Rev. Mat. Iberoam. **32** (2016), no. 1, 275–376
DOI 10.4171/RMI/887

Fitting a Sobolev function to data I

Charles Fefferman, Arie Israel, and Garving Luli

Abstract. In this paper and two companion papers, we produce efficient algorithms to solve the following interpolation problem: Let $m \geq 1$ and $p > n \geq 1$. Given a finite set $E \subset \mathbb{R}^n$ and a function $f : E \to \mathbb{R}$, compute an extension F of f belonging to the Sobolev space $W^{m,p}(\mathbb{R}^n)$ with norm having the smallest possible order of magnitude; secondly, compute the order of magnitude of the norm of F. The combined running time of our algorithms is at most $CN \log N$, where N denotes the cardinality of E, and C depends only on m, n, and p.

1. Introduction

In this paper and [20], [21], we interpolate data by a function $F : \mathbb{R}^n \to \mathbb{R}$ whose Sobolev norm has the least possible order of magnitude. Our computations consist of efficient algorithms, to be implemented on an (idealized) von Neumann computer.

Our results are the analogues for Sobolev spaces of some of the main results of Fefferman–Klartag [15], [16], [17] on interpolation of data by functions in $C^m(\mathbb{R}^n)$.

Let us set up notation and definitions. Fix $m, n \geq 1$ and $1 < p < \infty$. We work in the Sobolev space

$$\mathbb{X} = L^{m,p}(\mathbb{R}^n) \quad \text{with seminorm} \quad \|F\|_{\mathbb{X}} = \Big(\int_{\mathbb{R}^n} \sum_{|\alpha|=m} |\partial^\alpha F(x)|^p \, dx \Big)^{1/p} \tag{1.1}$$

or

$$\mathbb{X} = W^{m,p}(\mathbb{R}^n) \quad \text{with norm} \quad \|F\|_{\mathbb{X}} = \Big(\int_{\mathbb{R}^n} \sum_{|\alpha|\leq m} |\partial^\alpha F(x)|^p \, dx \Big)^{1/p}. \tag{1.2}$$

We make the assumption

$$p > n, \tag{1.3}$$

Mathematics Subject Classification (2010): Primary 65D17; Secondary 65D05.
Keywords: Algorithm, interpolation, Sobolev spaces.

so that the Sobolev theorem tells us that

$$\mathbb{X} \subset C^{m-1}_{loc}(\mathbb{R}^n). \tag{1.4}$$

We write c, C, C', etc. to denote "universal constants", i.e., constants determined by m, n, p in (1.1), (1.2). These symbols may denote different universal constants in different occurrences.

Now let

$$E = \{z_1, \dots, z_N\} \subset \mathbb{R}^n. \tag{1.5}$$

Then $\mathbb{X}(E)$ denotes the vector space of all real-valued functions on E, equipped with the norm (or seminorm)

$$\|f\|_{\mathbb{X}(E)} = \inf\left\{\|F\|_{\mathbb{X}} : F \in \mathbb{X}, F = f \text{ on } E\right\}.$$

Let $A \geq 1$ be a constant. An "A-optimal interpolant" for a function $f \in \mathbb{X}(E)$ is a function $F \in \mathbb{X}$ that satisfies $F = f$ on E and $\|F\|_{\mathbb{X}} \leq A \cdot \|f\|_{\mathbb{X}(E)}$.

Our goal here is to solve the following

Problems:

(A) Compute a C-optimal interpolant for a given function $f \in \mathbb{X}(E)$.

(B) Given $f \in \mathbb{X}(E)$, compute a number $|||f|||$ such that

$$c\,|||f||| \leq \|f\|_{\mathbb{X}(E)} \leq C\,|||f|||.$$

We owe the reader an explanation of what it means to "compute a function" in Problem (A). First of all, our computations are performed on a computer with standard von Neumann architecture. We assume that each memory cell can store a single integer or real number. We study two distinct models of computation. In the first model ("infinite-precision") we assume that our computer deals with exact real numbers, without roundoff errors. Our second, more realistic model ("finite-precision") assumes that our machine handles only S-bit machine numbers for some fixed, large S. To work with the finite-precision model, we make a rigorous study of the roundoff errors arising in our algorithms. For simplicity, in this introduction, we restrict attention to the infinite-precision model.

To "compute" a function $F \in C^{m-1}_{loc}(\mathbb{R}^n)$, the computer first performs "one-time work", then answers "queries." A query consists of a point $\underline{x} \in \mathbb{R}^n$, and the computer responds to a query $\underline{x}$ by computing $\partial^\alpha F(\underline{x})$ for all $|\alpha| \leq m-1$.

We want algorithms that make minimal use of the resources of our computer. For the computation of a function F as in Problem (A), the relevant resources are

- the number of computer operations used for the one-time work,
- the number of memory cells used for all the work,
- and the number of computer operations used in responding to a query.

We refer to these as the "one-time work", the "space" (or "storage"), and the "query work", respectively.

For the computation of the single number $|||f|||$ in Problem (B), the relevant computer resources are the number of operations used, and the number of memory cells required. We refer to these as, respectively, the "work" and "storage".

We are concerned with algorithms that work for arbitrary f and E. If we allowed ourselves favorable geometric assumptions on E, our problems would be much easier.

We can now state our results in their simplest form. Recall that N denotes the number of points in our finite set E.

Theorem 1. *One can compute a* C*-optimal interpolant for a given function* $f \in \mathbb{X}(E)$*, with one-time work* $\leq CN \log N$*, storage* $\leq CN$*, and query work* $\leq C \log N$*.*

Theorem 2. *Given* $f \in \mathbb{X}(E)$*, one can compute a number* $|||f|||$ *such that*

$$c\,|||f||| \leq \|f\|_{\mathbb{X}(E)} \leq C\,|||f|||;$$

the computation uses work $\leq CN \log N$ *and storage* $\leq CN$*.*

Obviously, in Theorem 1, the one-time work must be at least N, since we have to read the data; and the query work is at least 1, since we must at least read the query. Similarly, in Theorem 2, the work must be at least N. Hence, for trivial reasons, the work of our algorithms can be improved at most by a factor $\log N$.

Very likely, the work and storage asserted above are best possible.

To prepare to state our results in their full strength, we recall the following results from our previous paper [18].

Theorem 3 (Extension operators). *There exists a linear map* $T : \mathbb{X}(E) \to \mathbb{X}$ *such that* Tf *is a* C*-optimal interpolant of* f *for any* $f \in \mathbb{X}(E)$*.*

Theorem 4 (Formula for the norm). *There exist linear functionals* $\xi_l : \mathbb{X}(E) \to \mathbb{R}$ $(l = 1, \dots, L)$ *such that*

- $L \leq CN$,
- *and the quantity*

$$|||f||| = \left(\sum_{l=1}^{L} |\xi_l(f)|^p \right)^{1/p}$$

satisfies

$$c\,|||f||| \leq \|f\|_{\mathbb{X}(E)} \leq C\,|||f||| \quad \textit{for all } f \in \mathbb{X}(E).$$

To prove Theorems 1 and 2, we will compute the linear map T and the functionals ξ_l in Theorems 3 and 4. To do so, we exploit a sparse structure for T and ξ_l, established in [18].

We recall the relevant definitions.

Let $\Omega = \{\omega_1, \dots, \omega_{\nu_{\max}}\}$ be a finite list of linear functionals on $\mathbb{X}(E)$. Then we say that Ω is a "set of assists" if each ω_ν can be written as

$$\omega_\nu(f) = \sum_{i=1}^{I_\nu} \mu_{\nu i}\, f(z_{\nu i}) \qquad (f \in \mathbb{X}(E)); \tag{1.6}$$

where $I_\nu \geq 1$, $\mu_{\nu i} \in \mathbb{R}$, $z_{\nu i} \in E$ are independent of f, and

$$\sum_{\nu=1}^{\nu_{\max}} I_\nu \leq CN. \tag{1.7}$$

The point is that if (1.7) holds, then for a given $f \in \mathbb{X}(E)$ we can compute all the assists $\omega_1(f), \ldots, \omega_{\nu_{\max}}(f)$, using at most CN computer operations. It will be useful to precompute the $\omega_\nu(f)$, because each of these quantities may be used many times in subsequent calculations.

Let $\Omega = \{\omega_1, \ldots, \omega_{\nu_{\max}}\}$ be a set of assists.

A linear functional

$$\xi : \mathbb{X}(E) \to \mathbb{R}$$

has "Ω-assisted bounded depth" if it can be written in the form

$$\xi(f) = \sum_{i=1}^{I} \lambda_i f(z_i) + \sum_{j=1}^{J} \beta_j \, \omega_{\nu_j}(f) \text{ for all } f \in \mathbb{X}(E), \tag{1.8}$$

where $I, J, \lambda_i, \beta_j, \nu_j$ and $z_i \in E$ are independent of f, and

$$I + J \leq C. \tag{1.9}$$

If (1.8), (1.9) hold, and if we have precomputed $\omega_1(f), \ldots, \omega_{\nu_{\max}}(f)$, then we can calculate $\xi(f)$ using at most C computer operations.

We call (1.6) and (1.8) "short forms" of the assists ω_ν and the functional ξ, respectively. Note that a functional may be written in short form in more than one way.

A linear map $T : \mathbb{X}(E) \to C^{m-1}_{loc}(\mathbb{R}^n)$ will be said to have "Ω-assisted bounded depth" if for each $x \in \mathbb{R}^n$ and each multiindex α of order $|\alpha| \leq m-1$, the linear functional

$$\mathbb{X}(E) \ni f \longmapsto \partial^\alpha T f(x) \tag{1.10}$$

has Ω-assisted bounded depth.

In [18], we proved the following sharper version of Theorems 3 and 4.

Theorem 5. *There exists a set of assists $\Omega = \{\omega_1, \ldots, \omega_{\nu_{\max}}\}$ such that the linear map T in Theorem 3, and the linear functionals $\xi_1, \ldots, \xi_L$ in Theorem 4, may be taken to have Ω-assisted bounded depth.*

If we knew the assists $\omega_1, \ldots, \omega_{\nu_{\max}}$ and the functionals $\xi_1, \ldots, \xi_L$ in their short form, then we could easily compute $|||f|||$ in Theorem 4 by first computing the $\omega_\nu(f)$, then computing the $\xi_l(f)$. The whole computation would require only CN computer operations.

Similarly, suppose we knew the assists ω_ν and the linear functionals (1.10) in their short form.

Given $f \in \mathbb{X}(E)$, we could precompute $\omega_1(f), \ldots, \omega_{\nu_{\max}}(f)$ with at most CN operations, after which we could answer queries: Given a query point $x \in \mathbb{R}^n$, we could compute $\partial^\alpha T f(x)$ (all $|\alpha| \leq m-1$) in at most C operations. Thus, we could give highly efficient solutions to Problems (A) and (B) above.

Unfortunately, the proof of Theorem 5 in [18] is not constructive. It does not supply any formulas for the assists ω_ν, the functionals ξ_l, or the operator T. Our purpose here is to remedy this defect by proving the following result.

Theorem 6 (Main theorem). *For suitable $\Omega = \{\omega_1, \dots, \omega_{\nu_{\max}}\}$, T, $\xi_1, \dots, \xi_L$ as in Theorems 3, 4, 5, the assists ω_ν, and the functionals ξ_l can all be computed in short form, using work $\le CN \log N$ and storage $\le CN$. Moreover, after one-time work $\le CN \log N$ in space $\le CN$, we can answer queries as follows:*

> *A query consists of a point $x \in \mathbb{R}^n$. The response to a query x is a short-form description of the functional (1.10) for each $|\alpha| \le m-1$. The work to answer a query is $\le C \log N$.*

To prove Theorem 6, we modify the proofs of Theorems 3, 4, 5 in [18]. Let us first review some of the ideas in [18], and then explain some of the modifications needed for Theorem 6. Our discussion will be highly simplified, so that the basic ideas are not obscured by technical details.

We introduce a bit more notation. If $F \in C^{m-1}_{loc}(\mathbb{R}^n)$ and $x \in \mathbb{R}^n$, then we write $J_x(F)$ (the "jet" of F at x) to denote the $(m-1)^{\text{st}}$ degree Taylor polynomial of F at x. Thus, $J_x(F)$ belongs to $\mathcal{P}$, the vector space of all (real) polynomials of degree at most $(m-1)$ on $\mathbb{R}^n$.

We write Q, Q', etc. to denote cubes in $\mathbb{R}^n$ with sides parallel to the coordinate axes. We write δ_Q to denote the sidelength of a cube Q.

Our review of [18] starts with a local version of our present Problem (A). Let $Q \subset \mathbb{R}^n$ be a cube, let $x_0 \in Q$ be a point, and let $P_0 \in \mathcal{P}$ be a jet. We pose the following Local Interpolation Problem:

> $\underline{\mathrm{LIP}(Q, E, f, x_0, P_0)}$: Find a function $F \in L^{m,p}(\mathbb{R}^n)$, depending linearly on (f, P_0), such that
>
> $$F = f \text{ on } E \cap Q,$$
> $$J_{x_0}(F) = P_0, \text{ and}$$
> $$\int_Q \sum_{|\alpha|=m} |\partial^\alpha F(x)|^p \, dx \text{ is as small as possible up to a factor } C.$$

If we can solve $\mathrm{LIP}(Q, E, f, x_0, P_0)$ whenever Q is the unit cube Q°, then we can easily find a linear extension operator T as in Theorem 3. Moreover, careful inspection of our solution to $\mathrm{LIP}(Q^\circ, E, f, x_0, P_0)$ in [18] yields also Theorems 4 and 5. Thus, the heart of the matter is to solve $\mathrm{LIP}(Q^\circ, E, f, x_0, P_0)$.

To do so, we first associate to any point $x \in \mathbb{R}^n$ the crucial convex set

$$\sigma(x, E) = \{J_x(F) : F \in \mathbb{X}, \|F\|_{\mathbb{X}} \le 1, F = 0 \text{ on } E\}.$$

This set measures the ambiguity in $J_x(F)$ when we seek functions $F \in \mathbb{X}$ satisfying $F = f$ on E, with control on $\|F\|_{\mathbb{X}}$.

