where C depends only on m and n.

The converse of Theorem 3 is obvious. Also, for fixed S, the order of magnitude of the smallest M satisfying (a), (b), (c) above may be easily computed by linear algebra. Consequently, Theorem 3 gives rise to an obvious algorithm that solves Problem 1 with work CN^k. Since k is a large constant determined by m and n, the obvious algorithm does far too much work. Nevertheless, ideas in the proof of Theorem 3 will play a crucial role here and in [20].

The proof of Theorem 3 in [14] is based on the study of a certain family of convex sets $\mathcal{K}_f(x, k, M) \subset \mathcal{P}$. The sets $\mathcal{K}_f(x, k, M)$ help greatly in understanding $\|f\|_{C^m(E,\sigma)}$, but they are hard to compute.

Here, we will introduce another family of convex sets $\Gamma(x, \ell, M)$, just as useful as the $\mathcal{K}_f(x, k, M)$, but much easier to compute. In this paper, we define $\Gamma(x, \ell, M)$, prove its basic properties, show how to compute it, and use it to solve Problem 1 by quoting Theorem 3. In [20] we will solve Problem 2, by adapting the proof of Theorem 3, with $\Gamma(x, \ell, M)$ in place of $\mathcal{K}_f(x, k, M)$. We explain this idea further in Section 9.

Our results, here and in [20], imply a refinement of Theorem 3 that allows an alternate computation of the order of magnitude of $\|f\|_{C^m(E,\sigma)}$.

Theorem 4. *Given $m, n \geq 1$, there exist k, C, depending only on m and n, for which the following holds.*

Let $E \subset \mathbb{R}^n$ be finite, and let $\sigma : E \to [0,\infty)$. Suppose $\#(E) = N$. Then there exists a collection $\mathcal{S}$, consisting of subsets $S \subset E$, with the following properties:

(A) *Each $S \in \mathcal{S}$ satisfies $\#(S) \le k$.*

(B) *The number of distinct $S \in \mathcal{S}$ is at most CN.*

(C) *Let $f : E \to \mathbb{R}$, and let $M > 0$. Assume that for each $S \in \mathcal{S}$ there exists a map $y \mapsto P^y$, from S into $\mathcal{P}$, satisfying* (a), (b), (c) *from Theorem* 3. *Then $\|f\|_{C^m(E,\sigma)} \le CM$.*

(D) *Moreover, the collection $\mathcal{S}$ can be computed from E, σ, using at most $CN \log N$ work and at most CN storage.*

Thus, instead of examining all k-element subsets $S \subset E$ as in Theorem 3, it is enough to examine the $O(N)$ subsets S that belong to $\mathcal{S}$.

Consequently, we may compute the order of magnitude of $\|f\|_{C^m(E,\sigma)}$ as follows. Given E and σ, we perform $O(N \log N)$ one-time work to produce the collection $\mathcal{S}$. Having found $\mathcal{S}$, we can compute the order of magnitude of $\|f\|_{C^m(E,\sigma)}$ for any given $f : E \to \mathbb{R}$, with work $O(N)$. The total storage needed is $O(N)$. The proof of Theorem 4 is given in [21].

In the special case $m = 1, \sigma = 0$, Problem 1 amounts to computing the order of magnitude of the Lipschitz norm of a given function defined on a finite subset of $\mathbb{R}^n$. An essentially optimal solution of this problem is contained in the work of Callahan and Kosaraju [11] and Har-Peled and Mendel [23]. The paper [11] strongly influenced ours, as the reader will see below. We would like to thank A. Naor for pointing us to the relevant computer science literature, and to A. Razborov for useful discussions on models of computation, which we will treat further in [20]. We are grateful to A. Tsao, who brought to our attention the practical problem of fitting a smooth surface to N given points.

We are grateful also to Gerree Pecht for LaTeXing this article to ever-impeccable "Gerree standards".

2. The plan

First, we present a "pedagogical algorithm" that solves Problem 1 with running time CN^2; then we give a more sophisticated variant with running time $CN \log N$.

The idea behind our pedagogical algorithm starts with the following elementary remarks. Suppose $F \in C^m(\mathbb{R}^n)$. Let $\bar{M} > 0$ be an upper bound for $\|F\|_{C^m(\mathbb{R}^n)}$, and suppose that $|F(x) - f(x)| \le \bar{M}\sigma(x)$ for all $x \in E$. Let $x \in E$, and let $P = J_x F$ denote the $(m-1)^{\underline{\text{rst}}}$ degree Taylor polynomial of F at the point x. Then,

(1) $$|\partial^\alpha P(x)| \leq \bar{M} \text{ for } |\alpha| \leq m-1; \text{ and } |P(x) - f(x)| \leq \bar{M}\sigma(x).$$

Moreover, suppose $x' \in E$ is another point, and let $P' = J_{x'}F$, the $(m-1)^{\underline{\text{rst}}}$ degree Taylor polynomial of F at x'. Then Taylor's theorem gives

(2) $$|\partial^\alpha (P - P')(x)| \leq C\bar{M}|x - x'|^{m-|\alpha|} \text{ for } |\alpha| \leq m-1\,.$$

Here and below, C denotes a constant depending only on m and n.

To exploit these remarks, we recall that $\mathcal{P}$ was defined to be the vector space of all (real-valued) $(m-1)^{\underline{\text{rst}}}$ degree polynomials on $\mathbb{R}^n$; and we define a family of (possibly empty) convex subsets $\Sigma(x, \ell, M) \subset \mathcal{P}$ by the following induction on ℓ:

For $\ell = 1$, $x \in E$, $M \in (0, \infty)$, we define

(3) $$\Sigma(x,1,M) = \{P \in \mathcal{P} : |\partial^\alpha P(x)| \leq M \text{ for } |\alpha| \leq m-1, \text{ and } |P(x) - f(x)| \leq M\sigma(x)\}.$$

For $\ell \geq 1$, suppose we have already defined the sets $\Sigma(x, \ell, M)$ for all $x \in E$ and $M \in (0, \infty)$. Then, for any $x \in E$ and $M \in (0, \infty)$, we define

(4) $\Sigma(x, \ell+1, M) = \{P \in \mathcal{P}$: For each $x' \in E$, there exists $P' \in \Sigma(x', \ell, M)$, such that $|\partial^\alpha(P - P')(x)| \leq M|x - x'|^{m-|\alpha|}$ for $|\alpha| \leq m-1\}$.

Then observations (1),(2), and an obvious induction on ℓ, show that $J_xF \in \Sigma(x, \ell, C\bar{M})$ for each $x \in E, \ell \geq 1$. In particular,

(5) Whenever $M \geq C \cdot \|f\|_{C^m(E,\sigma)}$, we have $\Sigma(x, \ell, M) \neq \phi$ for all ℓ, x. (As usual, ϕ denotes the empty set.)

Conversely, for an $\ell_* \geq 1$, depending only on m and n, the following holds:

(6) Let $M > 0$, and suppose that $\Sigma(x, \ell_*, M) \neq \phi$ for all $x \in E$. Then $\|f\|_{C^m(E,\sigma)} \leq C \cdot M$.

We will prove (6) in Section 8 below, by reducing it to Theorem 3 from Section 1. From (5) and (6), we see that

(7) $\|f\|_{C^m(E,\sigma)}$ has the same order of magnitude as

$$\inf\{M > 0 : \Sigma(x, \ell_*, M) \neq \phi \text{ for each } x \in E\}.$$

The idea of our pedagogical algorithm is to compute an approximation to the size and shape of the convex sets $\Sigma(x, \ell_*, M)$ by an adaptation of the induction (3), (4), and then to read off the order of magnitude of $\|f\|_{C^m(E,\sigma)}$ from (7). In the next sections, we explain more precisely what this means, and how to carry it out.

3. The data structures

In this section, we define the basic data structures used to specify the "approximate size and shape" of the convex sets $\Sigma(x, \ell, M)$ and of similar sets. Let V be a finite-dimensional (real) vector space. A "blob" in V is a family $\mathcal{K} = (K_M)_{M>0}$ of (possibly empty) convex subsets $K_M \subseteq V$, parametrized by $M \in (0, \infty)$, such that $M < M'$ implies $K_M \subseteq K_{M'}$. The "onset" of a blob $\mathcal{K} = (K_M)_{M>0}$ is defined as the infimum of all the $M > 0$ for which $K_M \neq \phi$. (If all K_M are empty, then onset $\mathcal{K} = +\infty$.)

For fixed $x \in E, \ell \geq 1$, the family of sets $(\Sigma(x, \ell, M))_{M>0}$ from the previous section forms a blob in $\mathcal{P}$, which we call $\Sigma(x, \ell)$. In the language of blobs, the fundamental result (7) from the previous section becomes

(0)

$\|f\|_{C^m(E,\sigma)}$ has the same order of magnitude as $\max\{\text{onset } \Sigma(x, \ell_*) : x \in E\}$.

Suppose $\mathcal{K} = (K_M)_{M>0}$ and $\mathcal{K}' = (K'_M)_{M>0}$ are blobs in V, and let $C \geq 1$ be a constant. We say that $\mathcal{K}$ and $\mathcal{K}'$ are "C-equivalent" if they satisfy $K_M \subseteq K'_{CM}$ and $K'_M \subseteq K_{CM}$ for all $M \in (0, \infty)$. Note that, if $\mathcal{K}$ and $\mathcal{K}'$ are C_1-equivalent, and if $\mathcal{K}'$ and $\mathcal{K}''$ are C_2-equivalent, then $\mathcal{K}$ and $\mathcal{K}''$ are $C_1 \cdot C_2$-equivalent. Note also that, if $\mathcal{K}$ and $\mathcal{K}'$ are C-equivalent, then

$$(1/C) \cdot \text{onset } \mathcal{K} \leq \text{onset } \mathcal{K}' \leq C \cdot \text{onset } \mathcal{K}.$$

Rather than dealing with the exact blobs $\Sigma(x, \ell)$, we will be satisfied with having a computerized representation of blobs which are C_ℓ-equivalent to $\Sigma(x, \ell)$, for a constant C_ℓ that depends solely on ℓ, m and n. There are several data structures that suit our needs. Two of these, that are not necessarily optimal from practical aspects, are described here.

The first class of blobs we focus attention on, consists of those given by "Approximate Linear Algebra Problems", or "ALPs". To define these, let $\lambda_1, \ldots, \lambda_{\mathrm{L}}$ be (real) linear functionals on V, let $b_1, \ldots, b_{\mathrm{L}}$ be real numbers, let $\sigma_1, \ldots, \sigma_{\mathrm{L}}$ be nonnegative real numbers, and let $M_* \in [0, +\infty]$. We call

$$\mathcal{A} = [(\lambda_1, \ldots, \lambda_{\mathrm{L}}), (b_1, \ldots, b_{\mathrm{L}}), (\sigma_1, \ldots, \sigma_{\mathrm{L}}), M_*] \tag{1}$$

an "ALP" in V. With $\mathcal{A}$ given by (1), we define a blob $\mathcal{K}(\mathcal{A}) = (K_M(\mathcal{A}))_{M>0}$ in V, by setting

$$K_M(\mathcal{A}) = \{v \in V : |\lambda_\ell(v) - b_\ell| \leq M \cdot \sigma_\ell \text{ for } \ell = 1, \ldots, L\} \tag{2}$$

for $M \geq M_*$, and

$$K_M(\mathcal{A}) = \phi \text{ for } M < M_*. \tag{3}$$

(Our definition (2) motivates the use of the phrase "approximate linear algebra problem".) We allow $L = 0$ in (1), in which case (2) says simply that $K_M(\mathcal{A}) = V$ for $M \geq M_*$.