Using the geometry of the convex sets $\sigma(x, E)$, we will attach "labels" $\mathcal{A}$ to cubes $Q \subset \mathbb{R}^n$. A label is simply a set of multiindices of order $\le m-1$. Very roughly speaking, we say that Q is "tagged" with $\mathcal{A}$ if either

- Q consists of at most one point in E, or
- the scaled monomial $y \mapsto \delta_Q^{\text{power}} \cdot (y-x)^\alpha$ belongs to $\sigma(x, E)$ for all $\alpha \in \mathcal{A}$ and $x \in E \cap Q$.

If Q is tagged with $\mathcal{A}$, then we are relatively free to modify $\partial^\alpha F(x)$ for $\alpha \in \mathcal{A}$ and $x \in E$ when we seek a solution F to our local interpolation problem $\mathsf{LIP}(Q, E, f, x_0, P_0)$.

The notion of tagging gives rise to a Calderón–Zygmund decomposition $\mathsf{CZ}(\mathcal{A})$ of the unit cube Q° for each label $\mathcal{A}$. The cubes of $\mathsf{CZ}(\mathcal{A})$ are the maximal dyadic subcubes of Q° that are tagged with $\mathcal{A}$.

There is a natural order relation $<$ on labels. If labels $\mathcal{A}, \mathcal{B}$ satisfy $\mathcal{A} < \mathcal{B}$, then the decomposition $\mathsf{CZ}(\mathcal{A})$ of Q° refines the decomposition $\mathsf{CZ}(\mathcal{B})$. The maximal label under $<$ is the empty set $\emptyset$, and the Calderón–Zygmund decomposition $\mathsf{CZ}(\emptyset)$ consists of a single cube Q°. The minimal label under $<$ is the set $\mathcal{M}$ of all multiindices of order $\leq m-1$. The decomposition $\mathsf{CZ}(\mathcal{M})$ is so fine that each $Q \in \mathsf{CZ}(\mathcal{M})$ contains at most one point of our finite set E.

We now use the decomposition $\mathsf{CZ}(\mathcal{A})$ to solve Local Interpolation Problems. By induction on the label $\mathcal{A}$ (with respect to the order $<$), we solve the problem $\mathsf{LIP}(Q, E, f, x_0, P_0)$ whenever $Q \in \mathsf{CZ}(\mathcal{A})$.

In the base case, $\mathcal{A} = \mathcal{M}$, the minimal label.

Since any $Q \in \mathsf{CZ}(\mathcal{M})$ contains at most one point of E, our local interpolation problem $\mathsf{LIP}(Q, E, f, x_0, P_0)$ is trivial.

For the induction step, fix a label $\mathcal{A} \neq \mathcal{M}$. Let $\mathcal{A}^-$ be the label immediately preceding $\mathcal{A}$ in the order $<$. We make the inductive assumption that we can solve $\mathsf{LIP}(Q', E, f, x', P')$ whenever $Q' \in \mathsf{CZ}(\mathcal{A}^-)$. Using this assumption, we solve $\mathsf{LIP}(Q, E, f, x_0, P_0)$ when $Q \in \mathsf{CZ}(\mathcal{A})$. To do so, we recall that $\mathsf{CZ}(\mathcal{A}^-)$ refines $\mathsf{CZ}(\mathcal{A})$, hence our cube Q is partitioned into finitely many cubes $Q_\nu \in \mathsf{CZ}(\mathcal{A}^-)$. For each Q_ν, we carefully pick a point $x_\nu \in Q_\nu$ and a jet $P_\nu \in \mathcal{P}$. Our inductive assumption allows us to solve the local problem

$$\mathsf{LIP}(Q_\nu, E, f, x_\nu, P_\nu)$$

for each ν. Using a partition of unity, we patch together the solutions F_ν to the above local problems, and hope that the resulting function F solves our problem $\mathsf{LIP}(Q, E, f, x_0, P_0)$. It works provided we do a good job of picking the jets P_ν. We refer the reader to the introduction of [18] for some of the ideas involved in picking the P_ν. (See especially the discussion in [18] of "keystone cubes").

Thus, we can complete our induction on $\mathcal{A}$, and solve $\mathsf{LIP}(Q, E, f, x_0, P_0)$ whenever $Q \in \mathsf{CZ}(\mathcal{A})$.

In particular, since $\mathsf{CZ}(\emptyset)$ consists of the single cube Q°, we have succeeded in solving any local interpolation problem $\mathsf{LIP}(Q, E, f, x_0, P_0)$ with $Q = Q^\circ$. As explained above, this allows us to deduce Theorems 3, 4, 5. That's the good news.

The bad news is that we cannot tell whether a given cube Q is tagged with a given label $\mathcal{A}$, since that requires perfect knowledge of the convex sets $\sigma(x, E) \subset \mathcal{P}$. Therefore, our Calderón–Zygmund decomposition $\mathsf{CZ}(\mathcal{A})$ and our proofs of Theorems 3, 4, and 5 in [18] are non-constructive.

To overcome the obstacle, we introduce here a variant of our local interpolation problem, a variant of the convex set $\sigma(x, E)$, and a modified definition of tagging of a cube Q with a label $\mathcal{A}$. We still cannot tell whether a given Q is tagged with a given $\mathcal{A}$. However, using ideas from [18], we show how to *test* Q for tagging with $\mathcal{A}$. If Q passes the test, then it is tagged with $\mathcal{A}$. If Q fails the test, then we do not know whether Q is tagged with $\mathcal{A}$, but we know that a somewhat larger cube $Q' \supset Q$ cannot be tagged with $\mathcal{A}$.

We show how to implement the above test by efficient algorithms. Moreover, if we are given dyadic cubes $Q_1 \subset Q_2 \subset \cdots \subset Q_\nu$ such that $Q_1 \cap E = \cdots = Q_\nu \cap E$, then we can test all the Q_i simultaneously. This idea is useful if our set E involves vastly different lengthscales. It provides a crucial speedup that allows us to bound the work by $N \log N$ as promised in Theorem 6.

Using the above tests, we produce a decomposition $CZ(\mathcal{A})$ analogous to the decomposition defined in [18]. This allows a constructive proof of Theorems 3, 4, and 5. To implement that proof by efficient algorithms and thus establish Theorem 6 requires additional ideas not discussed in this introduction.

This concludes our sketch of the proof of Theorem 6. We again warn the reader that it is highly oversimplified. The sections that follow, along with [20], [21], will give the correct version. In the next section, we start from scratch.

We mention several open problems related to our work.

- In place of our standing assumption $p > n$, we may assume merely that $p > n/m$. The Sobolev theorem would then tell us that $L^{m,p}(\mathbb{R}^n) \subset C^0_{loc}(\mathbb{R}^n)$. Consequently, any $F \in L^{m,p}(\mathbb{R}^n)$ may be restricted to a finite set E, and our Problems (A) and (B) still make sense. It would be very interesting to understand the problems of interpolation and extension for $L^{m,p}(\mathbb{R}^n)$ and $W^{m,p}(\mathbb{R}^n)$ when $n/m < p \le n$.
- Is it possible to dispense with the assists $\Omega = \{\omega_1, \dots, \omega_{\nu_{\max}}\}$ in Theorem 4, and write each ξ_l in the form

$$\xi_l(f) = \sum_{i=1}^{I} \beta_{li} f(z_{li})$$

  with I, β_{li} and $z_{li} \in E$ independent of f, and with $|I| \le C$? Shvartsman [37] has proven this for $\mathbb{X} = L^{2,p}(\mathbb{R}^2)$ $(p > 2)$. Perhaps, it is true for general $L^{m,p}(\mathbb{R}^n)$. The analogous result for interpolation by functions in $C^m(\mathbb{R}^n)$ is contained in Fefferman–Klartag [17]. For the extension operators T in Theorem 3, one cannot get away without assists; see [19].

  These issues are connected with “sparsification”; see [2].
- We have constructed essentially optimal functions $F \in \mathbb{X}$ that agree perfectly with a given function f on E. It would be natural to require instead that F agree with f up to a given tolerance. More precisely, given $f \in \mathbb{X}(E)$ and a positive function $\mu : E \to (0, \infty]$, we should compute a function $F \in \mathbb{X}$ that minimizes

$$\|F\|^p_{\mathbb{X}} + \sum_{x \in E} \mu(x) |F(x) - f(x)|^p, \tag{1.11}$$

up to a universal constant factor C. (When $\mu(x) = +\infty$, we demand that $F(x) = f(x)$ and delete the corresponding term from the above sum.) Compare with Fefferman–Klartag [17].

It would be interesting to study the problem of optimizing (1.11) for general $L^{m,p}(\mathbb{R}^n)$ and $W^{m,p}(\mathbb{R}^n)$ $(p > n)$. The work of P. Shvartsman [36] on the Banach space $L^{1,p}(\mathbb{R}^n) + L^p(\mathbb{R}^n, d\mu)$ is surely relevant here.

- It would be very interesting to produce practical algorithms that (unlike our present algorithms) compute C-optimal interpolants for a not-so-big universal constant C.

This paper is a part of a literature on “Whitney's extension problem”, going back over 3/4 century and including contributions by many authors. See e.g., H. Whitney [40]–[42], G. Glaeser [22], Y. Brudnyi and P. Shvartsman [5]–[8], P. Shvartsman [32]–[35], J. Wells [39], E. Le Gruyer [28], [29], M. Hirn and E. Le Gruyer [23], C. Fefferman and B. Klartag [16], [17], N. Zobin [43], [44], E. Bierstone, P. Milman, and W. Pawłucki [3], [4], as well as our own works [10]–[15], [25], and [27].

We are grateful to Bernard Chazelle, Bo'az Klartag, Assaf Naor, Pavel Shvartsman, and Nahum Zobin for many enlightening conversations. We are grateful also to the American Institute of Mathematics, the College of William and Mary, the Fields Institute, and the Banff International Research Station for hosting workshops on Whitney's problems. The support of the AFOSR and the NSF is gratefully acknowledged.

Let us now begin the work of interpolating data.

2. Preliminaries

2.1. Notation

Fix integers $m, n \geq 1$ and a real number $p > n$. We work in $\mathbb{R}^n$ with the ℓ^∞ metric. Thus, given $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ we denote

$$|x| := \max_{1 \leq i \leq n} |x_i|.$$

Given nonempty subsets $S, S' \subset \mathbb{R}^n$, we denote

$$\operatorname{dist}(S, S') := \inf\{|x - y| : x \in S, y \in S'\},$$
$$\operatorname{diam}(S) := \sup\{|x - y| : x, y \in S\}.$$

A *cube* takes the form

$$Q = [x_1 - \delta/2, x_1 + \delta/2) \times \cdots \times [x_n - \delta/2, x_n + \delta/2).$$

Let $x_Q := (x_1, \ldots, x_n)$ and $\delta_Q := \delta$ denote the center and sidelength of the cube Q, respectively. Let $A \cdot Q$ $(A > 0)$ denote the A-dilate of Q about its center. Hence, the cube $A \cdot Q$ has center x_Q and sidelength $A\delta_Q$.

A *dyadic cube* takes the form

$$Q = \left[j_1 \cdot 2^k, (j_1+1) \cdot 2^k\right) \times \cdots \times \left[j_n \cdot 2^k, (j_n+1) \cdot 2^k\right)$$

for $j_1, \ldots, j_n, k \in \mathbb{Z}$.

We say that two dyadic cubes Q and Q' *touch* either if $Q = Q'$, or if Q is disjoint from Q' but the boundaries ∂Q and $\partial Q'$ have a nonempty intersection. We write $Q \leftrightarrow Q'$ to indicate that Q touches Q'.

We may bisect a dyadic cube Q into 2^n dyadic subcubes of sidelength $\frac{1}{2}\delta_Q$ in the natural way. We call these subcubes the *children* of Q. We write Q^+ to denote the *parent* of Q, i.e., the unique dyadic cube for which Q is a child of Q^+.

We let $\mathcal{P}$ denote the vector space of real-valued $(m-1)$-st degree polynomials on $\mathbb{R}^n$, and we set $D := \dim \mathcal{P}$. We identify $\mathcal{P}$ with $\mathbb{R}^D$, by identifying $P \in \mathcal{P}$ with $(\partial^\alpha P(0))_{|\alpha| \le m-1}$.

Given $F \in C^{m-1}_{\text{loc}}(\mathbb{R}^n)$ and a point $\underline{x} \in \mathbb{R}^n$, let $J_{\underline{x}} F \in \mathcal{P}$ (the "jet of F at $\underline{x}$") denote the $(m-1)$-st order Taylor polynomial

$$(J_{\underline{x}} F)(x) = \sum_{|\alpha| \le m-1} \frac{1}{\alpha!} \partial^\alpha F(\underline{x}) \cdot (x - \underline{x})^\alpha.$$

Given $P, R \in \mathcal{P}$, we define the product $P \odot_x R = J_x(P \cdot R) \in \mathcal{P}$.

Sobolev spaces. We work with the Sobolev space $\mathbb{X} = L^{m,p}(\mathbb{R}^n)$ with seminorm

$$\|F\|_{\mathbb{X}} = \left(\int_{\mathbb{R}^n} \sum_{|\alpha|=m} |\partial^\alpha F(x)|^p \, dx \right)^{1/p}.$$

We assume throughout that $p > n$. Given a connected domain $\Omega \subset \mathbb{R}^n$ with piecewise smooth boundary, let $\mathbb{X}(\Omega) = L^{m,p}(\Omega)$ be the Sobolev space consisting of functions $F : \Omega \to \mathbb{R}$ whose distributional derivatives $\partial^\alpha F$ (for $|\alpha| = m$) belong to $L^p(\Omega)$. On this space we define the seminorm

$$\|F\|_{\mathbb{X}(\Omega)} = \left(\int_{\Omega} \sum_{|\alpha|=m} |\partial^\alpha F(x)|^p \, dx \right)^{1/p}.$$

We may restrict attention to domains that are given as the union of two intersecting rectangular boxes. (A rectangular box is a Cartesian product of left-closed, right-open intervals.)

Lists. A *list* Ξ is a collection of elements that can contain duplicates. Hence, for a list

$$\Xi = \{\xi_1, \ldots, \xi_L\} \tag{2.1}$$

we may have $\xi_\ell = \xi_{\ell'}$ for distinct ℓ, ℓ'. We define $\#[\Xi] = L$ for the list (2.1).

Given a sequence (a_ξ) of real numbers, indexed by elements ξ in Ξ, we define

$$\sum_{\xi \in \Xi} a_\xi = \sum_{\ell=1}^{L} a_{\xi_\ell}.$$

Given a sequence $\Xi_1, \dots, \Xi_M$ of lists, where $\Xi_m = \{\xi_1^{[m]}, \dots, \xi_{L_m}^{[m]}\}$ for each $1 \leq m \leq M$, we define the list $\Xi_1 \cup \dots \cup \Xi_M$ by taking all the elements in the respective sublists together, namely

$$\Xi_1 \cup \dots \cup \Xi_M := \{\xi_1^{[1]}, \dots, \xi_{L_1}^{[1]}, \dots, \xi_1^{[M]}, \dots, \xi_{L_M}^{[M]}\}.$$

We do not remove duplicate elements when forming the union of lists.

Convention on constants. A *universal constant* is a positive number determined by m, n, and p. We use letters C, c, C', etc, to denote universal constants. Let $t \in \mathbb{R}$. We use the symbol $C(t)$ to denote a positive number that depends only on m, n, p, and t. A single letter or symbol may be used to denote different constants in separate occurrences.

We write $A \lesssim B$ or $A = \mathcal{O}(B)$ to indicate the estimate $A \leq CB$, and we write $A \sim B$ to indicate the estimate $C^{-1}B \leq A \leq CB$. Here, the constant C depends only on m, n, and p.

Similarly, we write $A \lesssim_t B$ or $A = \mathcal{O}_t(B)$ to indicate the estimate $A \leq C(t) \cdot B$, and we write $A \sim_t B$ to indicate the estimate $C(t)^{-1} \cdot B \leq A \leq C(t) \cdot B$. Here, the constant $C(t)$ depends only on m, n, p, and t.

2.2. The infinite-precision model of computation

For infinite-precision, our model of computation consists of an idealized von Neumann computer [38] able to work with exact real numbers. We assume that a single memory cell is capable of storing an arbitrary real number with perfect precision.

We assume that each of the following operations can be carried out using one unit of "work".

- We read the real number stored at a given address, or entered from an input device.
- We write a real number from a register to a given memory address or to an output device.
- Given real numbers x and y, we return the numbers $x + y$, $x - y$, xy, x/y (unless $y = 0$), $\exp(x)$, and $\ln(y)$ (if $y > 0$), and we decide whether $x < y$, $x > y$ or $x = y$.
- Given a real number x, we return the greatest integer less than or equal to x.
- Given dyadic intervals $I = [x, y)$ and $J = [a, b)$, both contained in $[0, \infty)$, we return the smallest dyadic interval containing both I and J.

The above model of computation is subject to serious criticism, even without our assumption on the rapid processing of dyadic intervals. (See [17], [24], [31].) Therefore, in [21], we will give a model of computation in finite-precision. We believe that our finite-precision model faithfully reflects a subset of the capabilities of an actual computer (e.g., we don't assume any possibility of parallel processing).

Presumably, few readers will want to wade through the issues arising from implementing our algorithms in finite-precision. Hence, we first present our results assuming the above infinite-precision model of computation. We then explain how to modify our algorithms in order to succeed in the finite-precision model. These modifications are described in [21].

2.3. Basic estimates on Sobolev functions

Let $\mathcal{P}$ denote the vector space of $(m-1)^{\mathrm{st}}$ degree polynomials.

Given a point $x \in \mathbb{R}^n$ and a number $\delta > 0$, we define

$$|P|_{x,\delta} := \max_{|\beta| \le m-1} |\partial^\beta P(x)| \cdot \delta^{|\beta| + n/p - m} \quad \text{for } P \in \mathcal{P}.$$

Thus, $|P|_{x,\delta}$ is a norm on $\mathcal{P}$. The unit ball of this norm is $\mathcal{B}(x,\delta) := \{P \in \mathcal{P} : |P|_{x,\delta} \le 1\}$.

We next present a few basic properties of the objects defined above.

Lemma 7. *Let $Q \subset \mathbb{R}^n$ be a cube and let $K \ge 1$. For any polynomial $P \in \mathcal{P}$, the following estimates hold:*

$$\delta_Q^{-m} \|P\|_{L^p(Q)} \sim \sum_{k=0}^{m-1} \delta_Q^{k-m} \|\nabla^k P\|_{L^p(Q)} \sim_K |P|_{x,\delta} \quad \textit{for } x \in KQ,\ \delta \in [K^{-1}\delta_Q, K\delta_Q].$$

$$|P|_{x',\delta'} \le C(K) \cdot |P|_{x,\delta} \qquad \textit{for } |x - x'| \le K\delta' \textit{ and } \delta \le K\delta'.$$

$$\mathcal{B}(x,\delta) \subseteq C(K) \cdot \mathcal{B}(x',\delta') \quad \textit{for } |x - x'| \le K\delta' \textit{ and } \delta \le K\delta'.$$

Here, $C(K)$ depends only on $m, n, p,$ and K.

We present a few useful estimates on functions in $\mathbb{X}(Q) = L^{m,p}(Q)$, where $Q \subset \mathbb{R}^n$ is a cube. These estimates originate from the classical Sobolev inequality (Proposition 8) and an interpolation inequality (Proposition 9).

Proposition 8 (Sobolev inequality). *Let $Q \subset \mathbb{R}^n$ be a cube, and let $F \in \mathbb{X}(Q)$. For any $x, y \in Q$, and any multiindex β with $|\beta| \le m-1$, we have*

$$\left|\partial^\beta (J_x F - F)(y)\right| \le C \cdot \delta_Q^{m-|\beta|-n/p} \|F\|_{\mathbb{X}(Q)}.$$

Proof. The estimate is an easy consequence of the Sobolev embedding theorem and Taylor's theorem.

We first review the Sobolev embedding theorem. Let $F \in \mathbb{X}(Q) = L^{m,p}(Q)$. Due to our standing assumption that $p > n$, we can use the Sobolev embedding theorem (see [30]), which implies that F belongs to the Hölder space[1] $C^{m-1,1-\frac{n}{p}}(Q)$, and moreover we have an estimate on the Hölder seminorm:

$$\|F\|_{C^{m-1,1-n/p}(Q)} \leq K \cdot \|F\|_{L^{m,p}(Q)}. \tag{2.2}$$

Here, $K = K(m, n, p)$. In particular, K is independent of Q. We refer the reader to [30] for a proof of the estimate (2.2) when $Q = [0, 1)^n$. We can prove (2.2) for the same choice of K and a general $Q \subset \mathbb{R}^n$ using a standard rescaling argument. We provide details below.

Let $Q \subset \mathbb{R}^n$, and let $F \in L^{m,p}(Q)$. Let $\tau : \mathbb{R}^n \to \mathbb{R}^n$ be a transformation of the form $\tau(x) = R\cdot x+x_0$ ($R > 0$, $x_0 \in \mathbb{R}^n$) satisfying that τ maps $Q^\circ := [0, 1)^n$ onto Q. We define a transformed function $\widetilde{F} = F \circ \tau : Q^\circ \mapsto \mathbb{R}$. The Sobolev and Hölder norms relevant to our discussion are transformed in a simple fashion. Indeed, by a change of variables we have

$$\begin{aligned}\|\widetilde{F}\|_{L^{m,p}(Q^\circ)} &= \left(\int_{Q^\circ} |\nabla^m(F(R \cdot x + x_0))|^p \, dx\right)^{1/p} \\ &= R^{m-n/p}\left(\int_{Q} |\nabla^m F(x)|^p \, dx\right)^{1/p} = R^{m-n/p}\|F\|_{L^{m,p}(Q)}.\end{aligned}$$

Similarly, we have $\|\widetilde{F}\|_{C^{m-1,1-n/p}(Q^\circ)} = R^{m-n/p}\|F\|_{C^{m-1,1-n/p}(Q)}$.

We apply the known version of the estimate (2.2) to the function $\widetilde{F}$. Thus, we obtain $\|\widetilde{F}\|_{C^{m-1,1-n/p}(Q^\circ)} \leq K \cdot \|\widetilde{F}\|_{L^{m,p}(Q^\circ)}$. From the above equations we thus deduce that

$$\|F\|_{C^{m-1,1-n/p}(Q)} \leq K \cdot \|F\|_{L^{m,p}(Q)}.$$

This completes the proof of (2.2).

Recall that J_xF is the $(m-1)^{\text{st}}$ degree Taylor polyomial of F at x. Hence, by definition,

$$J_xF(y) = \sum_{|\alpha|\leq m-1} \frac{1}{\alpha!}\partial^\alpha F(x) \cdot (y-x)^\alpha \quad (y \in \mathbb{R}^n). \tag{2.3}$$

Taylor's theorem states that

$$\left|\partial^\beta(J_xF - F)(y)\right| \leq C \cdot |x-y|^{m-1+\gamma-|\beta|} \cdot \|F\|_{C^{m-1,\gamma}(Q)}$$

for any $F \in C^{m-1,\gamma}(Q)$, any $x, y \in Q$, and any multiindex β with $|\beta| \leq m-1$. Here, $C = C(m, n, \gamma)$ depends only on m, n, and γ. We apply this estimate with $\gamma = 1 - n/p$ and use (2.2) to see that

$$\left|\partial^\beta(J_xF - F)(y)\right| \leq CK \cdot |x-y|^{m-n/p-|\beta|} \cdot \|F\|_{L^{m,p}(Q)}$$

for any $F \in L^{m,p}(Q)$. We bound $|x - y| \leq \delta_Q$ to prove the desired estimate and complete the proof of Proposition 8. □

[1]Here, $C^{m-1,\gamma}(Q)$ ($\gamma \in (0, 1]$) is the Hölder space consisting of all functions $F : Q \to \mathbb{R}$ that satisfy the estimate $|\partial^\alpha F(x) - \partial^\alpha F(y)| \leq A \cdot |x-y|^\gamma$ for some $A < \infty$ and for all multiindices α with $|\alpha| = m-1$ and all $x, y \in Q$. The Hölder seminorm of F in $C^{m-1,\gamma}(Q)$ is defined to be the infimum of all such constants A.

We can formulate the Sobolev inequality as an estimate involving the norms $|\cdot|_{x,\delta}$ introduced earlier. Indeed,

$$|J_x F - J_y F|_{y,\delta_Q} \leq C\, \|F\|_{\mathbb{X}(Q)} \quad \text{whenever } x, y \in Q.$$

Proposition 9. *Let $Q \subset \mathbb{R}^n$ be a cube, and let $F \in \mathbb{X}(Q)$. For any multiindex β with $|\beta| \leq m$, we have*

$$\|\partial^\beta F\|_{L^p(Q)} \leq C \cdot \left[\delta_Q^{-|\beta|} \|F\|_{L^p(Q)} + \delta_Q^{m-|\beta|} \|F\|_{\mathbb{X}(Q)}\right].$$

Proof. A standard scaling argument allows us to reduce to the case when $Q = [0,1)^n$. For a proof of this estimate, see [30]. □

Lemma 10. *Let $Q \subset \mathbb{R}^n$ be a cube, and let $F \in \mathbb{X}(Q)$. For any $x \in Q$, we have*

$$\delta_Q^{-m} \|F\|_{L^p(Q)} \leq C \cdot \left[\|F\|_{\mathbb{X}(Q)} + \sum_{|\beta| \leq m-1} |\partial^\beta F(x)| \delta_Q^{|\beta| + n/p - m}\right] \leq C' \cdot \left[\|F\|_{\mathbb{X}(Q)} + \delta_Q^{-m} \|F\|_{L^p(Q)}\right]. \tag{2.4}$$

For any cube $Q' \subset Q$, we have

$$\delta_Q^{-m} \|F\|_{L^p(Q)} \leq C'' \cdot \left[\delta_{Q'}^{-m} \|F\|_{L^p(Q')} + \|F\|_{\mathbb{X}(Q)}\right]. \tag{2.5}$$

Here, C, C', C'' denote constants depending only on m, n, and p.

Proof. Let $Q \subset \mathbb{R}^n$, and let $x, y \in Q$.

From the Sobolev inequality and the definition (2.3) of the Taylor polyomial, we have

$$\begin{aligned} |F(y)| &\leq |F(y) - J_x F(y)| + |J_x F(y)| \\ &\lesssim \delta_Q^{m-n/p} \|F\|_{\mathbb{X}(Q)} + \sum_{|\beta| \leq m-1} |\partial^\beta F(x)| \cdot |x-y|^{|\beta|}. \end{aligned}$$

Hence,

$$\begin{aligned} \int_Q |F(y)|^p \, dy &\lesssim \delta_Q^{mp-n} \|F\|_{\mathbb{X}(Q)}^p \cdot \mathrm{Vol}(Q) + \sum_{|\beta| \leq m-1} \left|\partial^\beta F(x)\right|^p \cdot \int_Q |x-y|^{|\beta| \cdot p} \, dy \\ &\lesssim \delta_Q^{mp} \|F\|_{\mathbb{X}(Q)}^p + \sum_{|\beta| \leq m-1} \left|\partial^\beta F(x)\right|^p \cdot \delta_Q^{|\beta| \cdot p + n} \end{aligned}$$

We raise each side to the power $1/p$. This implies the first estimate in (2.4).

We now complete the proof of (2.4). Let $|\beta| \leq m-1$, and let $x, y \in Q$. As before, from the Sobolev inequality and (2.3) we have

$$\begin{aligned} \left|\partial^\beta F(x)\right| &\leq \left|\partial^\beta F(x) - \partial^\beta (J_y F)(x)\right| + \left|\partial^\beta (J_y F)(x)\right| \\ &\lesssim \delta_Q^{m-|\beta|-n/p} \|F\|_{\mathbb{X}(Q)} + \sum_{|\gamma| \leq m-1-|\beta|} \left|\partial^{\beta+\gamma} F(y)\right| \cdot |x-y|^{|\gamma|} \\ &\leq \delta_Q^{m-|\beta|-n/p} \|F\|_{\mathbb{X}(Q)} + \sum_{|\gamma| \leq m-1-|\beta|} \left|\partial^{\beta+\gamma} F(y)\right| \cdot \delta_Q^{|\gamma|}. \end{aligned}$$

We raise this estimate to the power p and average over $y \in Q$. Hence,

$$|\partial^\beta F(x)| \lesssim \delta_Q^{m-|\beta|-n/p} \|F\|_{\mathbb{X}(Q)} + \sum_{|\gamma| \le m-1-|\beta|} \delta_Q^{|\gamma|-n/p} \|\partial^{\beta+\gamma} F\|_{L^p(Q)}.$$

We apply Proposition 9 to estimate the terms in the sum over γ. We see that $\|\partial^{\beta+\gamma}F\|_{L^p(Q)}$ is bounded by $C \cdot \left[\delta_Q^{m-|\beta|-|\gamma|} \cdot \|F\|_{\mathbb{X}(Q)} + \delta_Q^{-|\beta|-|\gamma|} \cdot \|F\|_{L^p(Q)}\right]$.