An “ALP” is intermediate in generality between a linear algebra problem and a linear programming problem. Unlike a general blob, an ALP is specified by finitely many (real) parameters, and may therefore be manipulated by algorithms. ALPs will be of great use in [20], where their implementation details will be discussed.

In this paper, we will mostly work with the “Ellipsoidal Blobs”, which are blobs defined by certain linear and quadratic constraints. For our purposes, there is no difference at all between ALPs and ellipsoidal blobs - except for the fact that the implementation of ellipsoidal blobs requires the operation of a square-root of a positive real number, while ALPs need only addition, subtraction, multiplication and division. The algorithms presented here may be easily converted to ALPs (as is in fact described in [20]), and our results here are valid also in the restricted model of computation, without square-roots. The reasons we discuss ellipsoidal blobs here, are that the basic operations on them are easier to describe, that they are perhaps more useful from a practical viewpoint, and that the competing, elementary, ALPs will be discussed in [20].

To define ellipsoidal blobs, let $0 \leq L \leq \dim V$ be an integer, $\lambda_1, \dots, \lambda_{\mathrm{L}}$ be (real) linear functionals on V, $b, b_1, \dots, b_{\mathrm{L}}$ nonnegative numbers, $x_0 \in V$ and let $q : V \to [0, \infty)$ be a nonnegative quadratic form. We call a blob of the form $\mathcal{K} = (K_M)_{M>0}$ for

$$K_M = \{v \in V : q(v - x_0) + b \leq M^2,\ \lambda_\ell(v) = b_\ell \text{ for } \ell = 1, \dots, L\} \tag{4}$$

an ellipsoidal blob in V. Note that possibly $K_M = V$ for all $M > 0$, and possibly $K_M = \phi$ for all $M > 0$. An ellipsoidal blob is specified by no more than $2(D+1)^2$ real parameters, with $D = \dim V$; thus the amount of storage it consumes is just a constant depending on D. Computing the onset of an ellipsoidal blob is a standard linear-algebra task, that may be performed in CD^3 operations.

To “compute the approximate size and shape” of the $\Sigma(x, \ell, M)$, and of other convex sets, we will exhibit ellipsoidal blobs, which are C_ℓ-equivalent to the blobs induced by those sets, for a constant C_ℓ depending only on ℓ, m and n. We will construct such ellipsoidal blobs in Sections 4 and 6. Here, we prepare the way by discussing two elementary operations on blobs in general, and on ellipsoidal blobs in particular, that will be used in the algorithm.

First, suppose that $\mathcal{K}^\nu = (K^\nu_M)_{M>0}$ is a blob in V, for each $\nu = 1, 2, \dots, N$. Then we define the intersection $\mathcal{K}^1 \cap \dots \cap \mathcal{K}^N$ by setting

$$\mathcal{K}^1 \cap \dots \cap \mathcal{K}^N = (K^1_M \cap \dots \cap K^N_M)_{M>0}\,.$$

If each $\mathcal{K}^\nu$ is an ellipsoidal blob, then the intersection need not be an ellipsoidal blob. Recall that $D = \dim V$. In this section, C, C', etc. denote constants that depend only on D. Next, we will describe an algorithm for computing, in CN operations, an ellipsoidal blob which is $8D$-equivalent to $\mathcal{K}^1 \cap \dots \cap \mathcal{K}^N$.

Assume that the ellipsoidal blob $\mathcal{K}^\nu$ is specified by $0 \leq L_\nu \leq D$, linear functionals $\lambda_1^\nu, \ldots, \lambda_{L_\nu}^\nu$, real numbers $b^\nu, b_1^\nu, \ldots, b_{L_\nu}^\nu$, a vector $x_0^\nu \in V$ and a nonnegative quadratic form q^ν. By Gauss elimination, we can compute in $C'N$ computer operations, an integer $0 \leq L \leq D$, linear functionals $\lambda_1, \ldots, \lambda_{\mathrm{L}}$ and nonnegative real numbers $b_1, \ldots, b_{\mathrm{L}}$ such that

$$(4') \quad K_M^1 \cap \cdots \cap K_M^N = \{v \in V \ : \ \max_{\nu=1,\ldots,N} q^\nu(v - x_0^\nu) + b^\nu \leq M^2, \\ \lambda_1(v) = b_1, \ldots, \lambda_{\mathrm{L}}(v) = b_{\mathrm{L}}\}.$$

We fix some nondegenerate scalar product $\langle \cdot, \cdot \rangle$ in V. For each ν, we find vectors $e_1^\nu, \ldots, e_D^\nu \in V$ such that

$$\forall v \in V, \qquad q^\nu(v) = \Sigma_{i=1}^D \langle v, e_i^\nu \rangle^2.$$

This may be done, using standard linear algebra in $\tilde{C}N$ operations. Note that some of the e_i^ν may be zero. Then, for any $v \in V$ and $\nu = 1, \ldots, N$,

$$(5) \qquad \max_{i=1,\ldots,D} \langle v - x_0^\nu, e_i^\nu \rangle^2 \leq q^\nu(v - x_0^\nu) \leq D \max_{i=1,\ldots,D} \langle v - x_0^\nu, e_i^\nu \rangle^2.$$

Let $\mathcal{K}' = (K'_M)_{M>0}$ be defined so that $K'_M = \emptyset$ for $M < \max_{\nu=1,\ldots,N} \sqrt{b^\nu}$, and

$$K'_M = \left\{ v \in V \ : \ \max_{\substack{i=1,\ldots,D \\ \nu=1,\ldots,N}} |\langle v - x_0^\nu, e_i^\nu \rangle| \leq M, \ \lambda_1(v) = b_1, \ldots, \lambda_{\mathrm{L}}(v) = b_{\mathrm{L}} \right\}$$

for $M \geq \max_{\nu=1,\ldots,N} \sqrt{b^\nu}$. Then by (4′), (5) and easy algebra, the blob $\mathcal{K}'$ is $2\sqrt{D}$-equivalent to $\mathcal{K}^1 \cap \ldots \cap \mathcal{K}^N$. Next, we will use Megiddo's linear programming algorithm [27] to compute the minimal M such that $K'_M \neq \emptyset$, to be denoted by M^*. Using the same linear programming algorithm, we may also compute a point $x_0 \in V$ such that $x_0 \in K'_{M^*}$. The number of operations required is CN. Let $\mathcal{K}'' = (K''_M)_{M>0}$ be the blob such that $K''_M = \emptyset$ for $M < M^*$, while for $M \geq M^*$,

$$K''_M = \left\{ v \in V \ : \ \max_{\substack{i=1,\ldots,D \\ \nu=1,\ldots,N}} |\langle v - x_0, e_i^\nu \rangle| \leq M, \ \lambda_1(v) = b_1, \ldots, \lambda_{\mathrm{L}}(v) = b_{\mathrm{L}} \right\}$$

(we replaced x_0^ν with x_0). It is easy to verify that $\mathcal{K}''$ is 2-equivalent to $\mathcal{K}'$ and hence $\mathcal{K}''$ is $4\sqrt{D}$-equivalent to $\mathcal{K}^1 \cap \ldots \cap \mathcal{K}^N$. Let $T \subset V$ denote the convex hull of $\{\pm e_i^\nu \ ; \ \nu = 1, \ldots, N, \ i = 1, \ldots, D\}$. Next, we will compute the Löwner-John ellipsoid of T, which is the ellipsoid of minimal volume that contains T. In [12] an algorithm for that purpose is described, based on Megiddo's algorithm [27]. Denote by W the subspace spanned by the vectors e_i^ν ($\nu = 1, \ldots, N, \ i = 1, \ldots, D$), and let $\pi_W : V \to W$ be the orthogonal projection onto W, to be computed within CN operations. The algorithm in [12] finds in $C'N$ operations, the unique positive definite linear operator $A : W \to W$ that maximizes $\det(A)$ under the linear constraints

$$\{\langle Ae_i^\nu, e_i^\nu \rangle \leq 1 : i = 1, .., D, \nu = 1, \ldots, N\}.$$

We now define a nonnegative quadratic form $q : V \to [0,\infty)$ by setting $q(v) = \langle A^{-1}\pi_W v, v\rangle$. According to a well-known theorem of John [24], for any $v \in V$,

$$\max_{\substack{i=1,\dots,D\\ \nu=1,\dots,N}} \langle v, e_i^\nu\rangle^2 \le q(v) \le D \max_{\substack{i=1,\dots,D\\ \nu=1,\dots,N}} \langle v, e_i^\nu\rangle^2. \tag{6}$$

We now define the ellipsoidal blob $\mathcal{K} = (K_M)_{M>0}$, by setting

$$K_M = \left\{v \in V \ : \ q(v - x_0) + b \le M^2,\ \lambda_1(v) = b_1, \dots, \lambda_{\mathrm{L}}(v) = b_{\mathrm{L}}\right\}$$

where b is chosen so that onset $\mathcal{K} = M^*$. By (6), the ellipsoidal blob $\mathcal{K}$ is $2\sqrt{D}$-equivalent to $\mathcal{K}''$, and hence $\mathcal{K}$ is $8D$-equivalent to $\mathcal{K}^1 \cap \cdots \cap \mathcal{K}^N$. Thus, we have described an algorithm that computes, in CN operations, an ellipsoidal blob $\mathcal{K}$ which is $8D$-equivalent to $\mathcal{K}^1 \cap \cdots \cap \mathcal{K}^N$.

The second operation we need is the computation of the Minkowski sum. Recall that for $A, B \subset V$, the Minkowski sum of A and B is defined as

$$A + B = \{a + b : a \in A, b \in B\}.$$

Suppose that $\mathcal{K}^\nu = (K_M^\nu)_{M>0}$ is a blob in V, for $\nu = 1, 2$. We define the Minkowski sum $\mathcal{K}^1 + \mathcal{K}^2$ as

$$\mathcal{K}^1 + \mathcal{K}^2 = (K_M^1 + K_M^2)_{M>0}.$$

Suppose now that $\mathcal{K}^1, \mathcal{K}^2$ are ellipsoidal blobs in V. The ellipsoidal blob $\mathcal{K}^\nu$ is specified by $0 \le L_\nu \le D$, linear functionals $\lambda_1^\nu, \dots, \lambda_{L_\nu}^\nu$, real numbers $b^\nu, b_1^\nu, \dots, b_{\mathrm{L}}^\nu$, a vector $x_0^\nu \in V$ and a nonnegative quadratic form q^ν. Then,

$$K_M^1 + K_M^2 = \{v_1 + v_2 \ : \ v_1, v_2 \in V,\ \max\{q^1(v_1 - x_0^1) + b^1, q^2(v_2 - x_0^2) + b^2\} \le M^2,$$
$$\lambda_i^\nu(v_\nu) = b_i^\nu \text{ for } \nu = 1, 2 \text{ and } 1 \le i \le L_\nu\}.$$

We define a blob $\mathcal{K}' = (K'_M)_{M>0}$ by

$$K'_M = \{v_1 + v_2 \ : \ v_1, v_2 \in V,\ q^1(v_1 - x_0^1) + q^2(v_2 - x_0^2) + b^1 + b^2 \le M^2, \\ \lambda_i^\nu(v_\nu) = b_i^\nu \text{ for } \nu = 1, 2 \text{ and } 1 \le i \le L_\nu\}; \tag{7}$$

i.e. we replace the maximum with a sum. The blob $\mathcal{K}'$ is $\sqrt{2}$-equivalent to $\mathcal{K}^1 + \mathcal{K}^2$. Furthermore, $\mathcal{K}'$ is actually an ellipsoidal blob, although (7) is not its standard description as such. The linear functionals, real parameters and nonnegative quadratic form needed for the standard representation of $\mathcal{K}'$ as an ellipsoidal blob, may be computed in C operations using straightforward linear algebra. We omit the details.