Thus, we conclude that

$$\left|\partial^\beta F(x)\right| \lesssim \delta_Q^{m-|\beta|-n/p} \|F\|_{\mathbb{X}(Q)} + \delta_Q^{-|\beta|-n/p} \|F\|_{L^p(Q)}.$$

We thus obtain the second estimate in (2.4).

We will finally prove the inequality (2.5). Let $Q' \subset Q$ be given cubes. Then (2.4) implies that

$$\delta_Q^{-m} \|F\|_{L^p(Q)} \lesssim \|F\|_{\mathbb{X}(Q)} + \sum_{|\beta| \le m-1} |\partial^\beta F(x)| \cdot \delta_{Q'}^{|\beta|+n/p-m} \quad \text{for any } x \in Q'.$$

(Here, we use that $\delta_{Q'} \le \delta_Q$ and $|\beta| + n/p - m < 0$.) By averaging p-th powers in the above estimate, we see that

$$\delta_Q^{-m} \|F\|_{L^p(Q)} \lesssim \|F\|_{\mathbb{X}(Q)} + \sum_{|\beta| \le m-1} \delta_{Q'}^{|\beta|-m} \|\partial^\beta F\|_{L^p(Q')}.$$

Finally, we apply Proposition 9 to estimate the terms in the sum over β. Thus, we see that $\delta_{Q'}^{|\beta|-m}\|\partial^\beta F\|_{L^p(Q')}$ $(0 \le |\beta| \le m-1)$ is bounded by $C \cdot \left[\|F\|_{\mathbb{X}(Q)} + \delta_{Q'}^{-m}\|F\|_{L^p(Q')}\right]$. This completes the proof of (2.5). This concludes the proof of Lemma 10. $\square$

Lemma 11. *Let $Q', Q'' \subset \mathbb{R}^n$ be cubes with intersecting closures, and suppose that $\frac{1}{2}\delta_{Q''} \le \delta_{Q'} \le 2\delta_{Q''}$. Then for any $R', R'' \in \mathcal{P}$ and any $H \in \mathbb{X}$, we have*

$$\begin{aligned}\delta_{Q'}^{-m}\|R' - R''\|_{L^p(Q')} \lesssim\ & \delta_{Q'}^{-m} \|H - R'\|_{L^p(\frac{65}{64}Q')} + \delta_{Q''}^{-m} \|H - R''\|_{L^p(\frac{65}{64}Q'')} \\ & + \|H\|_{\mathbb{X}(\frac{65}{64}Q')} + \|H\|_{\mathbb{X}(\frac{65}{64}Q'')}.\end{aligned}$$

Proof. First we write

$$\delta_{Q'}^{-m}\|R' - R''\|_{L^p(Q')} \le \delta_{Q'}^{-m}\|H - R'\|_{L^p(\frac{65}{64}Q')} + \delta_{Q'}^{-m}\|H - R''\|_{L^p(\frac{65}{64}Q')}.$$

For any fixed $x \in \frac{65}{64}Q' \cap \frac{65}{64}Q''$, Lemma 10 gives that

$$\delta_{Q'}^{-m}\|H - R''\|_{L^p(\frac{65}{64}Q')} \lesssim \|H\|_{\mathbb{X}(\frac{65}{64}Q')} + \sum_{|\beta| \le m-1} |\partial^\beta (H - R'')(x)| \delta_{Q'}^{|\beta|+\frac{n}{p}-m}$$

and also

$$\sum_{|\beta| \le m-1} |\partial^\beta (H - R'')(x)| \delta_{Q'}^{|\beta|+\frac{n}{p}-m} \lesssim \|H\|_{\mathbb{X}(\frac{65}{64}Q'')} + \delta_{Q''}^{-m}\|H - R''\|_{L^p(\frac{65}{64}Q'')}.$$

This implies the conclusion of the lemma. $\square$

A *rectangular box* $B \subset \mathbb{R}^n$ is a Cartesian product of coordinate intervals that are left-closed and right-open, where each interval has a nonempty interior. The length of each interval is a *sidelength* of B. The *aspect ratio* of B is the ratio of the longest to shortest sidelength of B.

Let $K \geq 1$. Suppose that B is a rectangular box with aspect ratio at most K. We can map a cube onto B by applying a transformation of the form $\tau : (x_1, \dots, x_n) \mapsto (\delta_1 x_1, \dots, \delta_n x_n)$, with $1 \leq |\delta_j| \leq K$ for $j = 1, \dots, n$. Let $F \in \mathbb{X}(B)$. By pre-composing F with the transformation τ, and using Proposition 8 (the Sobolev inequality), we see that

$$|\partial^\beta (F - J_y F)(x)| \leq C(K) \cdot \|F\|_{\mathbb{X}(B)} |x - y|^{m - n/p - |\beta|} \tag{2.6}$$

for all $x, y \in B$, $|\beta| \leq m - 1$. We raise this estimate to the power p, and integrate over $x \in B$ to obtain

$$\|\partial^\beta (F - J_y F)\|_{L^p(B)} \leq C(K) \cdot \|F\|_{\mathbb{X}(B)} \operatorname{diam}(B)^{m - |\beta|}. \tag{2.7}$$

Lemma 12. *Let B_1, B_2 be rectangular boxes with aspect ratio at most K and with $B_1 \cap B_2 \neq \emptyset$. Then for any $F \in \mathbb{X}(B_1 \cup B_2)$, any $x, y \in B_1 \cup B_2$, and any β with $|\beta| \leq m - 1$, we have*

$$|\partial^\beta (J_x F - F)(y)| \leq C(K) \cdot |x - y|^{m - |\beta| - n/p} \cdot \left\{ \|F\|_{\mathbb{X}(B_1)} + \|F\|_{\mathbb{X}(B_2)} \right\}. \tag{2.8}$$

Proof. If either $x, y \in B_1$ or $x, y \in B_2$ then (2.8) follows from the estimate (2.6).

Otherwise, we may assume that $x \in B_1$ and $y \in B_2$. Pick $z \in B_1 \cap B_2$ with the property that $|x - z| \leq |x - y|$ and $|y - z| \leq |x - y|$, and note that

$$\begin{aligned} |J_x F - J_y F|_{y, |x-y|} &\leq |J_x F - J_z F|_{y, |x-y|} + |J_z F - J_y F|_{y, |x-y|} \\ &\lesssim |J_x F - J_z F|_{z, |x-z|} + |J_z F - J_y F|_{z, |y-z|} \quad \text{(by Lemma 7)}. \end{aligned}$$

Now, (2.6) implies that

$$|J_x F - J_y F|_{y, |x-y|} \leq C(K) \cdot \left\{ \|F\|_{\mathbb{X}(B_1)} + \|F\|_{\mathbb{X}(B_2)} \right\}.$$

This completes the proof of the lemma. □

Our last result is a special case of the Jones extension theorem [26].

Proposition 13. *Let Q be a cube in $\mathbb{R}^n$. Then there exists a bounded linear map $T\colon \mathbb{X}(Q) \to \mathbb{X}$, such that $T(F) = F$ on Q, and $\|T(F)\|_{\mathbb{X}} \leq C \|F\|_{\mathbb{X}(Q)}$ for each $F \in \mathbb{X}(Q)$. Here, the constant C depends only on the parameters of the function space $\mathbb{X}$, i.e., on the numbers m, n, p.*

2.4. Trace norms

Given a finite subset $E \subset \mathbb{R}^n$, let $\mathbb{X}(E)$ denote the space of all functions $f : E \to \mathbb{R}$, with the trace seminorm

$$\|f\|_{\mathbb{X}(E)} := \inf\{\|F\|_{\mathbb{X}} : F \in \mathbb{X}, \quad F = f \text{ on } E\}.$$

Given a cube $Q \subseteq \mathbb{R}^n$, and given $(f, P) \in \mathbb{X}(E) \oplus \mathcal{P}$, let

(2.9)
$$\|(f,P)\|_Q := \inf\left\{ M \geq 0 \ : \ \exists\, F \in \mathbb{X} \quad \text{s.t.} \quad \begin{array}{c} F = f \text{ on } E \cap Q \\ \|F\|_{\mathbb{X}(Q)} + \delta_Q^{-m} \|F - P\|_{L^p(Q)} \leq M \end{array} \right\}.$$

Note that $\|(f,P)\|_Q$ is a seminorm on the space $\mathbb{X}(E) \oplus \mathcal{P}$.

Let

$$\sigma(Q) := \left\{ P \in \mathcal{P} \ : \ \exists\, \varphi \in \mathbb{X} \quad \text{s.t.} \quad \begin{array}{c} \varphi = 0 \text{ on } E \cap Q \\ \|\varphi\|_{\mathbb{X}(Q)} + \delta_Q^{-m} \|\varphi - P\|_{L^p(Q)} \leq 1 \end{array} \right\}. \tag{2.10}$$

Note that $\sigma(Q) \subset \mathcal{P}$ is convex and symmetric ($P \in \sigma(Q) \implies -P \in \sigma(Q)$).

As an easy consequence of our definitions, we have the following result.

Lemma 14. *Given cubes $Q_1 \subset Q_2$ such that $\delta_{Q_2} \leq A\delta_{Q_1}$, we have $\|(f,P)\|_{Q_1} \leq C(A) \cdot \|(f,P)\|_{Q_2}$ and $\sigma(Q_2) \subset C(A) \cdot \sigma(Q_1)$. In fact, one can take $C(A) = A^m$.*

Our next result relates the convex sets $\sigma(Q_1)$ and $\sigma(Q_2)$, where $Q_1 \subset Q_2$ are cubes that may have vastly different sizes.

Lemma 15. *Given cubes $Q_1 \subset Q_2$, we have*

$$\sigma(Q_2) \subset C \cdot [\sigma(Q_1) + \mathcal{B}(x_{Q_1}, \delta_{Q_2})]\,. \tag{2.11}$$

If additionally $Q_1 \cap E = Q_2 \cap E$, then also

$$c \cdot [\sigma(Q_1) + \mathcal{B}(x_{Q_1}, \delta_{Q_2})] \subset \sigma(Q_2). \tag{2.12}$$

Proof. Suppose that $R \in \sigma(Q_2)$. By definition, this means that there exists $\phi \in \mathbb{X}$ such that $\phi = 0$ on $Q_2 \cap E$, and $\|\phi\|_{\mathbb{X}(Q_2)} + \delta_{Q_2}^{-m} \|\phi - R\|_{L^p(Q_2)} \leq 1$. In particular, we have $\phi = 0$ on $Q_1 \cap E$. Also, the Sobolev inequality implies that

$$\|\phi\|_{\mathbb{X}(Q_1)} + \delta_{Q_1}^{-m} \|\phi - J_{x_{Q_1}}\phi\|_{L^p(Q_1)} \leq C\, \|\phi\|_{\mathbb{X}(Q_1)} \leq C\, \|\phi\|_{\mathbb{X}(Q_2)} \leq C.$$

Hence, $J_{x_{Q_1}}\phi \in C\sigma(Q_1)$. (Recall that x_{Q_1} is the center of the cube Q_1.)

Similarly, using the triangle inequality followed by the Sobolev inequality, we have

$$\begin{aligned} \delta_{Q_2}^{-m} \|J_{x_{Q_1}}\phi - R\|_{L^p(Q_2)} &\leq \delta_{Q_2}^{-m} \|\phi - R\|_{L^p(Q_2)} + \delta_{Q_2}^{-m} \|\phi - J_{x_{Q_1}}\phi\|_{L^p(Q_2)} \\ &\leq \delta_{Q_2}^{-m} \|\phi - R\|_{L^p(Q_2)} + C\|\phi\|_{\mathbb{X}(Q_2)} \leq 1 + C. \end{aligned}$$

Thus, $J_{x_{Q_1}}\phi - R \in C \cdot \mathcal{B}(x_{Q_1}, \delta_{Q_2})$.

Hence, we have shown that $R = J_{x_{Q_1}}\phi + (R - J_{x_{Q_1}}\phi) \in C\sigma(Q_1) + C\mathcal{B}(x_{Q_1}, \delta_{Q_2})$. Since $R \in \sigma(Q_2)$ was arbitrary, this proves the first inclusion (2.11).

We now assume that $Q_1 \cap E = Q_2 \cap E$ and prove the second inclusion (2.12).

Let $R \in \sigma(Q_1) + \mathcal{B}(x_{Q_1}, \delta_{Q_2})$, i.e., suppose that $R = P + P^\#$ with $P \in \sigma(Q_1)$ and $P^\# \in \mathcal{B}(x_{Q_1}, \delta_{Q_2})$.

By definition, $P^{\#} \in \mathcal{B}(x_{Q_1}, \delta_{Q_2})$ implies that $|P^{\#}|_{x_{Q_1},\delta_{Q_2}} \leq 1$, hence

$$\delta_{Q_2}^{-m} \|P^{\#}\|_{L^p(Q_2)} \leq C \quad \text{(see Lemma 7).} \tag{2.13}$$

By definition, $P \in \sigma(Q_1)$ implies that there exists $\varphi \in \mathbb{X}$ such that $\varphi = 0$ on $Q_1 \cap E$ and $\|\varphi\|_{\mathbb{X}(Q_1)} + \delta_{Q_1}^{-m}\|\varphi - P\|_{L^p(Q_1)} \leq 1$. We now pick $\widetilde{\varphi} \in \mathbb{X}$ with $\widetilde{\varphi} = \varphi$ on Q_1 and $\|\widetilde{\varphi}\|_{\mathbb{X}} \leq C\|\varphi\|_{\mathbb{X}(Q_1)}$. (See Proposition 13.) Note that $\widetilde{\varphi} = 0$ on $Q_1 \cap E = Q_2 \cap E$. Moreover,

$$\begin{aligned} \delta_{Q_2}^{-m} \|\widetilde{\varphi} - R\|_{L^p(Q_2)} &\leq \delta_{Q_2}^{-m} \|\widetilde{\varphi} - P\|_{L^p(Q_2)} + \delta_{Q_2}^{-m} \|P^{\#}\|_{L^p(Q_2)} \\ &\leq \delta_{Q_2}^{-m} \|\widetilde{\varphi} - P\|_{L^p(Q_2)} + C && \text{(by (2.13))} \\ &\leq C\delta_{Q_1}^{-m}\|\widetilde{\varphi} - P\|_{L^p(Q_1)} + C\|\widetilde{\varphi}\|_{\mathbb{X}(Q_2)} + C && \text{(by Lemma 10)} \\ &\leq C'\delta_{Q_1}^{-m}\|\varphi - P\|_{L^p(Q_1)} + C'\|\varphi\|_{\mathbb{X}(Q_1)} + C' \leq C''. \end{aligned}$$

Since we also have $\|\widetilde{\varphi}\|_{\mathbb{X}(Q_2)} \leq C$, it follows that $R \in C\sigma(Q_2)$. Since $R \in \sigma(Q_1) + \mathcal{B}(x_{Q_1}, \delta_{Q_2})$ was arbitrary, this proves (2.12) and completes the proof of the lemma. □

2.5. The depth of linear maps

Let $E = \{z_1, \dots, z_N\} \subset \mathbb{R}^n$. We fix this enumeration of E for the rest of the paper.