This completes our discussion on the implementation of ellipsoidal blobs. Note that the above operations on blobs behave well with respect to C-equivalence. In fact, assume that $\mathcal{K}_1, \dots, \mathcal{K}_N$ are blobs in V which are C-equivalent to $\mathcal{K}'_1, \dots, \mathcal{K}'_N$, respectively. Then trivially $\mathcal{K}_1 \cap \cdots \cap \mathcal{K}_N$ is C-equivalent to $\mathcal{K}'_1 \cap \cdots \cap \mathcal{K}'_N$ and $\mathcal{K}_1 + \mathcal{K}_2$ is C-equivalent to $\mathcal{K}'_1 + \mathcal{K}'_2$.

4. A pedagogical algorithm

In this section we will sketch a simpler, nonoptimal, algorithm for computing $\|f\|_{C^m(E,\sigma)}$, that arises from the induction (3), (4) in Section 2. We start with reformulating (3), (4) from Section 2 in the language of blobs. Denote $D = \dim(\mathcal{P})$. For $x \in E$, the blob $\Sigma(x, 1)$ is $\sqrt{D+1}$-equivalent to the blob $\mathcal{A}'_{x,1} = (A'_{x,1,M})_{M>0}$, defined by

$$A'_{x,1,M} = \left\{ P \in \mathcal{P} : \Sigma_{|\alpha| \le m-1} |\partial^\alpha P(x)|^2 + \left(\frac{|P(x) - f(x)|}{\sigma(x)} \right)^2 \le M^2 \right\}$$

in the case $\sigma(x) \neq 0$, and

$$A'_{x,1,M} = \left\{ P \in \mathcal{P} : \Sigma_{|\alpha| \le m-1} |\partial^\alpha P(x)|^2 \le M^2, \quad P(x) = f(x) \right\}$$

in the case $\sigma(x) = 0$. In both cases, the blob $\mathcal{A}'_{x,1}$ is clearly an ellipsoidal blob in $\mathcal{P}$.

We will make frequent use of the following blobs: For any $x \in \mathbb{R}^n, \delta \ge 0$ we set $\mathcal{B}(x, \delta) = (B(x, \delta, M))_{M>0}$, where

$$(1) \qquad B(x, \delta, M) = \left\{ P \in \mathcal{P} : |\partial^\alpha P(x)| \le M \delta^{m-|\alpha|} \text{ for } |\alpha| \le m-1 \right\}.$$

We will approximate $\mathcal{B}(x, \delta)$ with an ellipsoidal blob $\mathcal{B}'(x, \delta) = (B'(x, \delta, M))_{M>0}$, defined as

$$(2) \qquad B'(x, \delta, M) = \left\{ P \in \mathcal{P} : \Sigma_{|\alpha| \le m-1} \left(\frac{|\partial^\alpha P(x)|}{\delta^{m-|\alpha|}} \right)^2 \le M^2 \right\}.$$

(If $\delta = 0$, then we just set $B'(x, \delta, M) = \{0\}$; this yields an ellipsoidal blob.) The ellipsoidal blob $\mathcal{B}'(x, \delta)$ is $\sqrt{D}$-equivalent to $\mathcal{B}(x, \delta)$, for $x \in \mathbb{R}^n, \delta \ge 0$. Recall the definition (4) from Section 2, according to which, $P \in \Sigma(x, \ell+1, M)$ if and only if $P \in \Sigma(x, \ell, M)$, and for each $y \in E \setminus \{x\}$, there is $P' \in \Sigma(y, \ell, M)$ with $P - P' \in B(x, |x-y|, M)$. Therefore, relation (4) from Section 2 translates to

$$(3) \qquad \Sigma(x, \ell+1) = \Sigma(x, \ell) \cap \bigcap_{y \in E \setminus \{x\}} \left[\Sigma(y, \ell) + \mathcal{B}(x, |x-y|) \right].$$

We will show by induction on ℓ, that it is possible to compute for each $x \in E$, an ellipsoidal blob $\mathcal{A}'_{x,\ell}$ that is C_ℓ-equivalent to $\Sigma(x, \ell)$, for some constant C_ℓ depending only on ℓ, m and n. This holds for $\ell = 1$. Assume validity for ℓ, and let us prove it for $\ell+1$. Fix $x \in E$. Using the operations on ellipsoidal blobs discussed in Section 3, we may compute, in CN operations, an ellipsoidal blob $\mathcal{A}'_{x,\ell+1}$ that is $8\sqrt{2}D$-equivalent to

$$(4) \qquad \mathcal{A}'_{x,\ell} \cap \bigcap_{y \in E \setminus \{x\}} \left[\mathcal{A}'_{y,\ell} + \mathcal{B}'(x, |x-y|) \right].$$

Indeed, (4) involves the computation of $N-1$ Minkowski sums and the computation of an intersection of N ellipsoidal blobs. This requires CN operations, and creates an ellipsoidal blob that is $8\sqrt{2}\ D$-equivalent to the blob (4). Recall that $D = \dim(\mathcal{P})$ depends solely on m and n. By (3) and the induction hypothesis, the ellipsoidal blob $\mathcal{A}'_{x,\ell+1}$ is $C_{\ell+1}$-equivalent to the blob $\Sigma(x, \ell+1)$, for some constant $C_{\ell+1}$ depending only on ℓ, m and n.

The computation of $\mathcal{A}'_{x,\ell+1}$ for a single $x \in E$ requires CN operations. Given $\{\mathcal{A}'_{x,\ell}\}_{x\in E}$, the computation of $\{\mathcal{A}'_{x,\ell+1}\}_{x\in E}$ thus requires a total of CN^2 operations. We conclude that the total amount of work needed to compute the ellipsoidal blobs $\mathcal{A}'_{x,\ell_*}$ for all $x \in E$, is bounded by CN^2, where C is a constant that depends solely on m and n. (Recall that ℓ_* is the constant, depending on m and n, from (6) of Section 2.) By computing $\max\{\text{onset } \mathcal{A}'_{x,\ell_*} : x \in E\}$, in CN operations, we obtain a number that has the same order of magnitude as $\max\{\text{onset } \Sigma(x, \ell_*) : x \in E\}$ which in turn, by (6) of Section 2, has the same order of magnitude as $\|f\|_{C^m(E,\sigma)}$. This gives an inefficient algorithm, that computes the order of magnitude of $\|f\|_{C^m(E,\sigma)}$; we have obtained an algorithm that requires CN^2 operations, rather than $CN \log N$.

Let us carefully examine the algorithm just described. After the ℓ^{th} iteration, for each $x \in E$ we have computed an approximation to the blob $\Sigma(x, \ell)$, which represents candidate Taylor polynomials of the desired C^m function at x. At the next iteration we go over all other $\Sigma(y, \ell)$, for $y \neq x$, and eliminate from $\Sigma(x, \ell)$ the Taylor polynomials which are inconsistent with any of the $\Sigma(y, \ell)$. There is some redundancy here: Suppose that $y, z \in E$ are close points that are far away from x. Suppose also that the blob $\Sigma(y, \ell)$ is already consistent with $\Sigma(z, \ell)$, as sometimes happens. Then the information we collect from $\Sigma(y, \ell)$, and the information from $\Sigma(z, \ell)$ regarding Taylor polynomials at x, are roughly the same. Therefore, it is useless to compare $\Sigma(x, \ell)$ both with $\Sigma(y, \ell)$ and with $\Sigma(z, \ell)$; it is enough to consider only one of them. Savings may arise if we group the points in some geometric way, and compare only some of the pairs of the Σ's rather than all pairs. This idea will enable us to reduce the running time of the algorithm from CN^2 to $CN \log N$.

The basic technique we employ for grouping the points in clusters is the Callahan-Kosaraju decomposition from computer science (see [11]). In the next section we summarize the results of Callahan and Kosaraju.

5. Callahan-Kosaraju decomposition

Some notation is needed. For $A, B \subset \mathbb{R}^n$ and $\kappa > 0$, we say that A and B are "κ-separated" if

$$\max\{\text{diameter}(A), \text{diameter}(B)\} < \kappa \text{ distance}(A, B)$$

where, of course, $\text{diameter}(A) = \sup_{x,y\in A} |x-y|$ and $\text{distance}(A, B) = \inf_{x\in A, y\in B} |x-y|$.

A "proper box" in $\mathbb{R}^n$ is a Cartesian product of intervals $Q = I_1 \times \cdots \times I_n \subset \mathbb{R}^n$. Here, each I_j may be open, closed, half-open, or a single point, but it must be bounded and nonempty. For each of the I_j of nonzero length, we may "bisect" Q into two proper boxes Q_{L} and Q_{R} in an obvious way, by bisecting I_j into a left half I_j^L and a right half I_j^R. (To avoid ambiguity, we place the midpoint of I_j in I_j^R.) We say that $\{Q_{\mathrm{L}}, Q_{\mathrm{R}}\}$ forms a "proper bisection" of Q. There are at most n proper bisections of a given proper box in $\mathbb{R}^n$.

A "cell" will be a nonempty subset of E of the form

$$A = E \cap Q$$

for some proper box $Q \subset \mathbb{R}^n$. Recall that E is our set of input points. If A is a cell, then we denote by $Q(A)$ the smallest proper box that contains A, i.e., the intersection of all the proper boxes that contain A.

Let $\mathcal{T}$ be a collection of subsets of E. For $\Lambda \subset \mathcal{T}$ we write

$$\cup\Lambda = \bigcup_{A \in \Lambda} A = \{x : x \in A \text{ for some } A \in \Lambda\} .$$

Let $\mathcal{L}$ be a list of pairs (Λ_1, Λ_2) where Λ_1, Λ_2 are families of subsets of E that belong to $\mathcal{T}$, i.e., $\Lambda_1, \Lambda_2 \subset \mathcal{T}$. For $\kappa > 0$, we say that $(\mathcal{T}, \mathcal{L})$ is a "Callahan-Kosaraju decomposition of E with constant κ", or a "κ-CK decomposition" for short, if the following hold:

(1) $\bigcup_{(\Lambda_1,\Lambda_2)\in\mathcal{L}} (\cup\Lambda_1) \times (\cup\Lambda_2) = \{(x, y) : x, y \in E, x \neq y\}$.

(2) If (Λ_1, Λ_2) and (Λ_1', Λ_2') are distinct pairs in $\mathcal{L}$, then

$$[(\cup\Lambda_1) \times (\cup\Lambda_2)] \cap \left[(\cup\Lambda_1') \times (\cup\Lambda_2')\right] = \emptyset.$$

(3) $\cup\Lambda_1$ and $\cup\Lambda_2$ are κ-separated for any $(\Lambda_1, \Lambda_2) \in \mathcal{L}$.

(4) $\#(\mathcal{T}) < CN$ and $\#(\mathcal{L}) < CN$ where C is a constant depending solely on κ and n.

In the computer science literature a CK decomposition is called a "Well-Separated Pairs Decomposition" or "WSPD". A κ-CK decomposition as defined above is a pure mathematical object. A κ-CK decomposition $(\mathcal{T}, \mathcal{L})$ will be implemented in the computer, using a data structure that satisfies the following properties (in these properties the letter C stands for a constant depending only on n and κ):

(5) The amount of storage needed to hold the data structure is bounded by CN.

(6) The following tasks require $CN \log N$ operations and CN storage:

(6a) Go over all $A \in \mathcal{T}$, and for each A produce the list of elements of A.

(6b) Go over all $(\Lambda_1, \Lambda_2) \in \mathcal{L}$, and for each (Λ_1, Λ_2) produce the elements (in $\mathcal{T}$) of Λ_1 and of Λ_2.

(6c) Go over all $A \in \mathcal{T}$, and for each A produce the list of all $(\Lambda_1, \Lambda_2) \in \mathcal{L}$ such that $A \in \Lambda_1$.

(6d) Go over all $x \in E$, and for each $x \in E$ produce the list of $A \in \mathcal{T}$ such that $x \in A$.

Producing a list of elements in $\mathcal{L}, \mathcal{T}$ or E, means outputting a list of indices or pointers to the corresponding elements. The algorithms in (6) will have the following interface: The first time we invoke one of them, say (6a), the output will be some node $A \in \mathcal{T}$ together with a list of its elements. The next time we call (6a), the output will be another $A' \in \mathcal{T}$, together with the elements of A', and so on, until the algorithm in (6a) signals that the entire collection $\mathcal{T}$ was exhausted. We are guaranteed that the algorithm will go over all $\mathcal{T}$, and exactly once over each $A \in \mathcal{T}$. The total amount of time that the algorithm in (6a) requires, to go over the whole collection $\mathcal{T}$, is $CN \log N$. The amount of storage that the algorithm requires does not exceed CN at any given moment, i.e., when computing the elements of a set $A \in \mathcal{T}$, we do not need more than CN storage.

The algorithms in (6) use no more than $CN \log N$ computer operations. This includes, of course, the time needed to prepare the output. In particular, it follows from (6) that

$$\sum_{(\Lambda_1,\Lambda_2)\in\mathcal{L}} (\#(\Lambda_1) + \#(\Lambda_2)) < CN \log N, \qquad \sum_{A\in\mathcal{T}} \#(A) < CN \log N. \tag{7}$$

For a subset $A \subset \mathbb{R}^n$, set

$$\operatorname{diam}_\infty(A) = \sqrt{n} \sup_{x,y\in A} \max_{1\le i\le n} |x_i - y_i|,$$

the ℓ_∞-diameter of A. (Here of course, x_i and y_i denote the i^{th} coordinates of x and y, respectively.) Note that $\operatorname{diam}_\infty(A) \ge \operatorname{diameter}(A) \ge \frac{\operatorname{diam}_\infty(A)}{\sqrt{n}}$ for any $A \subset \mathbb{R}^n$. We will implement the data structure that holds the CK-decomposition of E such that,

(8) Given $A \in \mathcal{T}$ we can compute $\operatorname{diam}_\infty(A)$ within C operations.
Given (Λ_1, Λ_2) we compute $\operatorname{diam}_\infty(\cup\Lambda_1)$ and $\operatorname{diam}_\infty(\cup\Lambda_2)$ in C operations.

The following theorem is an adaptation of [11, Th. 4.2], and is basically an expansion of the remark after Theorem 5.3 in [11].

Theorem 5. *There exists an algorithm, whose inputs are a parameter $\kappa > 0$ and a subset $E \subset \mathbb{R}^n$ with $\#(E) = N$, and that outputs a κ-CK decomposition $(\mathcal{T}, \mathcal{L})$ of E such that conditions* (1), …, (8) *are fulfilled. The* *algorithm performs no more than $CN \log N$ operations and uses no more than CN storage, where C is a constant that depends solely on the dimension n and on the parameter κ.*

Theorem 5 follows from the considerations in [11], together with some elementary computer programming tricks. We devote the rest of this section to the proof of Theorem 5, along the lines of [11].

Theorem 3.1 in [11] is the following result, which one might view as a Calderón-Zygmund type decomposition of E.

LEMMA 1. *There exists an algorithm, whose input consists of the set $E \subset \mathbb{R}^n$, and that outputs a tree T with the following properties*:

(a) *The tree T is a binary tree. A node in T has either two children ("an internal node") or no children at all ("a leaf").*

(b) *Each node of T corresponds to a cell. We do not distinguish here between a node of T and the cell $A \subset E$ to which it corresponds.*

(c) *The leaves of T are cells which are singletons. For each $x \in E$ there is a unique leaf in T which is the cell $\{x\}$.*

(d) *The root of T is the entire set E.*

(e) *Let A be an internal node. Then there exists a proper bisection of $Q(A)$, to be denoted $\{Q_{\rm L}, Q_{\rm R}\}$, such that the children of A are*

$$A_{\rm L} = E \cap Q_{\rm L} \ \text{ and } \ A_{\rm R} = E \cap Q_{\rm R}\,.$$

The running time of the algorithm is no more than $CN \log N$, and the storage is no more than CN, where C depends only on the dimension n.

The proof of Lemma 1 appears in [11]. Note that the tree constructed in Lemma 1 is not balanced, i.e. it might have branches whose length is much larger than $\log N$. Next, we quote the remarkable Theorem 4.2 from [11].

THEOREM 6. *Given a set $E \subset \mathbb{R}^n$ with $\#(E) = N$, a parameter $\kappa > 0$, and a tree T that satisfies* (a)–(e), *there is an algorithm that constructs a set L with the following properties*:

(1′) *The elements of L are pairs (A, B) where A, B are nodes of T.*

(2′) $\bigcup_{(A,B)\in L} A \times B = \{(x, y) : x, y \in E, x \neq y\}$.

(3′) *If (A_1, B_1) and (A_2, B_2) are distinct elements of L then $(A_1 \times B_1) \cap (A_2 \times B_2) = \emptyset$.*

(4′) *A and B are κ-separated for any $(A, B) \in L$.*

(5′) *$\#(L) < CN$ where C depends solely on n and κ.*

The algorithm terminates after no more than CN operations, where C depends only on the dimension n and the parameter κ. Obviously, the storage needed is just CN.

Proof of Theorem 5. Let T be the tree computed in Lemma 1. Assume that for any internal node A of T, the two children of A are labeled. One of them, denoted by A_{L} is the left child, and the other, A_{R}, is the right child. Recall that the leaves of T are the N singletons $\{x\}$ for $x \in E$. The tree T induces an order relation, called "inorder", on the leaves (see [25, §2.3.1]). Let us describe it briefly.

Given $x, y \in E, x \neq y$, consider the node A in the tree which is the least common ancestor of the leaves $\{x\}$ and $\{y\}$ in the tree. If $\{x\}$ is a descendant of A_{L} (and then $\{y\}$ is a descendant of A_{R}), we say that

$$x \prec y.$$

Otherwise, $y \prec x$. Then $\prec$ is an order relation. We order the N points of E according to $\prec$, and denote the resulting permutation by $y_1 \prec y_2 \prec \ldots \prec y_N$. This permutation may be computed in no more than CN operations and storage (see [25, §2.3.1]). We will store the permutation in memory; i.e.,

- In the data structure for holding the CK-decomposition of E, we will store an ordered list $y_1, \ldots, y_N$ of the elements of E.

Observe that any node in T is a cell of the form $\{y_\mu, y_{\mu+1}, \ldots, y_\nu\}$ for $\mu \leq \nu$. Let $\mathcal{T}$ denote the set of nodes of a completely balanced binary tree on $\{y_1, \ldots, y_N\}$, defined recursively as follows:

(i) The root of $\mathcal{T}$ is the set $\{y_1, \ldots, y_N\}$.

(ii) For any node $A = \{y_\mu, y_{\mu+1}, \ldots, y_\nu\}$, if $\mu = \nu$ then A will be a leaf. If $\mu < \nu$, then the children of A are

$$A_{\mathrm{L}} = \left\{y_\mu, \ldots, y_{\lfloor \frac{\mu+\nu}{2} \rfloor}\right\}, \qquad A_{\mathrm{R}} = \left\{y_{\lfloor \frac{\mu+\nu}{2} \rfloor + 1}, \ldots, y_\nu\right\}.$$

Then $\#(\mathcal{T}) \leq 2N-1$. Our data structure for holding the CK-decomposition of E will store in memory the following items.

- The tree $\mathcal{T}$.
- For each node $A = \{y_\mu, y_{\mu+1}, \ldots, y_\nu\}$ of $\mathcal{T}$, we will store the indices μ and ν.

For each $A \in \mathcal{T}$ and $1 \leq i \leq n$, we will calculate the numbers

$$m_i(A) = \min_{x \in A} x_i, \quad M_i(A) = \max_{x \in A} x_i,$$

that may be computed in CN operations in a standard, recursive manner. The computation of $\mathcal{T}$ requires no more than CN storage and operations, excluding the work needed for the creation of T in Lemma 1. Note also that the height of $\mathcal{T}$ is no greater than $\lceil \log_2 N \rceil + 1$, and that for any $x \in E$, there are at most $\lceil \log_2 N \rceil + 1$ nodes of $\mathcal{T}$ that contain x. The implementation of the task (6a) is very easy, and to implement (6d) we simply climb down the tree. We omit the details.

The next observation, is that any interval $I = \{y_\mu, y_{\mu+1}, \dots, y_\nu\}$ may be expressed as the union of at most $2\left(\lceil \log_2 N \rceil + 1\right)$ nodes of $\mathcal{T}$: These are exactly the nodes $B \in \mathcal{T}$ such that B is contained in the interval I, but the parent of B is not contained in the interval I. Given indices $\mu < \nu$, such a union may be computed in less than $C \log N$ operations, as follows: Compute A, the least common ancestor of $\{y_\mu\}$ and $\{y_\nu\}$ in $\mathcal{T}$ (in $C \log N$ operations). If $A = I = \{y_\mu, \dots, y_\nu\}$ then we are done. Otherwise, go over the path from $\{y_\mu\}$ to A in $\mathcal{T}$, starting from $\{y_\mu\}$ and excluding A. Let A' be the first node in the path which is not a left child of another node in the path. Note that $A' \subset I = \{y_\mu, \dots, y_\nu\}$, but the parent of A' is not contained in I. We mark the node A'.

Next, we go over the path from A' to A, excluding A' and A. For each node B in the path, if its right child B_{R} is not in the path, mark the child B_{R}. Note that the marked nodes are contained in I, yet their parents are not contained in I. Similarly, we go over the path from $\{y_\nu\}$ to A excluding A, and mark A'', the first node in the path which is not a right child of another node in the path. We then go over the nodes in the path from A'' to A excluding A'' and A, and for each node, we mark its left child if the left child is not in the path. It is straightforward to verify that $I = \{y_\mu, y_{\mu+1}, \dots, y_\nu\}$ is the disjoint union of the marked nodes; see Figure 2.

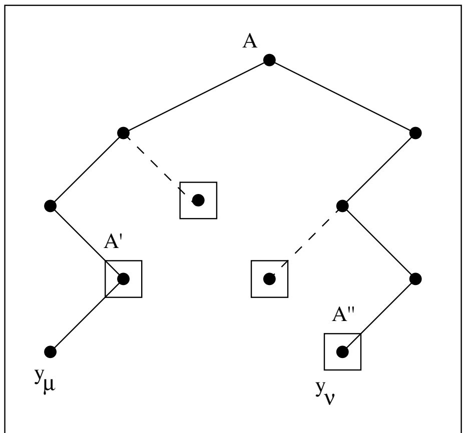

Figure 2

We invoke now the algorithm from Theorem 6, with the set E, the tree T and κ as inputs. The outputted list is denoted by L. We will construct a modified list $\mathcal{L}$ as follows: For every $(A, B) \in L$, we let Λ_1 and Λ_2 be the subsets of $\mathcal{T}$ whose unions equal A and B respectively, described in the previous paragraph. The list of all pairs (Λ_1, Λ_2) obtained that way is the desired set $\mathcal{L}$. Thus $\#(\mathcal{L}) < CN$ and $\#(\Lambda_1), \#(\Lambda_2) < C \log N$ for any $(\Lambda_1, \Lambda_2) \in \mathcal{L}$. Note that properties (1), ..., (4) follow from the corresponding properties of L. We will represent $\mathcal{L}$ in the data structure by storing the following items:

- Two lists of the elements of $\mathcal{L}$, one sorted according to the left endpoint of the interval $\cup\Lambda_1$, and one sorted according to the right endpoint of the interval $\cup\Lambda_1$.
- For each $(\Lambda_1, \Lambda_2) \in \mathcal{L}$ we store the indices of the endpoints of the intervals $\cup\Lambda_1, \cup\Lambda_2$.