A linear functional $\omega : X(E) \to \mathbb{R}$ may be written as

$$\omega(f) = \sum_{j=1}^{N} \mu_j \cdot f(z_j) \quad \text{for real coefficients } \mu_1, \dots, \mu_N. \tag{2.14}$$

That is the *long form of* ω. Let depth(ω) ("the depth of ω") be the number of nonzero coefficients μ_j in (2.14).

Suppose depth(ω) $= d$. Then let $1 \leq j_1 < j_2 < \cdots < j_d \leq N$ be the indices for which $\mu_j \neq 0$ above. Also, let $\widetilde{\mu}_k = \mu_{j_k}$ for $k = 1, \dots, d$. Then we can write ω in the form

$$\omega(f) = \sum_{k=1}^{d} \widetilde{\mu}_k \cdot f(z_{j_k}).$$

That is the *short form of* ω.

To store ω in its long form, we store $\mu_1, \dots, \mu_N$.

To store ω in its short form, we store d, $\widetilde{\mu}_1, \dots, \widetilde{\mu}_d$, and $j_1, \dots, j_d$.

Let $\Omega = \{\omega_1, \dots, \omega_K\}$ be a list of linear functionals on $\mathbb{X}(E)$, each given in short form. Recall that a list may contain duplicates. Hence, we can have $\omega_k = \omega_{k'}$ with $k \neq k'$. We store the list Ω by storing a list of pointers to the functionals in Ω. If we have stored two lists of functionals Ω and Ω', then we can compute and store $\Omega \cup \Omega'$ using work at most $C \cdot [\#(\Omega) + \#(\Omega')]$.

A functional $\xi : \mathbb{X}(E) \oplus \mathcal{P} \to \mathbb{R}$ may be written in the form

$$\xi(f, P) = \lambda(P) + \sum_{j=1}^{N} \mu_j f(z_j) \tag{2.15}$$

for coefficients $\mu_1, \dots, \mu_N$ and a functional $\lambda : \mathcal{P} \to \mathbb{R}$. That's the *long form of* ξ.

Let $d \in \mathbb{N}$. A functional $\xi : \mathbb{X}(E) \oplus \mathcal{P} \to \mathbb{R}$ has Ω-*assisted depth* d provided that

$$\xi(f, P) = \lambda(P) + \eta(f) + \sum_{\nu=1}^{\nu_{\max}} \gamma_\nu \, \omega_{k_\nu}(f), \tag{2.16}$$

where $\eta : \mathbb{X}(E) \to \mathbb{R}$ is a linear functional, and $\text{depth}(\eta) + \nu_{\max} \le d$. That is a *short form of* ξ in terms of the assists Ω. Note that perhaps we can describe a given ξ in many different ways in short form.

To store the long form of ξ, we store λ, $\mu_1, \dots, \mu_N$. See (2.15).

To store a short form of ξ (in terms of the assists Ω), we store $\lambda, \nu_{\max}$, $\gamma_1, \dots, \gamma_{\nu_{\max}}$, $k_1, \dots, k_{\nu_{\max}}$, and the short form of η. See (2.16).

A linear map $S : \mathbb{X}(E) \oplus \mathcal{P} \to \mathcal{P}$ has Ω-*assisted depth* d provided that

$$(f, P) \mapsto \partial^\alpha \left[S(f, P)\right](0) \quad \text{has } \Omega\text{-assisted depth } d, \text{ for each } |\alpha| \le m - 1.$$

To store a short form of the map $(f, P) \mapsto S(f, P)$, we store a short form of each of the linear functionals $(f, P) \mapsto \partial^\alpha \left[S(f, P)\right](0)$ (for $|\alpha| \le m - 1$).

A linear map $T : \mathbb{X}(E) \oplus \mathcal{P} \to \mathbb{X}$ has Ω-*assisted depth* d provided that

$$(f, P) \mapsto \partial^\alpha \left[T(f, P)\right](x) \quad \text{has } \Omega\text{-assisted depth } d, \text{ for each } x \in \mathbb{R}^n, \ |\alpha| \le m - 1.$$

We can represent the map T on a computer by giving an algorithm that accepts queries: A query consists of a point $x \in \mathbb{R}^n$. The response to a query is a short form of each of the linear functionals $(f, P) \mapsto \partial^\alpha \big[T(f, P)\big](x)$ (for $|\alpha| \le m - 1$).

When we say that a linear functional ω has *bounded depth*, we mean that its depth is bounded by a universal constant C.

When we say that a linear map T (or linear functional ξ) has Ω-*assisted bounded depth*, we mean that T (or ξ) has Ω-assisted depth d, where d is at most a universal constant C.

2.6. Sets of multi-indices

Let $\mathcal{M}$ denote the collection of all multiindices $\alpha = (\alpha_1, \dots, \alpha_n)$ of order $|\alpha| = \alpha_1 + \dots + \alpha_n \le m - 1$.

We define a total order relation $<$ on $\mathcal{M}$ as follows: Given distinct $\alpha = (\alpha_1, \dots, \alpha_n), \beta = (\beta_1, \dots, \beta_n) \in \mathcal{M}$, let $k \in \{1, \dots, n\}$ be the maximal index such that $\alpha_1 + \dots + \alpha_k \ne \beta_1 + \dots + \beta_k$. Then we write $\alpha < \beta$ if $\alpha_1 + \dots + \alpha_k < \beta_1 + \dots + \beta_k$, and we write $\alpha > \beta$ otherwise.

We also define a total order relation $<$ on $2^{\mathcal{M}}$. Given distinct subsets $\mathcal{A}, \mathcal{B} \subset \mathcal{M}$, pick the minimal element $\alpha \in \mathcal{A} \Delta \mathcal{B}$ (with respect to the order relation defined above). Then we write $\mathcal{A} < \mathcal{B}$ if $\alpha \in \mathcal{A}$, and we write $\mathcal{B} < \mathcal{A}$ otherwise. Here, $\mathcal{A} \Delta \mathcal{B}$ denotes the symmetric difference $(\mathcal{A} \setminus \mathcal{B}) \cup (\mathcal{B} \setminus \mathcal{A})$. Note that $\mathcal{M}$ is minimal and that the empty set $\emptyset$ is maximal with respect to this order relation on $2^{\mathcal{M}}$.

Lemma 16. *The following properties hold.*

- *If* $\alpha, \beta \in \mathcal{M}$ *and* $|\alpha| < |\beta|$ *then* $\alpha < \beta$.
- *If* $\alpha, \beta \in \mathcal{M}$, $\alpha < \beta$ *and* $|\gamma| \le m - 1 - |\beta|$, *then* $\alpha + \gamma < \beta + \gamma$.

Given $\mathcal{A} \subset \mathcal{M}$, we say that $\mathcal{A}$ is *monotonic* if for every $\alpha \in \mathcal{A}$ and $\gamma \in \mathcal{M}$ with $|\gamma| \leq m - 1 - |\alpha|$, we have $\alpha + \gamma \in \mathcal{A}$.

Remark 17. Assume that $\mathcal{A} \subset \mathcal{M}$ is monotonic, $P \in \mathcal{P}$, $x_0 \in \mathbb{R}^n$, and $\partial^\alpha P(x_0) = 0$ for all $\alpha \in \mathcal{A}$. Then, for $x \in \mathbb{R}^n$ and $\alpha \in \mathcal{A}$, we have

$$\partial^\alpha P(x) = \sum_{|\gamma| \leq m-1-|\alpha|} \frac{1}{\gamma!} \partial^{\alpha+\gamma} P(x_0) \cdot (x - x_0)^\gamma = 0.$$

Hence, $\partial^\alpha P \equiv 0$ for any $\alpha \in \mathcal{A}$.

2.7. Bases for the space of polynomials

Let $\epsilon \in (0, 1)$ be a given real number. We assume throughout this section that

$$\epsilon < \text{ small enough constant determined by } m, n, p.$$

2.7.1. Bases. Suppose we are given the following.

- A set of multiindices $\mathcal{A} \subset \mathcal{M}$.
- A collection of polynomials $(P_\alpha)_{\alpha \in \mathcal{A}}$ with each $P_\alpha \in \mathcal{P}$.
- A symmetric convex subset $\sigma \subset \mathcal{P}$.
- A point $x \in \mathbb{R}^n$, and real numbers $\Lambda \geq 1$, $\delta > 0$ (we call δ a "lengthscale").

We say that $(P_\alpha)_{\alpha \in \mathcal{A}}$ forms an $(\mathcal{A}, x, \epsilon, \delta)$-*basis for* σ if the following conditions are met.

(B1) $P_\alpha \in \epsilon \cdot \delta^{|\alpha|+n/p-m} \cdot \sigma$ for all $\alpha \in \mathcal{A}$.

(B2) $\partial^\beta P_\alpha(x) = \delta_{\alpha\beta}$ for all $\alpha, \beta \in \mathcal{A}$.

(B3) $|\partial^\beta P_\alpha(x)| \leq \epsilon \cdot \delta^{|\alpha|-|\beta|}$ for all $\alpha \in \mathcal{A}$, $\beta \in \mathcal{M}$, $\beta > \alpha$.

(Here, $\delta_{\alpha\beta}$ denotes the Kronecker delta: $\delta_{\alpha\beta} = 1$ if $\alpha = \beta$; $\delta_{\alpha\beta} = 0$ if $\alpha \neq \beta$.) We say that $(P_\alpha)_{\alpha \in \mathcal{A}}$ forms an $(\mathcal{A}, x, \epsilon, \delta, \Lambda)$-*basis for* σ if, in addition to **(B1)**–**(B3)**, the following condition is met.

(B4) $|\partial^\beta P_\alpha(x)| \leq \Lambda \cdot \delta^{|\alpha|-|\beta|}$ for all $\alpha \in \mathcal{A}$, $\beta \in \mathcal{M}$.

Remark 18. An $(\mathcal{A}, x, \epsilon, \delta)$-basis is automatically an $(\mathcal{A}, x, \epsilon', \delta')$-basis, for $\epsilon' \geq \epsilon$ and $\delta' \leq \delta$. An $(\mathcal{A}, x, \epsilon, \delta, \Lambda)$-basis is automatically an $(\mathcal{A}, x, \epsilon', \delta, \Lambda')$-basis, for $\epsilon' \geq \epsilon$ and $\Lambda' \geq \Lambda$. However, there is no simple relationship between an $(\mathcal{A}, x, \epsilon, \delta, \Lambda)$-basis and an $(\mathcal{A}, x, \epsilon, \delta', \Lambda)$-basis, due to the positive powers of δ appearing in condition **(B4)**.

Note that an $(\mathcal{A}, x, \epsilon, \delta)$-basis is also an $(\mathcal{A}, x, C^m\epsilon, C\delta)$-basis for any $C \geq 1$.

Remark 19. The notion of bases admits a natural rescaling, described below.

Given $P \in \mathcal{P}$ define the polynomial $\tau_{x,\delta}(P) \in \mathcal{P}$ by

$$\tau_{x,\delta}(P)(z) = P(\delta \cdot (z - x) + x).$$

Assume that $(P_\alpha)_{\alpha\in\mathcal{A}}$ forms an $(\mathcal{A}, x, \epsilon, \delta)$-basis for a symmetric convex set $\sigma \subset \mathcal{P}$. Define the rescaled polynomials $\overline{P}_\alpha = \delta^{-|\alpha|}\tau_{x,\delta}(P_\alpha)$ for $\alpha \in \mathcal{A}$. Also define the convex set of polynomials

$$\overline{\sigma} = \{\delta^{n/p-m}\tau_{x,\delta}(P) : P \in \sigma\}.$$

Then **(B1)–(B3)** imply that $(\overline{P}_\alpha)_{\alpha\in\mathcal{A}}$ forms an $(\mathcal{A}, x, \epsilon, 1)$-basis for $\overline{\sigma}$.

Similarly, under the assumption that $(P_\alpha)_{\alpha\in\mathcal{A}}$ forms an $(\mathcal{A}, x, \epsilon, \delta, \Lambda)$-basis for σ, we deduce that $(\overline{P}_\alpha)_{\alpha\in\mathcal{A}}$ forms an $(\mathcal{A}, x, \epsilon, 1, \Lambda)$-basis for $\overline{\sigma}$.

2.7.2. Tagged cubes. Assume that we are given a subset $E \subset \mathbb{R}^n$, a set of multiindices $\mathcal{A} \subset \mathcal{M}$, and a cube $Q \subset \mathbb{R}^n$.

We say that Q is *tagged with* $(\mathcal{A}, \epsilon)$ provided that $\#(E \cap Q) \leq 1$ or there exists $\mathcal{A}' \leq \mathcal{A}$ such that

$$\sigma(Q) \text{ has an } (\mathcal{A}', x_Q, \epsilon, \delta_Q)\text{-basis} \quad (\text{recall that } x_Q = \text{ center of } Q).$$

Remark 20. Note that every cube is tagged with $(\mathcal{A}, \epsilon)$ with $\mathcal{A} = \emptyset$.

Let real numbers $\eta \in (0, 1)$, $\Lambda \geq 1$, and $\delta > 0$ be given. Let $\mathcal{A}$ be a set of multiindices of order $\leq m - 1$, and let $M = (M_{\alpha\beta})_{\alpha,\beta\in\mathcal{A}}$ be a matrix (with real entries).