Given $(\Lambda_1, \Lambda_2) \in \mathcal{L}$, we may compute in $C \log N$ time, the actual nodes of $\mathcal{T}$ that correspond to Λ_1, Λ_2, and also the numbers $\mathrm{diam}_\infty(\cup\Lambda_1), \mathrm{diam}_\infty(\cup\Lambda_2)$ (using $m_i(A), M_i(A)$ that were computed before). This completes the implementation of (6b). We will also store in our data structure implementing the CK-decomposition the following data, to be computed within $CN \log N$ operations:

- For each $(\Lambda_1, \Lambda_2) \in \mathcal{L}$, we store $\mathrm{diam}_\infty(\cup\Lambda_1), \mathrm{diam}_\infty(\cup\Lambda_2)$.
- For each $A \in \mathcal{T}$ we store $\mathrm{diam}_\infty(A)$.

Thus, the task (8) is easily implemented. Clearly, our data structure uses CN storage and (5) is satisfied. Thus, it only remains to implement (6c). To that end, we go over all $A \in \mathcal{T}$, except for the root (which is not contained in any Λ_1). Fix this $A \in \mathcal{T}$. Say A is the child of the node $B \in \mathcal{T}$. Without loss of generality, we may suppose that $A = B_{\mathrm{R}}$, the right child of B. We will use also B_{L}, the left child of B. We want to list all the $(\Lambda_1, \Lambda_2) \in \mathcal{L}$ such that $A \in \Lambda_1$.

Let $(\Lambda_1, \Lambda_2) \in \mathcal{L}$ be given. Recall that A, B_{L} and $\cup\Lambda_1$ are all identified with intervals. Call these intervals $[y_A^-, y_A^+]$, $[y_{B_{\mathrm{L}}}^-, y_{B_{\mathrm{L}}}^+]$ and $[y_{\Lambda_1}^-, y_{\Lambda_1}^+]$, respectively. Then it is straightforward to check that $A \in \Lambda_1$ if and only if

(i) $[y_A^-, y_A^+] \subset [y_{\Lambda_1}^-, y_{\Lambda_1}^+]$, and

(ii) $y_{B_{\mathrm{L}}}^- \prec y_{\Lambda_1}^-$, in the order defined above

(because (i) and (ii) are equivalent to (i′) $A \subset \cup\Lambda_1$ and (ii′) $B \not\subset \cup\Lambda_1$). Therefore, we may proceed as follows: Using binary search, we find all $(\Lambda_1, \Lambda_2) \in \mathcal{T}$ such that $y_{\Lambda_1}^- \in (y_{B_{\mathrm{L}}}^-, y_A^-]$. Among all these (Λ_1, Λ_2), we output only the $(\Lambda_1, \Lambda_2) \in \mathcal{T}$ such that $y_{\Lambda_1}^+$ is greater than or equal to y_A^+. Note that the

$(\Lambda_1, \Lambda_2) \in \mathcal{L}$ we output, are precisely those that satisfy (i) and (ii). Hence we produced all $(\Lambda_1, \Lambda_2) \in \mathcal{L}$ such that $A \in \Lambda_1$. The algorithm repeats this procedure for all nodes $A \in \mathcal{T}$ other than the root. Each $(\Lambda_1, \Lambda_2) \in \mathcal{L}$ was accessed at most $C \log_2 N$ times in the algorithm, as the number of nodes $B \in \mathcal{T}$ such that $y^-_{\Lambda_1} \in B$ or $y^+_{\Lambda_1} \in B$ is at most $2(\lceil \log_2 N \rceil + 1)$. Including the binary searches, the total amount of work of the algorithm for (6c) is $CN \log N$, and the storage is CN. This completes the proof. $\square$

6. The algorithm

In this section we describe the algorithm promised in Theorem 1. We are given a set $E \subset \mathbb{R}^n$ of size N, and functions $f : E \to \mathbb{R}$, $\sigma : E \to [0, \infty)$. We start by applying the algorithm from Theorem 5 to the set E with constant $\kappa = \frac{1}{2}$. The resulting κ-CK decomposition is denoted by $(\mathcal{T}, \mathcal{L})$. For each node $A \in \mathcal{T}$ we pick a representative, a point $x_A \in A$. We also select, for each $(\Lambda_1, \Lambda_2) \in \mathcal{L}$, points $x_{\Lambda_1} \in \cup\Lambda_1$, $x_{\Lambda_2} \in \cup\Lambda_2$. This is done within CN operations.

Rather than computing the blobs $\Sigma(x, \ell)$, we will work with *a priori* different blobs, to be denoted by $\Gamma(x, \ell) = (\Gamma(x, \ell, M))_{M>0}$. For $\ell = 1$ we set

$$\Gamma(x, 1, M) = \Sigma(x, 1, M) = \{P \in \mathcal{P} : |\partial^\alpha P(x)| \leq M \text{ for } |\alpha| \leq m-1, \text{ and } |P(x) - f(x)| \leq M\sigma(x)\}$$

for $x \in E$ and $M > 0$. Recall the definition of the blob $\mathcal{B}(x, \delta)$ defined in (1) of Section 4. Having defined $\{\Gamma(x, \ell)\}_{x \in E}$, we will now define $\Gamma(x, \ell+1)$ in five simple steps.

Step 1. For any node $A \in \mathcal{T}$ we form the blob

$$\Gamma(A, \ell) = (\Gamma(A, \ell, M))_{M>0} = \bigcap_{x \in A} \left[\Gamma(x, \ell) + \mathcal{B}(x, \mathrm{diam}_\infty(A))\right].$$

Step 2. For any $(\Lambda_1, \Lambda_2) \in \mathcal{L}, i = 1, 2$, we form the blob

$$\Gamma_i(\Lambda_i, \ell) = (\Gamma_i(\Lambda_i, \ell, M))_{M>0} = \bigcap_{A \in \Lambda_i} \left[\Gamma(A, \ell) + \mathcal{B}(x_A, \mathrm{diam}_\infty(\cup\Lambda_i))\right].$$

Step 3. For any $(\Lambda_1, \Lambda_2) \in \mathcal{L}$, we form the blob

$$\begin{aligned}\bar{\Gamma}(\Lambda_1, \Lambda_2, \ell) &= (\bar{\Gamma}(\Lambda_1, \Lambda_2, \ell, M))_{M>0} \\ &= \Gamma_1(\Lambda_1, \ell) \cap \left[\Gamma_2(\Lambda_2, \ell) + \mathcal{B}(x_{\Lambda_1}, |x_{\Lambda_1} - x_{\Lambda_2}|)\right].\end{aligned}$$

Step 4. For any node $A \in \mathcal{T}$ we form the blob

$$\Gamma'(A, \ell+1) = (\Gamma'(A, \ell+1, M))_{M>0} = \bigcap_{\substack{A \in \Lambda_1 \\ (\Lambda_1, \Lambda_2) \in \mathcal{L}}} \bar{\Gamma}(\Lambda_1, \Lambda_2, \ell).$$

Step 5. For any $x \in E$, we set

$$\Gamma(x,\ell+1) \;=\; (\Gamma(x,\ell+1,M))_{M>0} \;=\; \Gamma(x,\ell) \cap \bigcap_{\substack{x\in A\\ A\in\mathcal{T}}} \Gamma'(A,\ell+1).$$

This finishes the mathematical definition of the blobs $\Gamma(x,\ell)$, for $x \in E, \ell \geq 1$. The five steps above suggest an obvious algorithm for the computation of ellipsoidal blobs $\mathcal{A}_{x,\ell}$ that approximate $\Gamma(x,\ell)$; i.e. for any $x \in E, \ell \geq 1$, the ellipsoidal blob $\mathcal{A}_{x,\ell}$ is C_ℓ-equivalent to $\Gamma(x,\ell)$, where C_ℓ is a constant that depends solely on ℓ, m and n. Indeed, the blobs $\Gamma(x,1)$ and $\mathcal{B}(x,\delta)$ are C-equivalent to easily computed ellipsoidal blobs, as was described in Section 4, for C depending only on m and n. Assume $\ell \geq 1$, and that we have already computed ellipsoidal blobs $\mathcal{A}_{x,\ell}$ for $x \in E$, which are C_ℓ-equivalent to the corresponding $\Gamma(x,\ell)$. We will follow the five steps above, using the operations on ellipsoidal blobs discussed in Section 3 (in an analogous way to the pedagogical algorithm from Section 4), and the operations on the CK decomposition described in (6) and (8) of Section 5. Thus in five steps we compute from the ellipsoidal blobs $\{\mathcal{A}_{x,\ell}\}_{x\in E}$ the new ellipsoidal blobs $\{\mathcal{A}_{x,\ell+1}\}_{x\in E}$. The ellipsoidal blobs $\{\mathcal{A}_{x,\ell+1}\}_{x\in E}$ are $C_{\ell+1}$-equivalent to $\{\Gamma(x,\ell+1)\}_{x\in E}$, for $C_{\ell+1} = C \cdot C_\ell$ depending solely on ℓ, m and n.

Our algorithm computes the ellipsoidal blobs $\{\mathcal{A}_{x,\ell}\}_{x\in E}$ for $\ell = 1, \ldots, \ell_*$, for the same number ℓ_* as in Section 2, which is a constant that depends solely on m and n. The algorithm returns the number

$$\max\{\text{onset } \mathcal{A}_{x,\ell_*} : x \in E\} \tag{2}$$

which may be easily computed, as was explained in Section 3, and that has the same order of magnitude as $\max\{\text{onset } \Gamma(x,\ell_*) : x \in E\}$. This completes the description of our algorithm.

How many computer operations are involved in the computation of $\{\mathcal{A}_{x,\ell+1}\}_{x\in E}$ from $\{\mathcal{A}_{x,\ell}\}_{x\in E}$? Recall that calculating the Minkowski sum of two ellipsoidal blobs requires C operations, while computing the intersection of k ellipsoidal blobs takes Ck operations, for C depending only on m and n. The amount of work in step 1 and in step 5 is thus bounded by

$$C \cdot \sum_{A\in\mathcal{T}} \#(A) < C' N \log N$$

by (7) from Section 5, where C, C' depend only on m and n. We also used properties (6) and (8) from Section 5, to produce the elements of all $A \in \mathcal{T}$ in Step 1, to compute $\operatorname{diam}_\infty(A)$ and to find for all $x \in E$ the set of $A \in \mathcal{T}$ such that $x \in A$. Step 2 requires no more than $CN \log N$ operations, by (7) from Section 5, where again we also use (6) and (8) of Section 5 to produce the nodes that belong to Λ_1, Λ_2 for all $(\Lambda_1,\Lambda_2) \in \mathcal{L}$ and to find $\operatorname{diam}_\infty(\cup\Lambda_i)$.

Step 3 takes CN operations. For step 4, we need no more than

$$C \cdot \sum_{A \in \mathcal{T}} \# \left(\{ (\Lambda_1, \Lambda_2) \in \mathcal{L} : A \in \Lambda_1 \} \right) < C' N \log N$$

operations, by (6), (7) and (8) of Section 5. In total, the number of operations for each iteration is no more than $CN \log N$, where C depends solely on m and n. The algorithm performs ℓ_* iterations, each taking $CN \log N$ operations, and then computes onset $\mathcal{A}(x, \ell_*)$ for $x \in E$, a task for which CN operations suffice. Thus the algorithm terminates after no more than $C'N \log N$ operations, where C' depends solely on m and n. The amount of storage needed is bounded by $C'N$.

It remains to prove (6) of Section 2, and also to show that

$$\max\{\text{onset } \Gamma(x, \ell_*) : x \in E\}$$

has the same order of magnitude as $\max\{\text{onset } \Sigma(x, \ell_*) : x \in E\}$. Once we have established that, Theorem 1 will follow from (7) of Section 2. The relation between the Σ's and the Γ's is the subject of the next section.

7. Equivalence of algorithms

The first lemma describes simple properties of the blob $\mathcal{B}(x, \delta)$.

Lemma 1. *Let $x, y \in \mathbb{R}^n$. Then, for any $a, M > 0$ and $\delta \geq 0$,*

$$B(x, \delta, M) \subset B(y, \delta + |x - y|, CM), \tag{1}$$

$$B(x, a\delta, M) \subset B(x, \delta, \max\{a^m, a\} M). \tag{2}$$

for some constant $C > 0$ depending solely on m and n.

Proof. Let $P \in B(x, \delta, M)$ be a polynomial. Then, for any $|\alpha| \leq m - 1$,

$$|\partial^\alpha P(x)| \leq M \delta^{m - |\alpha|}.$$

By Taylor's theorem

$$\begin{aligned} |\partial^\alpha P(y)| &= \left| \sum_{|\beta| \leq m - 1 - |\alpha|} \frac{\partial^{\alpha+\beta} P(x)}{\beta!} (y - x)^\beta \right| \\ &\leq C \sum_\beta M \delta^{m - |\alpha| - |\beta|} |x - y|^{|\beta|} \leq CM(\delta + |x - y|)^{m - |\alpha|} \end{aligned}$$

and (1) is proven. Now, (2) is immediate from the definition of $\mathcal{B}(x, \delta)$. $\square$

Our next lemma shows that $\|f\|_{C^m(E,\sigma)}$ is not smaller by an order of magnitude than $\max\{\text{onset } \Gamma(x, \ell_*) : x \in E\}$.

Lemma 2. *Let $F : \mathbb{R}^n \to \mathbb{R}$ be a function with $\|F\|_{C^m(\mathbb{R}^n)} \le M$, and such that*
$|F(x) - f(x)| \le M\sigma(x)$ for $x \in E$. Then for any integer $\ell > 0$ and $x \in E$,

$$J_xF \in \Gamma(x, \ell, C_\ell M). \tag{1}$$

where C_ℓ depends solely on ℓ, m and n.

Proof. By induction: For $\ell = 1$ we trivially have that $J_xF \in \Gamma(x, 1, CM)$. For the inductive step, suppose Lemma 2 holds for a given ℓ. We will prove Lemma 2 with $\ell + 1$ instead of ℓ. Note that $\|F\|_{C^m(\mathbb{R}^n)} \le M$ implies that for any $x, y \in \mathbb{R}^n$,

$$J_xF - J_yF \in B(y, |x - y|, CM) \tag{2}$$

for some constant C that depends solely on m and n. Let $x \in E$, and let $A \in \mathcal{T}$, be such that $x \in A$. For any $y \in A$ we have that $|x - y| \le \operatorname{diam}_\infty(A)$. By (2),

$$J_xF \in J_yF + B(y, \operatorname{diam}_\infty(A), CM) \subset \Gamma(y, \ell, C_\ell M) + B(y, \operatorname{diam}_\infty(A), CM)$$

because $J_yF \in \Gamma(y, \ell, C_\ell M)$ by the induction hypothesis. According to Step 1 in our algorithm,

$$J_xF \in \Gamma(A, \ell, \max\{C, C_\ell\}M) \tag{3}$$

for all $x \in E, A \in \mathcal{T}$ such that $x \in A$. Move now to Step 2. Let $x \in E$, and let $(\Lambda_1, \Lambda_2) \in \mathcal{L}, i = 1, 2$, be such that $x \in \cup\Lambda_i$. For any $A \in \Lambda_i$, according to (2),

$$J_xF - J_{x_A}(F) \in B(x_A, |x - x_A|, CM) \subset B(x_A, \operatorname{diam}_\infty(\cup\Lambda_i), CM).$$

By (3) we have that $J_{x_A}(F) \in \Gamma(A, \ell, \max\{C, C_\ell\}M)$. Hence

$$J_xF \in \Gamma(A, \ell, \max\{C, C_\ell\}M) + B(x_A, \operatorname{diam}_\infty(\cup\Lambda_i), CM),$$

and consequently

$$J_xF \in \Gamma_i(\Lambda_i, \ell, \max\{C, C_\ell\}M) \tag{4}$$

whenever $x \in \cup\Lambda_i$. Next is Step 3. Let $(\Lambda_1, \Lambda_2) \in \mathcal{L}$ and $x \in \cup\Lambda_1, y \in \cup\Lambda_2$. Since $(\mathcal{T}, \mathcal{L})$ is a $\frac{1}{2}$-CK decomposition, we have that $\frac{1}{2}|x - y| < |x_{\Lambda_1} - x_{\Lambda_2}|$ and that $|x - x_{\Lambda_1}| < |x_{\Lambda_1} - x_{\Lambda_2}|$. Therefore

$$\begin{aligned} J_xF - J_yF \in B(x, |x - y|, CM) &\subset B(x, |x_{\Lambda_1} - x_{\Lambda_2}|, 2^m CM) \\ &\subset B(x_{\Lambda_1}, |x_{\Lambda_1} - x_{\Lambda_2}|, C'M) \end{aligned}$$

for some constant C' depending only on m and n, where we used Lemma 1. By (4) we have that $J_yF \in \Gamma_2(\Lambda_2, \ell, \max\{C, C_\ell\}M)$; hence

$$J_xF \in \Gamma_2(\Lambda_2, \ell, \max\{C, C_\ell\}M) + B(x_{\Lambda_1}, |x_{\Lambda_1} - x_{\Lambda_2}|, C'M).$$

Comparing this result and (4) with Step 3, we see that

$$J_xF \in \bar{\Gamma}(\Lambda_1, \Lambda_2, \ell, \max\{C', C_\ell\}M)$$

for any $x \in \cup\Lambda_1$. Suppose that $A \in \mathcal{T}, x \in A$. If for $(\Lambda_1, \Lambda_2) \in \mathcal{L}$ we have $A \in \Lambda_1$, then $x \in \cup\Lambda_1$. In this case, Step 4 entails that

$$J_xF \in \Gamma'(A, \ell+1, \max\{C', C_\ell\}M) = \bigcap_{\substack{A \in \Lambda_1 \\ (\Lambda_1,\Lambda_2)\in\mathcal{L}}} \bar{\Gamma}(\Lambda_1, \Lambda_2, \ell, \max\{C', C_\ell\}M).$$

This, in turn, gives

$$\begin{aligned} J_xF &\in \Gamma(x, \ell, C_\ell M) \cap \bigcap_{x \in A \in \mathcal{T}} \Gamma'(A, \ell+1, \max\{C', C_\ell\}M) \\ &\subset \Gamma(x, \ell+1, \max\{C', C_\ell\}M)\,. \end{aligned}$$

The induction step follows, with $C_{\ell+1} = \max\{C', C_\ell\}$, and the lemma is proven. Note that actually we may select C_ℓ in (1) to depend only on m and n, and not on ℓ. $\square$

Lemma 3. *Let $x \in E$ and let $\ell > 0$ be an integer, $M > 0$. Then,*

$$\Gamma(x, \ell, M) \subset \Sigma(x, \ell, CM)$$

where C is a constant that depends solely on m and n.

Proof. By induction. The case $\ell = 1$ is easy, as

$$\Sigma(x, 1, M) = \Gamma(x, 1, M).$$

For the inductive step, suppose Lemma 3 holds for a given ℓ, with some constant C that will be specified later. We will prove Lemma 3 with $\ell+1$ in place of ℓ. Let $P \in \Gamma(x, \ell+1, M)$ be a polynomial. Then also $P \in \Gamma(x, \ell, M) \subset \Sigma(x, \ell, CM)$ by Step 5 and the induction hypothesis. Let $x \neq x' \in E$. By (4) of Section 2, it is enough to show that there exists $P' \in \Sigma(x', \ell, CM)$ such that

$$P - P' \in B(x, |x - x'|, CM) \tag{5}$$

Since $(\mathcal{T}, \mathcal{L})$ is a $\frac{1}{2}$-CK decomposition, there exists $(\Lambda_1, \Lambda_2) \in \mathcal{L}$ such that $x \in \cup\Lambda_1, x' \in \cup\Lambda_2$. As $P \in \Gamma(x, \ell+1, M)$, inspection of Steps 4 and 5 in the algorithm shows that $P \in \bar{\Gamma}(\Lambda_1, \Lambda_2, \ell, M)$. By Step 3, there exists a polynomial $\tilde{P} \in \mathcal{P}$ such that

$$P - \tilde{P} \in B(x_{\Lambda_1}, |x_{\Lambda_1} - x_{\Lambda_2}|, M) \text{ and } \tilde{P} \in \Gamma_2(\Lambda_2, \ell, M). \tag{6}$$

Let $A \in \Lambda_2$ be such that $x' \in A$. Clearly $|x' - x_A| \leq \text{diam}_\infty(\cup\Lambda_2)$. Recalling Step 2, we see that $\tilde{P} \in \Gamma_2(\Lambda_2, \ell, M) \subset \Gamma(A, \ell, M) + B(x_A, \text{diam}_\infty(\cup\Lambda_2), M)$

$$\subset \Gamma(A, \ell, M) + B(x', 2\text{diam}_\infty(\cup\Lambda_2), C'M)$$

by Lemma 1, for C' depending only on m and n. Recalling Step 1, we conclude that

$$\tilde{P} \in \Gamma(x', \ell, M) + B(x', 3\mathrm{diam}_\infty(\cup\Lambda_2), C''M)$$

for a constant C'' that depends only on m and n. Let $P' \in \Gamma(x', \ell, M)$ be such that

$$\tilde{P} - P' \in B(x', 3\mathrm{diam}_\infty(\cup\Lambda_2), C''M) \tag{7}$$

for a constant C'' depending only on m and n. By the induction hypothesis, $P' \in \Sigma(x', \ell, CM)$. Recall that, as $(\mathcal{T}, \mathcal{L})$ is a $\frac{1}{2}$-CK decomposition, $|x_{\Lambda_1} - x_{\Lambda_2}| < 2|x - x'|$ and also $\mathrm{diam}_\infty(\cup\Lambda_2) < \sqrt{n}|x_{\Lambda_1} - x_{\Lambda_2}|$. Lemma 1, together with (6) and (7), implies that

$$P - P' \in B(x, |x - x'|, \tilde{C}M) \tag{8}$$

for $\tilde{C}$ depending only on m and n. Note that $\tilde{C}$ is a constant depending on m and n, which is independent of ℓ and of the constant — not specified yet — from the induction hypothesis. Hence we may select $C = \tilde{C}$. Now (8) gives (5), and the lemma is proven. $\square$

Using (6) of Section 2, we conclude from Lemma 3 that if $\Gamma(x, \ell_*, M) \neq \emptyset$ for all $x \in E$, then $\|f\|_{C^m(E,\sigma)} < CM$ for some constant C that depends solely on m and n. Together with Lemma 2, this shows that

$$\max\{\text{onset } \Gamma(x, \ell_*) : x \in E\} \tag{9}$$

has the same order of magnitude as $\|f\|_{C^m(E,\sigma)}$ (and also has the same order of magnitude as $\max\{\text{onset } \Sigma(x, \ell_*) : x \in E\}$). The order of magnitude of the quantity (9) was computed by the algorithm from Section 6, using $CN \log N$ operations and CN storage. This completes the proof of Theorem 1, up to the proof of (6) from Section 2. This proof will be presented in the next section.