We say that M is (η, Λ, δ)-*near triangular* provided that

$$|M_{\alpha\beta} - \delta_{\alpha\beta}| \leq \begin{cases} \eta \cdot \delta^{|\alpha|-|\beta|} & \text{if } \alpha, \beta \in \mathcal{A},\ \beta \geq \alpha, \\ \Lambda \cdot \delta^{|\alpha|-|\beta|} & \text{if } \alpha, \beta \in \mathcal{A}. \end{cases}$$

Lemma 21. *If the matrices* $M = (M_{\alpha\beta})_{\alpha,\beta\in\mathcal{A}}, \widetilde{M} = (\widetilde{M}_{\alpha\beta})_{\alpha,\beta\in\mathcal{A}}$ *are* (η, Λ, δ)-*near triangular and* $(\widetilde{\eta}, \widetilde{\Lambda}, \delta)$-*near triangular, respectively, then* $M\widetilde{M}$ *is* $(\check{\eta}, \check{\Lambda}, \delta)$-*near triangular if* $\check{\eta} < 1$*, where* $\check{\eta} = C \cdot (\eta\widetilde{\Lambda} + \widetilde{\eta}\Lambda)$ *and* $\check{\Lambda} = C\Lambda\widetilde{\Lambda}$ *for a universal constant* C*.*

Proof. Suppose that $\alpha, \beta \in \mathcal{A}$ and $\beta > \alpha$. Then

$$(M\widetilde{M})_{\alpha\beta} = \sum_{\gamma\in\mathcal{A}} M_{\alpha\gamma}\widetilde{M}_{\gamma\beta}.$$

If $\gamma \in \mathcal{A}$ and $\gamma > \alpha$, then the corresponding term in the above sum is bounded in magnitude by $\eta\widetilde{\Lambda}\delta^{|\alpha|-|\gamma|}\delta^{|\gamma|-|\beta|} = \eta\widetilde{\Lambda}\delta^{|\alpha|-|\beta|}$. Alternatively, if $\gamma \in \mathcal{A}$ and $\gamma < \beta$ then the relevant term is bounded in magnitude by $\widetilde{\eta}\Lambda\delta^{|\alpha|-|\beta|}$. The total number of terms is at most $D = \dim(\mathcal{P})$, hence we see that $|(M\widetilde{M})_{\alpha\beta}| \leq \check{\eta}\delta^{|\alpha|-|\beta|}$ with $\check{\eta}$ as in the statement of the lemma.

Next, observe that

$$(M\widetilde{M})_{\alpha\alpha} = \sum_{\gamma\in\mathcal{A}} M_{\alpha\gamma}\widetilde{M}_{\gamma\alpha}.$$

If $\gamma \in \mathcal{A}$ and either $\gamma > \alpha$ or $\gamma < \alpha$, then the relevant term in the above sum is bounded in magnitude either by $\eta\widetilde{\Lambda}\delta^{|\alpha|-|\gamma|}\delta^{|\gamma|-|\alpha|} = \eta\widetilde{\Lambda}$ or by $\widetilde{\eta}\Lambda\delta^{|\alpha|-|\gamma|}\delta^{|\gamma|-|\alpha|} = \widetilde{\eta}\Lambda$, respectively. If $\gamma = \alpha$, then the relevant term in the sum is equal to

$$M_{\alpha\alpha}\widetilde{M}_{\alpha\alpha} = (1+\mathcal{O}(\eta))(1+\mathcal{O}(\widetilde{\eta})) = 1+\mathcal{O}(\eta+\widetilde{\eta}).$$

Hence, we find that $|(M\widetilde{M})_{\alpha\alpha} - 1|| \leq \check{\eta}$.

Finally, we assume that $\alpha, \beta \in \mathcal{A}$ and $\beta < \alpha$. Then the estimates $|M_{\alpha\gamma}| \leq \Lambda\delta^{|\alpha|-|\gamma|}$ and $|\widetilde{M}_{\gamma\beta}| \leq \widetilde{\Lambda}\delta^{|\gamma|-|\beta|}$ imply that $|(M\widetilde{M})_{\alpha\beta}| \leq \check{\Lambda}\delta^{|\alpha|-|\beta|}$ with $\check{\Lambda}$ as in the statement of the lemma.

This concludes the proof of Lemma 21. □

Lemma 22. *Assume that $\eta\Lambda^D$ is less than a small enough constant depending on m and n. Then the following holds.*

- *If the matrix $M = (M_{\alpha\beta})_{\alpha,\beta\in\mathcal{A}}$ is (η, Λ, δ)-near triangular, then M is invertible and the inverse matrix M^{-1} is $(C\eta\Lambda^D, C\Lambda^D, \delta)$-near triangular.*

Here, $D = \dim(\mathcal{P})$, and C depends only on m and n.

Proof. Let $Y = (\delta_{ij} + X_{ij})_{i,j=1,\dots,K}$ be a $K \times K$ matrix, where the X_{ij} are variables. Let $(Y^{-1})_{ab}$ be the entries of Y^{-1} $(a, b = 1, \dots, K)$. Cramer's rule gives

$$\det Y = P(X) \quad \text{and} \quad (\det Y) \cdot \left[(Y^{-1})_{ab} - \delta_{ab}\right] = P_{ab}(X),$$

where $P(X), P_{ab}(X)$ are K-th degree polynomials in $X = (X_{ij})_{i,j=1,\dots,K}$. In P, P_{ab}, we separate the monomials containing only the variables X_{ij} with $i < j$ from the monomials containing at least one variable X_{ij} with $i \geq j$. We write $P = P_0 + P_1$ and $P_{ab} = P_{ab,0} + P_{ab,1}$, where the monomials in $P_0, P_{ab,0}$ contain only X_{ij} with $i < j$, and the monomials in $P_1, P_{ab,1}$ contain at least one X_{ij} with $i \geq j$.

Suppose that $X_{ij} = 0$ for $i \geq j$. Then Y is upper triangular with 1's on the main diagonal, hence the same is true of Y^{-1}. It follows that $P_0 \equiv 1$, and $P_{ab,0} \equiv 0$ for $a \geq b$. Now we drop the assumption that $X_{ij} = 0$ for $i \geq j$, and assume instead that $|X_{ij}| \leq \eta$ for $i \geq j$ and $|X_{ij}| \leq \Lambda$ for all i, j. (Here, $0 < \eta < 1 \leq \Lambda$.)

We write C, C', C'', etc. to denote constants depending only on K. By examining each monomial separately, we see that

$$|P_1(X)|, |P_{ab,1}(X)| \leq C\eta\Lambda^{K-1}, \quad \text{and} \quad |P_{ab,0}(X)| \leq C\Lambda^K.$$

Combining these estimates with our knowledge of P_0 and $P_{ab,0}$ $(a \geq b)$, we conclude that

$$|\det Y - 1| \leq C\eta\Lambda^{K-1}, \quad \text{and} \quad |(\det Y)\cdot((Y^{-1})_{ab} - \delta_{ab})| \leq \begin{cases} C\eta\Lambda^{K-1} & \text{if } a \geq b, \\ C\Lambda^K & \text{all } a, b. \end{cases}$$

This immediately implies the following result:

(*) Let $Y = (Y_{ij})$ be a $K \times K$ matrix, satisfying

$$|Y_{ij} - \delta_{ij}| \leq \eta \ \text{ for } i \geq j \quad \text{and} \quad |Y_{ij}| \leq \Lambda \text{ (all } i, j),$$

where $\eta\Lambda^{K-1}$ is less than a small enough constant depending only on K.

Then the inverse matrix $Y^{-1} = ((Y^{-1})_{ab})$ satisfies

$$|(Y^{-1})_{ab} - \delta_{ab}| \le C\eta\Lambda^{K-1} \text{ for } a \ge b \quad \text{and} \quad |(Y^{-1})_{ab}| \le C\Lambda^K \text{ (all } a, b).$$

Let $M = (M_{\alpha\beta})_{\alpha,\beta\in\mathcal{A}}$ be an (η, Λ, δ)-near triangular matrix. Let $\alpha_1 < \alpha_2 < \cdots < \alpha_K$ be the elements of $\mathcal{A}$. Applying **(*)** to the matrix $Y_{ij} = \delta^{|\alpha_i|-|\alpha_j|} M_{\alpha_i\alpha_j}$, we find that M^{-1} is $(C\eta\Lambda^{K-1}, C\Lambda^K, \delta)$-near triangular, as claimed in Lemma 22. □

Lemma 23. *Let* $x \in \mathbb{R}^n$. *Suppose* $\sigma_2 \subset C \cdot [\sigma_1 + \mathcal{B}(x, \delta)]$, *and suppose* σ_2 *has an* $(\mathcal{A}, x, \epsilon, \delta, \Lambda)$*-basis. Then* σ_1 *has an* $(\mathcal{A}, x, C\epsilon\Lambda, \delta, C\Lambda)$*-basis. Here,* C *depends only on* m, n, *and* p.

Proof. By rescaling, we may assume without loss of generality that $\delta = 1$. (See Remark 19.)

Let $(\widetilde{P}_\alpha)_{\alpha\in\mathcal{A}}$ be an $(\mathcal{A}, x, \epsilon, 1, \Lambda)$-basis for σ_2. Then

- $\widetilde{P}_\alpha \in \epsilon\sigma_2 \subset C\epsilon\,[\sigma_1 + \mathcal{B}(x, 1)] \quad (\alpha \in \mathcal{A})$;
- $\partial^\beta \widetilde{P}_\alpha(x) = \delta_{\beta\alpha} \quad (\beta, \alpha \in \mathcal{A})$;
- $|\partial^\beta \widetilde{P}_\alpha(x)| \le \epsilon \quad (\alpha \in \mathcal{A}, \beta \in \mathcal{M}, \beta > \alpha)$;
- $|\partial^\beta \widetilde{P}_\alpha(x)| \le \Lambda \quad (\alpha \in \mathcal{A}, \beta \in \mathcal{M})$.

The first bullet point above gives $\widetilde{P}_\alpha = P_\alpha + (\widetilde{P}_\alpha - P_\alpha)$ with $P_\alpha \in C\epsilon\sigma_1$ (all $\alpha \in \mathcal{A}$) and $|\partial^\beta(\widetilde{P}_\alpha - P_\alpha)(x)| \le C\epsilon$ (all $\alpha \in \mathcal{A}, \beta \in \mathcal{M}$).

The four bullet point properties of the $\widetilde{P}_\alpha$ now yield the following properties of the P_α.

- $P_\alpha \in C\epsilon\sigma_1 \quad (\alpha \in \mathcal{A})$;
- $|\partial^\beta P_\alpha(x) - \delta_{\beta\alpha}| \le C\epsilon \quad (\beta, \alpha \in \mathcal{A})$;
- $|\partial^\beta P_\alpha(x)| \le C\epsilon \quad (\alpha \in \mathcal{A}, \beta \in \mathcal{M}, \beta > \alpha)$;
- $|\partial^\beta P_\alpha(x)| \le C\Lambda \quad (\alpha \in \mathcal{A}, \beta \in \mathcal{M})$.

Inverting the matrix $(\partial^\beta P_\alpha(x))_{\beta,\alpha\in\mathcal{A}}$, we obtain a matrix $(M_{\alpha\gamma})_{\alpha,\gamma\in\mathcal{A}}$ such that

$$\sum_{\alpha\in\mathcal{A}} \partial^\beta P_\alpha(x) \cdot M_{\alpha\gamma} = \delta_{\beta\gamma} \quad (\beta, \gamma \in \mathcal{A}) \quad \text{and} \quad |M_{\alpha\gamma} - \delta_{\alpha\gamma}| \le C\epsilon \quad (\alpha, \gamma \in \mathcal{A}).$$

Set $P^\#_\gamma = \sum_{\alpha\in\mathcal{A}} P_\alpha M_{\alpha\gamma}$ for $\gamma \in \mathcal{A}$. Then

- $P^\#_\gamma \in C\epsilon\sigma_1 \quad (\gamma \in \mathcal{A})$;
- $\partial^\beta P^\#_\gamma(x) = \delta_{\beta\gamma} \quad (\beta, \gamma \in \mathcal{A})$;
- $|\partial^\beta P^\#_\gamma(x)| \le C\Lambda \quad (\gamma \in \mathcal{A}, \beta \in \mathcal{M})$.

For $\beta > \gamma$, we have

$$\partial^\beta P^\#_\gamma(x) = \sum_{\alpha \leq \gamma} \partial^\beta P_\alpha(x) M_{\alpha\gamma} + \sum_{\alpha > \gamma} \partial^\beta P_\alpha(x) M_{\alpha\gamma}.$$

For the sum over $\alpha \leq \gamma$, we note that $\beta > \gamma \geq \alpha$, hence

$$|\partial^\beta P_\alpha(x)| \leq C\epsilon, \quad \text{whereas} \quad |M_{\alpha\gamma}| \leq C.$$

For the sum over $\alpha > \gamma$, we note that

$$|\partial^\beta P_\alpha(x)| \leq \Lambda \quad \text{and} \quad |M_{\alpha\gamma}| \leq C\epsilon.$$

Therefore,

- $|\partial^\beta P^\#_\gamma(x)| \leq C\epsilon\Lambda \quad (\gamma \in \mathcal{A}, \beta \in \mathcal{M}, \beta > \gamma)$.

Thus, the $(P^\#_\gamma)_{\gamma\in\mathcal{A}}$ form an $(\mathcal{A}, x, C\epsilon\Lambda, 1, C\Lambda)$-basis for σ_1, which is what we asserted, since $\delta = 1$. □

Lemma 24. *Let $x \in \mathbb{R}^n$, $\epsilon > 0$, and $1 \leq Z \leq \epsilon^{-1/2}$ be given. Suppose that Z exceeds a large enough universal constant. Let $(P_\alpha)_{\alpha\in\mathcal{A}}$ be an $(\mathcal{A}, x, \epsilon, \delta)$-basis for σ, with*

$$\max\left\{|\partial^\beta P_\alpha(x)|\delta^{|\beta|-|\alpha|} : \alpha \in \mathcal{A},\ \beta \in \mathcal{M}\right\} \geq Z. \tag{2.17}$$

Then σ has an $(\mathcal{A}', x, Z^{-\kappa}, \delta)$-basis, with $\mathcal{A}' < \mathcal{A}$. Here, $\kappa > 0$ is a universal constant.