8. Proof of a key estimate

In this section, we complete the proof of Theorem 1 by establishing the key estimate (6) from Section 2. We will reduce matters to Theorem 3 (from Section 1), which is one of the main results in [14]. We will also use an elementary "clustering lemma" from [15].

Lemma 1. *When $\ell \geq 1$, and $S \subset \mathbb{R}^n$, with $\#(S) = \ell + 1$, we can partition S into subsets $S_0, S_1, \ldots, S_{\nu_{\max}}$, such that*

(a) *$\#(S_\nu) \leq \ell$ for each ν, and*

(b) *distance $(S_\mu, S_\nu) > c\cdot$ diameter (S) for $\mu \neq \nu$, with c depending only on ℓ.*

By induction on $\ell \geq 1$, we will establish the following result.

LEMMA 2. *Suppose $y_0 \in E$, $\ell \geq 1$, $M > 0$, and $P \in \Sigma(y_0, \ell, M)$. Then, for any $S \subseteq E$, with $y_0 \in S$ and $\#(S) \leq \ell$, there exists a map $y \mapsto P^y$ from S into $\mathcal{P}$, such that*:

(A) $P^{y_0} = P$;

(B) $|\partial^\alpha P^y(y)| \leq CM$ *for* $|\alpha| \leq m-1$, $y \in S$;

(C) $|\partial^\alpha (P^y - P^{y'})(y)| \leq CM|y-y'|^{m-|\alpha|}$ *for* $|\alpha| \leq m-1$, $y, y' \in S$; *and*

(D) $|P^y(y) - f(y)| \leq CM\sigma(y)$ *for all* $y \in S$.

Here, C depends only on m, n, ℓ.

Proof. For $\ell = 1$, we have $S = \{y_0\}$. We take $P^{y_0} = P$. Thus, (A) holds by definition, (B) and (D) hold since $P \in \Sigma(y_0, 1, M)$, and (C) holds since $y = y' = y_0$ for $y, y' \in S$. This proves Lemma 2 for $\ell = 1$.

For the inductive step, suppose Lemma 2 holds for a given ℓ. We will prove Lemma 2 with $(\ell + 1)$ in place of ℓ.

Thus, suppose $P \in \Sigma(y_0, \ell+1, M)$, and let $S \subseteq E$, with $y_0 \in S$ and $\#(S) \leq \ell + 1$. We must produce a map $y \mapsto P^y$ satisfying (A),..., (D). If $\#(S) \leq \ell$, then the desired map exists, thanks to our induction hypothesis. Hence, we may suppose that $\#(S) = \ell + 1$. Let δ= diameter $(S) > 0$. We write c, C, C', etc., to denote constants depending only on m, n, and ℓ.

By Lemma 1, we may partition S into nonempty subsets S_ν $(0 \leq \nu \leq \nu_{\max})$, with the following properties.

(1) $\#(S_\nu) \leq \ell$ for each ν.

(2) $y_0 \in S_0$.

(3) distance $(S_\nu, S_{\nu'}) \geq c\delta$ if $\nu \neq \nu'$.

For each ν $(1 \leq \nu \leq \nu_{\max})$, pick $y_\nu \in S_\nu$. Since $P \in \Sigma(y_0, \ell+1, M)$, we know that, for each ν (including $\nu = 0$), there exists $P_\nu \in \Sigma(y_\nu, \ell, M)$ with

$$|\partial^\alpha (P_\nu - P)(y_0)| \leq M|y_\nu - y_0|^{m-|\alpha|} \text{ for } |\alpha| \leq m-1. \tag{4}$$

In particular, for $\nu = 0$ we necessarily have

$$P_0 = P. \tag{5}$$

For each ν, we may apply our induction hypothesis to the point y_ν, the set S_ν, and the polynomial P_ν, thanks to (1). Hence, there is a map $y \mapsto P^y$ from S_ν into $\mathcal{P}$, satisfying:

(6) $P^{y_\nu} = P_\nu$;

(7) $|\partial^\alpha P^y(y)| \le CM$ for $|\alpha| \le m-1$, $y \in S_\nu$;

(8) $|\partial^\alpha (P^y - P^{y'})(y)| \le CM|y-y'|^{m-|\alpha|}$ for $|\alpha| \le m-1$, $y, y' \in S_\nu$; and

(9) $|P^y(y) - f(y)| \le CM\sigma(y)$ for all $y \in S_\nu$.

Combining the above maps on the S_ν into a single map $y \mapsto P^y$ from S into $\mathcal{P}$, we obtain the following results from (5),…,(9).

- $P^{y_0} = P_0 = P$;
- $|\partial^\alpha P^y(y)| \le CM$ for $|\alpha| \le m-1$, $y \in S$;
- $|P^y(y) - f(y)| \le CM\sigma(y)$ for all $y \in S$.

Thus, our map $y \mapsto P^y$, from S into $\mathcal{P}$, satisfies properties (A), (B), (D) from the statement of Lemma 2. Also, (8) shows that property (C) holds, provided y and y' belong to the same S_ν.

Hence, to complete the proof of Lemma 2, it is enough to prove (C) in the case $y \in S_\nu$, $y' \in S_{\nu'}$, $\nu \ne \nu'$. In view of (3) (with $\delta =$ diameter S), this means that

(10)
$$|\partial^\alpha (P^y - P^{y'})(y)| \le CM\delta^{m-|\alpha|} \text{ for } |\alpha| \le m-1,\ y \in S_\nu,\ y' \in S_{\nu'},\ \nu \ne \nu'.$$

Thus, the proof of Lemma 2 is reduced to (10). Let $y \in S_\nu$, $y' \in S_{\nu'}$, with $\nu \ne \nu'$. From (6) and (8), we have

(11) $|\partial^\alpha (P^y - P_\nu)(y)| \le CM|y - y_\nu|^{m-|\alpha|} \le CM\,\delta^{m-|\alpha|}$ for $|\alpha| \le m-1$,

and

(12) $|\partial^\alpha (P^{y'} - P_{\nu'})(y')| \le CM|y' - y_{\nu'}|^{m-|\alpha|} \le CM\,\delta^{m-|\alpha|}$ for $|\alpha| \le m-1$.

Also, (4) shows that

(13) $|\partial^\alpha (P_\nu - P_{\nu'})(y_0)| \le M|y_\nu - y_0|^{m-|\alpha|} + M|y_{\nu'} - y_0|^{m-|\alpha|} \le CM\,\delta^{m-|\alpha|}$

for $|\alpha| \le m-1$. By Lemma 1 from Section 7, the estimates (12) and (13) imply

(14) $$|\partial^\alpha (P^{y'} - P_{\nu'})(y)| \le CM\,\delta^{m-|\alpha|} \text{ for } |\alpha| \le m-1,$$

and

(15) $$|\partial^\alpha (P_\nu - P_{\nu'})(y)| \le CM\,\delta^{m-|\alpha|} \text{ for } |\alpha| \le m-1.$$

From (11), (14), (15), we obtain the desired estimate (10), and the proof of Lemma 2 is complete. □

We now prove estimate (6) from Section 2. We take $\ell_* = k$ as in Theorem 3. Thus, ℓ_* depends only on m and n. Let $M > 0$ satisfy the hypothesis of (6), namely

(16) $$\Sigma(x, \ell_*, M) \ne \phi \text{ for each } x \in E.$$

We will show that the hypothesis of Theorem 3 holds, for the set E, the functions f and σ, and the constant CM, for a large enough C depending only on m and n. To see this, let $S \subset E$, with $\#(S) \leq k$. If S is empty, there is nothing to prove. If S is nonempty, then we pick $y_0 \in S$ and then pick $P \in \Sigma(y_0, \ell_*, M)$. (We can find such a P, thanks to (16).) Applying Lemma 2, with $\ell = \ell_* = k$, we obtain a map $y \mapsto P^y$ from S into $\mathcal{P}$, satisfying $P^{y_0} = P$ and

(17) $|\partial^\alpha P^y(y)| \leq CM$ for $|\alpha| \leq m-1$, $y \in S$;

(18) $|\partial^\alpha (P^y - P^{y'})(y)| \leq CM|y-y'|^{m-|\alpha|}$ for $|\alpha| \leq m-1$, $y, y' \in S$; and

(19) $|P^y(y) - f(y)| \leq CM\sigma(y)$ for all $y \in S$.

In (17), (18), (19), the constant C depends only on m, n and ℓ_*. Since ℓ_* depends only on m and n, it follows that C depends only on m and n. The existence of a map $y \mapsto P^y$ satisfying (17), (18), (19) is precisely the hypothesis of Theorem 3.

Applying Theorem 3, we conclude that $\|f\|_{C^m(E,\sigma)} \leq CM$, with C depending only on m and n. This is precisely the conclusion of (6) from Section 2. This is complete, and with it, the proof of Theorem 1. $\square$

9. On the proof of Theorem 2

In this section, we explain a key idea in the proof of Theorem 2. Recall that we have defined convex sets $\Gamma(x, \ell, M)$, starting from a finite set $E \subset \mathbb{R}^n$ and functions $f : E \to \mathbb{R}$ and $\sigma : E \to [0, \infty)$. If we keep E and σ unchanged, but replace f by zero, then in place of $\Gamma(x, \ell, M)$, we obtain by the same construction a new family of convex sets $\Gamma_0(x, \ell, M)$. An easy induction on ℓ shows that this new family has the form $\Gamma_0(x, \ell, M) = M\sigma(x, \ell)$ for a convex, symmetric set $\sigma(x, \ell) \subset \mathcal{P}$. The convex sets $\sigma(x, \ell)$ play a basic rôle along with the $\Gamma(x, \ell, M)$.

We prepare to state the main properties of the Γ's and the σ's, to be used in the proof of Theorem 2. We write c, C to denote constants depending only on m, n; and we write C_ℓ to denote constants depending only on m, n and ℓ. For $x \in \mathbb{R}^n$ and $P, Q \in \mathcal{P}$, we write $P \odot_x Q$ to denote the unique $S \in \mathcal{P}$ such that, for smooth F and G, $J_x(F) = P$ and $J_x(G) = Q$ imply $J_x(FG) = S$.

For convex sets $A, B \subset \mathcal{P}$, we write $A - B$ for the Minkowski difference $\{P - P' : P \in A, P' \in B\}$.

The basic properties of the Γ's and σ's are as follows.

(0) Suppose $F \in C^m(\mathbb{R}^n)$ and $M > 0$ with

$$\|F\|_{C^m(\mathbb{R}^n)} \leq M \text{ and } |F(x) - f(x)| \leq M\sigma(x) \text{ for all } x \in E.$$

Then

$$J_x(F) \in \Gamma(x, \ell, C_\ell M)$$

for all $x \in E, \ell \geq 1$.