Proof. By rescaling, we may assume without loss of generality that $\delta = 1$. (See Remark 19.)

Our hypothesis tells us that $(P_\alpha)_{\alpha\in\mathcal{A}}$ is an $(\mathcal{A}, x, \epsilon, 1)$-basis for σ, meaning that

$$P_\alpha \in \epsilon \cdot \sigma; \tag{2.18}$$

$$\partial^\beta P_\alpha(x) = \delta_{\alpha\beta} \quad (\alpha, \beta \in \mathcal{A}); \text{ and} \tag{2.19}$$

$$|\partial^\beta P_\alpha(x)| \leq \epsilon \quad (\alpha \in \mathcal{A},\ \beta \in \mathcal{M},\ \beta > \alpha). \tag{2.20}$$

Pick the minimal multiindex $\overline{\alpha} \in \mathcal{A}$ with $\max_{\beta\in\mathcal{M}} |\partial^\beta P_{\overline{\alpha}}(x)| \geq Z$. (See (2.17).) Thus,

$$|\partial^\beta P_\alpha(x)| < Z, \quad \text{for all } \beta \in \mathcal{M},\ \alpha \in \mathcal{A},\ \alpha < \overline{\alpha}, \tag{2.21}$$

and there exists $\beta_0 \in \mathcal{M}$ such that

$$|\partial^{\beta_0} P_{\overline{\alpha}}(x)| = \max_{\beta\in\mathcal{M}} |\partial^\beta P_{\overline{\alpha}}(x)| \geq Z. \tag{2.22}$$

Note that $\beta_0 \neq \overline{\alpha}$ by (2.19), and $\beta_0 \leq \overline{\alpha}$ by (2.20). Thus, $\beta_0 < \overline{\alpha}$.

Let the elements of $\mathcal{M}$ between β_0 and $\overline{\alpha}$ be ordered as follows:

$$\beta_0 < \beta_1 < \cdots < \beta_k = \overline{\alpha}.$$

Note that $k + 1 \leq \#\mathcal{M} = D$.

Pick $\overline{k} \in \{0, \dots, k\}$ such that

$$|\partial^{\beta_{\overline{k}}} P_{\overline{\alpha}}(x)| Z^{\overline{k}/(D+1)} \ge |\partial^{\beta_\ell} P_{\overline{\alpha}}(x)| Z^{\ell/(D+1)} \quad \text{for all } \ell \in \{0, \dots, k\}.$$

In particular, setting $\overline{\beta} = \beta_{\overline{k}}$, we have

$$|\partial^{\overline{\beta}} P_{\overline{\alpha}}(x)| \ge Z^{-D/(D+1)} |\partial^{\beta_0} P_{\overline{\alpha}}(x)| \overset{(2.22)}{\ge} Z^{1/(D+1)}, \text{ and} \tag{2.23}$$

$$|\partial^{\overline{\beta}} P_{\overline{\alpha}}(x)| \ge Z^{1/(D+1)} |\partial^{\beta_l} P_{\overline{\alpha}}(x)| \quad \text{for } \ell = \overline{k}+1, \dots, k. \tag{2.24}$$

If $\beta \in \mathcal{M}$, $\beta > \overline{\alpha}$, then (2.20) and (2.23) give

$$|\partial^{\beta} P_{\overline{\alpha}}(x)| \le \epsilon \le 1 \le Z^{-1/(D+1)} |\partial^{\overline{\beta}} P_{\overline{\alpha}}(x)|.$$

Meanwhile, if $\beta \in \mathcal{M}$, $\overline{\beta} < \beta \le \overline{\alpha}$, then (2.24) states that

$$|\partial^{\beta} P_{\overline{\alpha}}(x)| \le Z^{-1/(D+1)} |\partial^{\overline{\beta}} P_{\overline{\alpha}}(x)|.$$

Thus,

$$|\partial^{\beta} P_{\overline{\alpha}}(x)| \le Z^{-1/(D+1)} |\partial^{\overline{\beta}} P_{\overline{\alpha}}(x)| \quad \text{for any } \beta \in \mathcal{M},\ \beta > \overline{\beta}. \tag{2.25}$$

Note that $|\partial^{\overline{\beta}} P_{\overline{\alpha}}(x)| > 1$, thanks to (2.23). Hence, (2.19) and (2.20) show that

$$\overline{\beta} < \overline{\alpha} \text{ and } \overline{\beta} \notin \mathcal{A}. \tag{2.26}$$

Set $P_{\overline{\beta}} = P_{\overline{\alpha}} / \partial^{\overline{\beta}} P_{\overline{\alpha}}(x)$. Then

(2.27) $P_{\overline{\beta}} \in \epsilon \cdot \sigma$, from (2.18) and $|\partial^{\overline{\beta}} P_{\overline{\alpha}}(x)| > 1$;
(2.28) $|\partial^{\beta} P_{\overline{\beta}}(x)| \le Z^{-1/(D+1)}$ $(\beta \in \mathcal{M},\ \beta > \overline{\beta})$, from (2.25);
(2.29) $|\partial^{\beta} P_{\overline{\beta}}(x)| \le Z^{D/(1+D)}$ $(\beta \in \mathcal{M})$, from (2.22) and (2.23); and
(2.30) $\partial^{\overline{\beta}} P_{\overline{\beta}}(x) = 1$.

Now define

$$P^{\#}_{\overline{\beta}} := P_{\overline{\beta}} - \sum_{\gamma \in \mathcal{A}, \gamma < \overline{\beta}} \partial^{\gamma} P_{\overline{\beta}}(x) P_{\gamma}.$$

We derive some estimates on $P^{\#}_{\overline{\beta}}$. From (2.19) we see that

$$\partial^{\alpha} P^{\#}_{\overline{\beta}}(x) = \partial^{\alpha} P_{\overline{\beta}}(x) - \sum_{\gamma \in \mathcal{A}, \gamma < \overline{\beta}} \partial^{\gamma} P_{\overline{\beta}}(x) \delta_{\alpha\gamma} = 0 \quad (\alpha \in \mathcal{A}, \alpha < \overline{\beta}).$$

Thanks to (2.20), (2.29), and (2.30), we have

$$|\partial^{\overline{\beta}} P^{\#}_{\overline{\beta}}(x) - 1| \le \sum_{\gamma \in \mathcal{A}, \gamma < \overline{\beta}} |\partial^{\gamma} P_{\overline{\beta}}(x)| \cdot |\partial^{\overline{\beta}} P_{\gamma}(x)| \le C Z^{D/(D+1)} \epsilon. \tag{2.31}$$

Meanwhile, if $\beta > \overline{\beta}$ and $\gamma < \overline{\beta}$, then $\beta > \gamma$. Hence, by (2.20), (2.28), and (2.29), we have

$$\begin{aligned}|\partial^\beta P^\#_{\overline{\beta}}(x)| &\leq |\partial^\beta P_{\overline{\beta}}(x)| + \sum_{\gamma\in\mathcal{A},\gamma<\overline{\beta}} |\partial^\gamma P_{\overline{\beta}}(x)| \cdot |\partial^\beta P_\gamma(x)| \\ &\leq Z^{-\frac{1}{D+1}} + CZ^{\frac{D}{D+1}}\epsilon \quad (\beta \in \mathcal{M}, \beta > \overline{\beta}).\end{aligned}$$

From (2.18), (2.27), and (2.29), we have

$$P^\#_{\overline{\beta}} \in \epsilon \cdot \sigma + C \cdot Z^{D/(D+1)}\epsilon \cdot \sigma \subseteq \left(CZ^{D/(D+1)}\epsilon\right) \cdot \sigma.$$

For each $\gamma \in \mathcal{A}$, $\gamma < \overline{\beta}$, (2.26) implies that $\gamma < \overline{\alpha}$. Hence, from (2.21) and (2.29) we have

$$|\partial^\beta P^\#_{\overline{\beta}}(y)| \leq C \cdot Z^{(2D+1)/(D+1)} \quad (\beta \in \mathcal{M}).$$

Since $\epsilon \leq Z^{-1}$, if ϵ is sufficiently small then (2.31) implies that $\partial^{\overline{\beta}} P^\#_{\overline{\beta}}(x) \in [1/2, 2]$. Hence, we may define $\widehat{P}_{\overline{\beta}} = P^\#_{\overline{\beta}}/\partial^{\overline{\beta}} P^\#_{\overline{\beta}}(x)$. The estimates written above show that

$$\widehat{P}_{\overline{\beta}} \in \left(C \cdot Z^{D/(D+1)}\epsilon\right) \cdot \sigma; \tag{2.32}$$

$$\partial^\beta \widehat{P}_{\overline{\beta}}(x) = \delta_{\beta\overline{\beta}} \quad (\beta \in \mathcal{A}, \beta < \overline{\beta} \text{ or } \beta = \overline{\beta}); \tag{2.33}$$

$$|\partial^\beta \widehat{P}_{\overline{\beta}}(x)| \leq C \cdot Z^{-1/(D+1)} + C \cdot Z^{D/(D+1)}\epsilon \quad (\beta \in \mathcal{M},\ \beta > \overline{\beta}); \text{ and} \tag{2.34}$$

$$|\partial^\beta \widehat{P}_{\overline{\beta}}(x)| \leq C \cdot Z^{(2D+1)/(D+1)} \quad (\beta \in \mathcal{M}). \tag{2.35}$$

For each $\alpha \in \mathcal{A}$, $\alpha < \overline{\beta}$, set $\widehat{P}_\alpha = P_\alpha - \partial^{\overline{\beta}} P_\alpha(x) \widehat{P}_{\overline{\beta}}$. Note that $|\partial^{\overline{\beta}} P_\alpha(x)| \leq \epsilon \leq 1$, thanks to (2.20). From (2.18) and (2.32), we have

$$\widehat{P}_\alpha \in \left(CZ^{D/(D+1)}\epsilon\right) \cdot \sigma. \tag{2.36}$$

From (2.20) and (2.35), we have

$$\begin{aligned}|\partial^\beta \widehat{P}_\alpha(x)| &\leq |\partial^\beta P_\alpha(x)| + |\partial^{\overline{\beta}} P_\alpha(x)| \cdot |\partial^\beta \widehat{P}_{\overline{\beta}}(x)| \leq \epsilon + \epsilon \cdot CZ^{(2D+1)/(D+1)} \\ &\leq C\epsilon \cdot Z^{(2D+1)/(D+1)} \quad (\beta \in \mathcal{M},\ \beta > \alpha).\end{aligned} \tag{2.37}$$

From (2.19) and (2.33), we have

$$\begin{aligned}\partial^\beta \widehat{P}_\alpha(x) &= \partial^\beta P_\alpha(x) - \partial^{\overline{\beta}} P_\alpha(x) \partial^\beta \widehat{P}_{\overline{\beta}}(x) \\ &= \begin{cases} \delta_{\alpha\beta} - \partial^{\overline{\beta}} P_\alpha(x)\delta_{\beta\overline{\beta}} = \delta_{\alpha\beta} & : \text{if } \beta \in \mathcal{A},\ \beta < \overline{\beta} \\ \partial^{\overline{\beta}} P_\alpha(x) - \partial^{\overline{\beta}} P_\alpha(x)\delta_{\overline{\beta}\,\overline{\beta}} = 0 & : \text{if } \beta = \overline{\beta} \end{cases} \\ &= \delta_{\alpha\beta} \qquad \text{if either } \beta < \overline{\beta} \text{ and } \beta \in \mathcal{A}, \text{ or } \beta = \overline{\beta}.\end{aligned} \tag{2.38}$$

Set $\overline{\mathcal{A}} = \{\alpha \in \mathcal{A} : \alpha < \overline{\beta}\} \cup \{\overline{\beta}\}$. Then (2.26) shows that the minimal element of $\mathcal{A}\Delta\overline{\mathcal{A}}$ is $\overline{\beta}$. Therefore, $\overline{\mathcal{A}} < \mathcal{A}$.

From (2.32)–(2.34) and (2.36)–(2.38) we deduce that

$$(\widehat{P}_\alpha)_{\alpha\in\overline{\mathcal{A}}} \text{ is an } (\overline{\mathcal{A}}, x, C\cdot(Z^{-\frac{1}{D+1}} + Z^{\frac{2D+1}{D+1}}\cdot\epsilon), 1)\text{-basis for } \sigma.$$

Since $\epsilon \leq Z^{-2}$ and $\delta = 1$, this implies the conclusion of Lemma 24. □

Lemma 25. *There exist constants* $\kappa_1, \kappa_2 \in (0,1]$ *depending only on* m, n, *and* p *such that the following holds.*

Let $x \in \mathbb{R}^n$. *Suppose that* σ *has an* $(\mathcal{A}, x, \epsilon, \delta)$*-basis.*

Then there exists a multiindex set $\mathcal{A}' \leq \mathcal{A}$, *and there exist numbers* $\kappa' \in [\kappa_1, \kappa_2]$ *and* $\Lambda \geq 1$ *with* $\epsilon^{\kappa'}\Lambda^{100D} \leq \epsilon^{\kappa'/2}$, *such that* σ *has an* $(\mathcal{A}', x, \epsilon^{\kappa'}, \delta, \Lambda)$*-basis.*

Here, $D = \dim \mathcal{P}$.

Proof. By rescaling, we may assume without loss of generality that $\delta = 1$. (See Remark 19.)

Let $\mathcal{A}_0 = \mathcal{A}$ and $L = 2^D$, and let $\kappa \in (0,1)$ be as in Lemma 24. Set

$$\epsilon_0 = \epsilon,\ \ \epsilon_\ell = \epsilon^{\kappa^\ell/(200D)^\ell} \quad \text{and} \quad Z_\ell = \epsilon^{-\kappa^{\ell-1}/(200D)^\ell} \quad \text{for } \ell = 1,\dots,L.$$

Note that $(Z_\ell)^{-\kappa} = \epsilon_\ell$ and $Z_\ell \leq (\epsilon_{\ell-1})^{-1/2}$ for each $\ell \geq 1$.