(1) For $x \in E$, the blob $\Gamma(x,1)$ is C-equivalent to the blob $\mathcal{K} = (K_M)_{M>0}$ defined as

$$K_M = \{P \in \mathcal{P} : |\partial^\beta P(x)| \leq M \text{ for } |\beta| \leq m-1, |P(x) - f(x)| \leq M\sigma(x)\}.$$

(2) For any $x, y \in E, M > 0$,

$$\Gamma(x, \ell+1, M) \subset \Gamma(y, \ell, C_\ell M) + B(y, |x-y|, C_\ell M)\,.$$

Similarly,

$$\sigma(x, \ell+1) \subset C_\ell \sigma(y, \ell) + B(y, |x-y|, C_\ell).$$

(3) $\Gamma(x, \ell, M) + M\sigma(x, \ell) \subset \Gamma(x, \ell, C_\ell M)$, and $\Gamma(x, \ell, M) - \Gamma(x, \ell, M) \subset C_\ell M \sigma(x, \ell)$, for any $x \in E, \ell \geq 1$.

(4) Suppose $P \in \sigma(x, \ell), Q \in \mathcal{P}$ and $0 < \delta \leq 1$. Assume that $|\partial^\beta P(x)| \leq \delta^{m-|\beta|}$ and $|\partial^\beta Q(x)| \leq \delta^{-|\beta|}$ for $|\beta| \leq m-1$. Then $P \odot_x Q \in C_\ell \sigma(x, \ell)$.

In fact, we have already proven (0) and (2) in Section 7 (see (8) in the proof of Lemma 3 in Section 7); and (1), (3) are easy consequences of the definitions. We will prove (4) in [20]; it is an instance of "Whitney ω-convexity" (see [17]). Our proof of Theorem 2 will be based on the above properties of the Γ's and σ's. The convex sets $\Sigma(x, \ell, M)$ and appropriate $\sigma(x, \ell)$ also satisfy (0),…,(4). On the other hand, the proof of Theorem 3 is based on the study of convex sets $\mathcal{K}_f(x, k, M)$ defined as follows.

(5) A given $P \in \mathcal{P}$ belongs to $\mathcal{K}_f(x, k, M)$ if, for any $S \subset E$ of cardinality at most k, there exists $F^S \in C^m(\mathbb{R}^n)$ with
   - (a) $\|F^S\|_{C^m(\mathbb{R}^n)} \leq M$;
   - (b) $|F^S(x) - f(x)| \leq M\sigma(x)$ for all $x \in S$; and
   - (c) $J_x(F^S) = P$.

(See [14], [17].) Suppose we write $\Gamma_{\text{old}}(x, \ell, M) := \mathcal{K}_f(x, (D+1)^\ell, M)$, with $D = \dim \mathcal{P}$. Then the $\Gamma_{\text{old}}(x, \ell, M)$, together with appropriate $\sigma_{\text{old}}(x, \ell)$, also satisfy (0),…,(4). In particular, (2) follows by applying Helly's theorem [34] on convex sets (see Lemma 10.2 in [14]). Thus, properties (0),…,(4) hold for more than one family of Γ's and σ's.

The proof of Theorem 3, as given in [14], makes little use of the exact definition (5). Instead, it relies almost entirely on properties (0),…,(4) for the families $\Gamma_{\text{old}}, \sigma_{\text{old}}$. Hence, it is natural to guess that there is a version of Theorem 3 valid for any Γ's and σ's that satisfy (0),…,(4). More precisely, we assert the following:

(6) Let $\Gamma(x, \ell, M) \subset \mathcal{P}$ be convex sets, defined for $x \in E, \ell \geq 1, M > 0$ (and increasing in M); and let $\sigma(x, \ell) \subset \mathcal{P}$ be symmetric convex sets, defined for $x \in E, \ell \geq 1$. Assume that the Γ's and σ's satisfy (0),…,(4). Let $M_0 > 0$ and suppose that $\Gamma(x, \ell_*, M_0)$ is nonempty for each $x \in E$, where ℓ_* is a large constant determined by m and n.

Then, there exists $F \in C^m(\mathbb{R}^n)$, such that $\|F\|_{C^m(\mathbb{R}^n)} \leq CM_0$, and $|F(x) - f(x)| \leq CM_0\sigma(x)$ for all $x \in E$.

In fact, a sharper version of (6) follows at once from our work in Sections 7 and 8. We have only to observe that our proof that $\|f\|_{C^m(E,\sigma)}$ is comparable to $\max_{x \in E}$ onset $\Sigma(x, \ell)$ used only properties (0), (1), (2) of the $\Sigma(x, \ell, M)$. Hence, (6) holds without the need to assume (3) or (4). The main step here is to quote Theorem 3.

However, if we assume all the key properties (0),…,(4), then we can do better. By adapting the proof of Theorem 3 rather than quoting the result, one can construct an F as in (6) rather than merely proving its existence. This leads to an algorithm to solve Problem 2. To make the algorithm run efficiently, we have to decide, with $\log N$ work, which cube from a relevant Calderón-Zygmund decomposition contains a given point $x \in \mathbb{R}^n$. We achieve this by bringing in the work of Arya, Mount, Netanyahu, Silverman and Wu [1], associating a balanced tree to a given finite set $E \subset \mathbb{R}^n$. We also make further use of the Callahan-Kosaraju decomposition. Thus, we are led to the proof of Theorem 2. Full details will appear in [20].

Princeton University, Princeton, New Jersey
E-mail address: cf@math.princeton.edu

Institute for Advanced Study, Princeton, New Jersey
Current address: School of Mathematical Sciences, Tel-Aviv University, Tel-Aviv, Israel
E-mail address: klartagb@tau.ac.il

References

[1] S. Arya, D. Mount, N. Netanyahu, R. Silverman, and A. Wu, An optimal algorithm for approximate nearest neighbor searching in fixed dimensions, *J. Assoc. for Computing Machinery* **45** (1998), 891–923.

[2] E. Bierstone, P. Milman, and W. Pawłucki, Differentiable functions defined in closed sets. A problem of Whitney, *Invent. Math.* **151** (2003), 329–352.

[3] ———, Higher-order tangents and Fefferman's paper on Whitney's extension problem, *Ann. of Math.* **164** (2006), 361–370.

[4] A. Brudnyi and Y. Brudnyi, Metric spaces with linear extensions preserving Lipschitz condition, *Amer. J. Math.*, to appear.

[5] Y. Brudnyi, A certain extension theorem, *Funk. Anal. Prilozhen* **4** (1970), 97–98; English transl. in *Func. Anal. Appl.* **4** (1970), 252–253.

[6] Y. Brudnyi and P. Shvartsman, The traces of differentiable functions to subsets of $\mathbb{R}^n$, in *Linear and Complex Analysis*, *Lecture Notes in Math.* **1574**, 279–281, Springer-Verlag, New York (1994).

[7] ———, A linear extension operator for a space of smooth functions defined on closed subsets of $\mathbb{R}^n$, *Dokl. Akad. Nauk SSSR* **280** (1985), 268–272; English transl. in *Soviet Math. Dokl.* **31** (1985), 48–51.

[8] ———, Generalizations of Whitney's extension theorem, *IMRN* **1994**, no. 3, 129–139.

[9] ———, The Whitney problem of existence of a linear extension operator, *J. Geometric Anal.* **7** (1997), 515–574.

[10] ———, Whitney's extension problem for multivariate $C^{1,w}$-functions, *Trans. Amer. Math. Soc.* **353** (2001), 2487–2512.

[11] P. B. Callahan and S. R. Kosaraju, A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields, *J. Assoc. for Computing Machinery* **42** (1995), 67–90.

[12] M. Dyer, A class of convex programs with applications to computational geometry, *Proc. Eighth Annual Symposium on Computational Geometry* (1992), 9–15.

[13] C. Fefferman, Interpolation and extrapolation of smooth functions by linear operators, *Revista Matemática Iberoamericana* **21** (2005), 313–348.

[14] ———A sharp form of Whitney's extension theorem, *Ann. of Math.* **161** (2005), 509–577.

[15] ———Whitney's extension problem for C^m, *Ann. of Math.* **164** (2006), 313–359.

[16] ———Whitney's extension problem in certain function spaces, preprint.

[17] ———A generalized sharp Whitney theorem for jets, *Revista Matemática Iberoamericana* **21** (2005), 577–688.

[18] ———Extension of $C^{m,\omega}$ smooth functions by linear operators, *Revista Matemática Iberoamericana*, to appear.

[19] ———C^m extension by linear operators, *Ann. of Math.* **166** (2007), 779–835.

[20] ———, Fitting a C^m-smooth function to data III, *Ann. of Math.*, to appear, 2009.

[21] C. Fefferman and B. Klartag, Fitting a C^m-smooth function to data II, *Revista Matematica Iberoamericana*, to appear.

[22] G. Glaeser, Étude de quelques algèbres tayloriennes, *J. Analyse Math.* **6** (1958), 1–124.

[23] S. Har-Peled and M. Mendel, Fast construction of nets in low-dimensional metrics, and their applications, *SIAM J. Comput.* **35** (2006), 1148–1184.

[24] F. John, Extremum problems with inequalities as subsidiary conditions, in *Studies and Essays Presented to R. Courant on his 60th Birthday*, Interscience Publ. Inc., New York (1948), 187–204.

[25] D. Knuth, *The Art of Computer Programming*, *Volume* 1: *Fundamental Algorithms*, 3^{rd} edition, Addison-Wesley, Reading, MA, 1997.

[26] B. Malgrange, *Ideals of Differentiable Functions*, Oxford Univ. Press, London, 1966.

[27] N. Megiddo, Linear programming in linear time when the dimension is fixed, *J. Assoc. for Computing Machinery* **31** (1984), 114–127.

[28] F. P. Preparata and M. I. Shamos, *Computational Geometry*: *An Introduction*, 2nd edition, *Texts and Monographs in Computer Science*, Springer-Verlag, New York, 1985.

[29] P. Shvartsman, Lipschitz selections of set-valued mappings and traces of functions from the Zygmund class on an arbitrary compactum, *Dokl. Acad. Nauk SSSR* **276** (1984), 559–562; English transl. in *Soviet Math. Dokl.* **29** (1984), 565–568.

[30] On traces of functions of Zygmund classes, *Sibirskyi Mathem. J.* **28** (1987), 203–215; English transl. in *Siberian Math. J.* **28** (1987), 853–863.

[31] ———, Lipschitz selections of set-valued mappings and Helly's theorem, *J. Geometric Anal.* **12** (2002), 289–324.

[32] E. M. Stein, *Singular Integrals and Differentiability Properties of Functions*, *Princeton Math. Series* **30**, Princeton Univ. Press, Princeton, NJ, 1970.

[33] J. von Neumann, First draft of a report on the EDVAC, Contract No. W-670-ORD-492, Moore School of Electrical Engineering, Univ. of Penn., Philadelphia, 1945. Reprinted in *IEEE Annals of the History of Computing* **15** (1993), 27–75.

[34] R. Webster, *Convexity*, Oxford Science Publications, Oxford Univ. Press, New York, 1994.

[35] H. Whitney, Analytic extensions of differentiable functions defined in closed sets, *Trans. Amer. Math. Soc.* **36** (1934), 63–89.

[36] ———, Differentiable functions defined in closed sets. I, *Trans. Amer. Math. Soc.* **36** (1934), 369–387.

[37] ———, Functions differentiable on the boundaries of regions, *Ann. of Math.* **35** (1934), 482–485.

[38] N. Zobin, Whitney's problem on extendability of functions and an intrinsic metric, *Adv. Math.* **133** (1998), 96–132.

[39] ———, Extension of smooth functions from finitely connected planar domains, *J. Geom. Anal.* **9** (1999), 491–511.

(Received October 6, 2005)
(Revised October 16, 2006)