Let $(P_\alpha^{(0)})_{\alpha\in\mathcal{A}}$ be an $(\mathcal{A}_0, x, \epsilon_0, 1)$-basis for σ. We carry out the following iterative procedure:

Stage 0. From Lemma 24, we have either

Case A: $|\partial^\beta P_\alpha^{(0)}(x)| \leq Z_1 \quad$ for all $\alpha \in \mathcal{A}_0, \beta \in \mathcal{M}$,

or

Case B: There exist polynomials $(P_\alpha^{(1)})_{\alpha\in\mathcal{A}_1}$, such that $(P_\alpha^{(1)})_{\alpha\in\mathcal{A}_1}$ is an $(\mathcal{A}_1, x, \epsilon_1, 1)$-basis for σ, for some $\mathcal{A}_1 < \mathcal{A}_0$.

In Case A we terminate. In Case B, we pass to:

Stage 1. From Lemma 24, we have either

Case A: $|\partial^\beta P_\alpha^{(1)}(x)| \leq Z_2 \quad$ for all $\alpha \in \mathcal{A}_1, \beta \in \mathcal{M}$,

or

Case B: There exist polynomials $(P_\alpha^{(2)})_{\alpha\in\mathcal{A}_2}$, such that $(P_\alpha^{(2)})_{\alpha\in\mathcal{A}_2}$ is an $(\mathcal{A}_2, x, \epsilon_2, 1)$-basis for σ, for some $\mathcal{A}_2 < \mathcal{A}_1$.

In Case A we terminate. In Case B, we pass to Stage 2, and so forth.

Since $\mathcal{A}_0 > \mathcal{A}_1 > \mathcal{A}_2 > \cdots$ and $\#\{\mathcal{A} : \mathcal{A} \subseteq \mathcal{M}\} = L$, there exists $\ell \in \{0,\dots,L-1\}$ such that Case A occurs in Stage ℓ. Thus,

$$(P_\alpha^{(\ell)})_{\alpha\in\mathcal{A}_\ell} \text{ is an } (\mathcal{A}_\ell, x, \epsilon_\ell, 1)\text{-basis for } \sigma, \text{ with}$$
$$|\partial^\beta P_\alpha^{(\ell)}(x)| \leq Z_{\ell+1} \quad \text{for all } \alpha \in \mathcal{A}_\ell, \beta \in \mathcal{M}.$$

Note that

$$\epsilon_\ell \cdot Z_{\ell+1}^{100D} = \epsilon^{\frac{\kappa^\ell}{(200D)^\ell} - \frac{\kappa^\ell}{(200D)^{\ell+1}}100D} = \epsilon^{\frac{\kappa^\ell}{2(200D)^\ell}} = \sqrt{\epsilon_\ell}.$$

Note that $\epsilon_\ell = \epsilon^{\kappa'}$ for $\kappa' = \kappa^\ell/(200D)^\ell$. We set $\Lambda = Z_{\ell+1}$. Then the above conditions imply the conclusion of Lemma 25, since $\delta = 1$. □

Lemma 26. *Let* $x, y \in \mathbb{R}^n$*; assume that* $|x-y| \leq C\delta$*. Suppose* σ *has an* $(\mathcal{A}, x, \epsilon, \delta, \Lambda)$*-basis. Assume that* $\epsilon\Lambda^D$ *is less than a small enough constant depending on* m*,* n*, and* p*.*

Then σ *has an* $(\mathcal{A}, y, C\epsilon\Lambda^{2D+1}, \delta, C\Lambda^{2D+1})$*-basis.*

Proof. By rescaling, we may assume that $\delta = 1$. (See Remark 19.) Let $(P_\alpha)_{\alpha\in\mathcal{A}}$ be an $(\mathcal{A}, x, \epsilon, \delta, \Lambda)$-basis for σ. Thus,

- $P_\alpha \in \epsilon\sigma \quad (\alpha \in \mathcal{A})$;
- $\partial^\beta P_\alpha(x) = \delta_{\beta\alpha} \quad (\beta, \alpha \in \mathcal{A})$;
- $|\partial^\beta P_\alpha(x)| \leq \epsilon \quad (\alpha \in \mathcal{A}, \beta \in \mathcal{M}, \beta > \alpha)$;
- $|\partial^\beta P_\alpha(x)| \leq \Lambda \quad (\alpha \in \mathcal{A}, \beta \in \mathcal{M})$.

For $\beta \in \mathcal{M}$ and $\alpha \in \mathcal{A}$ with $\beta > \alpha$, we have

$$|\partial^\beta P_\alpha(y)| = \left| \sum_\gamma \frac{1}{\gamma!} \partial^{\beta+\gamma} P_\alpha(x) \cdot (y-x)^\gamma \right| \leq C\epsilon.$$

Also,

$$\partial^\alpha P_\alpha(y) = \partial^\alpha P_\alpha(x) + \sum_{\gamma\neq 0} \frac{1}{\gamma!} \partial^{\alpha+\gamma} P_\alpha(x) \cdot (y-x)^\gamma = 1 + \text{Error}, \quad \text{where } |\text{Error}| \leq C\epsilon.$$

On the other hand, for general $\beta \in \mathcal{M}$ and $\alpha \in \mathcal{A}$, we have

$$|\partial^\beta P_\alpha(y)| = \left| \sum_\gamma \frac{1}{\gamma!} \partial^{\beta+\gamma} P_\alpha(x) \cdot (y-x)^\gamma \right| \leq C\Lambda.$$

Thus, $(\partial^\beta P_\alpha(y))_{\beta,\alpha\in\mathcal{A}}$ is a $(C\epsilon, C\Lambda, 1)$-near triangular matrix. Therefore,

$$\text{the inverse } (M_{\alpha\gamma})_{\alpha,\gamma\in\mathcal{A}} \text{ of } (\partial^\beta P_\alpha(y))_{\beta,\alpha\in\mathcal{A}} \text{ is } (C\epsilon\Lambda^{2D}, \Lambda^{2D})\text{-near triangular.} \tag{2.39}$$

For each $\gamma \in \mathcal{A}$, we define $P^\#_\gamma = \sum_{\alpha\in\mathcal{A}} P_\alpha \cdot M_{\alpha\gamma}$. From the properties of the P_α, we read off the following.

- $|\partial^\beta P^\#_\gamma(y)| \leq C\Lambda^{2D+1} \quad (\beta \in \mathcal{M}, \gamma \in \mathcal{A})$;
- $\partial^\beta P^\#_\gamma(y) = \delta_{\beta\gamma} \quad (\beta, \gamma \in \mathcal{A})$;
- $P^\#_\gamma \in C\epsilon\Lambda^{2D+1}\sigma \quad (\gamma \in \mathcal{A})$.

Finally, for each $\beta \in \mathcal{M}$ and $\gamma \in \mathcal{A}$ with $\beta > \gamma$, we have

$$\begin{aligned} |\partial^\beta P^\#_\gamma(y)| &\leq \sum_{\alpha\leq\gamma} |\partial^\beta P_\alpha(y)| \cdot |M_{\alpha\gamma}| + \sum_{\alpha>\gamma} |\partial^\beta P_\alpha(y)| \cdot |M_{\alpha\gamma}| \\ &\leq \sum_{\alpha\leq\gamma} C\epsilon \cdot C\Lambda^{2D+1} + \sum_{\alpha>\gamma} C\Lambda \cdot C\epsilon\Lambda^{2D} \quad \text{(see (2.39))} \\ &\leq C\epsilon\Lambda^{2D+1}. \end{aligned}$$

Thus, $(P^\#_\gamma)_{\gamma\in\mathcal{A}}$ is an $(\mathcal{A}, y, C\epsilon\Lambda^{2D+1}, 1, C\Lambda^{2D+1})$-basis for σ. □

Lemma 27. *There exists $\kappa > 0$ depending only on m, n, and p, such that the following holds. Let $x, y \in \mathbb{R}^n$. Suppose that σ has an $(\mathcal{A}, x, \epsilon, \delta)$-basis and that $|x - y| \leq C\delta$. Then, there exists $\mathcal{A}' \leq \mathcal{A}$ such that σ has an $(\mathcal{A}', y, \epsilon^\kappa, \delta)$-basis.*

Proof. By Lemma 25, there exist $\kappa' \in [\kappa_1, \kappa_2]$, $\mathcal{A}' \leq \mathcal{A}$, and $\Lambda \geq 1$, such that

$$\sigma \text{ has an } (\mathcal{A}', x, \epsilon^{\kappa'}, \delta, \Lambda)\text{-basis, and } \epsilon^{\kappa'} \Lambda^{100D} \leq \epsilon^{\kappa'/2}.$$

Here, $\kappa_1, \kappa_2 > 0$ are universal constants.

Thus, σ has an $(\mathcal{A}', y, C\epsilon^{\kappa'}\Lambda^{2D+1}, \delta, C\Lambda^{2D+1})$-basis, due to Lemma 26.

Note that $C\epsilon^{\kappa'}\Lambda^{2D+1} \leq C\epsilon^{\kappa'/2} \leq \epsilon^{\kappa_1/4}$, if ϵ is less than a small enough universal constant. Hence, σ has an $(\mathcal{A}', y, \epsilon^{\kappa_1/4}, \delta)$-basis. This completes the proof of Lemma 27. □

Lemma 28. *Suppose that $Q' \subset Q$ and Q is tagged with $(\mathcal{A}, \epsilon)$. Then Q' is tagged with $(\mathcal{A}, \epsilon^\kappa)$, where $\kappa > 0$ depends only on m, n, and p.*

Proof. Let $Q' \subset Q$, and suppose Q is tagged with $(\mathcal{A}, \epsilon)$. Then either $\#(Q \cap E) \leq 1$ or $\sigma(Q)$ has an $(\mathcal{A}', x_Q, \epsilon, \delta_Q)$-basis for some $\mathcal{A}' \leq \mathcal{A}$. (Recall that x_Q is the center of Q.)

If $\#(Q \cap E) \leq 1$ then $\#(Q' \cap E) \leq 1$, hence Q' is tagged with $(\mathcal{A}, \epsilon^\kappa)$ for any $\kappa > 0$, which implies the conclusion of Lemma 28.

Suppose instead that

$$\sigma(Q) \text{ has an } (\mathcal{A}', x_Q, \epsilon, \delta_Q)\text{-basis with } \mathcal{A}' \leq \mathcal{A}.$$

Then Lemma 25 implies that there exist $\kappa' \in [\kappa_1, \kappa_2]$, $\Lambda \geq 1$, and $\mathcal{A}'' \leq \mathcal{A}'$, such that

$$\sigma(Q) \text{ has an } (\mathcal{A}'', x_Q, \epsilon^{\kappa'}, \delta_Q, \Lambda)\text{-basis,}$$

with $\epsilon^{\kappa'} \cdot \Lambda^{100D} \leq \epsilon^{\kappa'/2}$. Here, $\kappa_1, \kappa_2 > 0$ are universal constants.

We have $|x_{Q'} - x_Q| \leq \delta_Q$, since $x_{Q'} \in Q' \subset Q$ and $x_Q \in Q$. Hence, Lemma 26 implies that

$$\sigma(Q) \text{ has an } (\mathcal{A}'', x_{Q'}, C\epsilon^{\kappa'}\Lambda^{2D+1}, \delta_Q, C\Lambda^{2D+1})\text{-basis.}$$

Since Lemma 15 gives $\sigma(Q) \subset C\,[\sigma(Q') + \mathcal{B}(x_{Q'}, \delta_Q)]$, Lemma 23 implies that

$$\sigma(Q') \text{ has an } (\mathcal{A}'', x_{Q'}, C\epsilon^{\kappa'}\Lambda^{10D}, \delta_Q, C\Lambda^{10D})\text{-basis.}$$

Since $\delta_{Q'} \leq \delta_Q$ and $C\epsilon^{\kappa'}\Lambda^{100D} \leq C\epsilon^{\kappa'/2} \leq \epsilon^{\kappa'/4} \leq \epsilon^{\kappa_1/4}$, we see that

$$\sigma(Q') \text{ has an } (\mathcal{A}'', x_{Q'}, \epsilon^{\kappa_1/4}, \delta_{Q'})\text{-basis.}$$

Recall that $\mathcal{A}'' \leq \mathcal{A}' \leq \mathcal{A}$. This completes the proof of Lemma 28. □

2.7.3. Computing a basis. We fix $x \in \mathbb{R}^n$ and $\mathcal{A} \subset \mathcal{M}$.

Recall that $\mathcal{P}$ is the vector space of polynomials on $\mathbb{R}^n$ of degree at most $m-1$, and $D = \dim \mathcal{P}$. We identify $\mathcal{P}$ with $\mathbb{R}^D$, by identifying $P \in \mathcal{P}$ with $(\partial^\alpha P(x))_{\beta \in \mathcal{M}}$. We define

$$|P|_x = \sum_\beta |\partial^\beta P(x)|.$$

Suppose we are given $\Lambda \geq 1$. In this subsection, we write $c(\Lambda)$, $C(\Lambda)$, etc. to denote constants determined by m, n, p, and Λ. We write c, C, etc. to denote constants determined by m, n, and p.

Let q be a nonnegative quadratic form on $\mathcal{P}$; thus, $q(P) \geq 0$ for all $P \in \mathcal{P}$. We are given a symmetric D x D matrix $(q_{\beta\gamma})_{\beta,\gamma\in\mathcal{M}}$, with

$$q(P) = \sum_{\beta,\gamma\in\mathcal{M}} q_{\beta\gamma} \cdot \partial^\beta P(x) \cdot \partial^\gamma P(x) \quad \text{for } P \in \mathcal{P}. \tag{2.40}$$

Let $\overline{\overline{\sigma}} \subset \mathcal{P}$ be a symmetric convex set with

$$\left\{P \in \mathcal{P} : q(P) \leq \Lambda^{-1}\right\} \subset \overline{\overline{\sigma}} \subset \{P \in \mathcal{P} : q(P) \leq \Lambda\}. \tag{2.41}$$

Given $x \in \mathbb{R}^n$, $(q_{\beta\gamma})_{\beta,\gamma\in\mathcal{M}}$, $\delta \in (0,\infty)$, and $\mathcal{A} \subset \mathcal{M}$, we want to compute (approximately) the least η for which there exists a collection $(P_\alpha)_{\alpha\in\mathcal{A}}$ of $(m-1)$-st degree polynomials such that

$$P_\alpha \in \eta^{1/2}\delta^{|\alpha|+n/p-m} \cdot \overline{\overline{\sigma}} \quad (\alpha \in \mathcal{A}), \tag{2.42}$$

$$\partial^\beta P_\alpha(x) = \delta_{\beta\alpha} \quad (\beta, \alpha \in \mathcal{A}), \tag{2.43}$$

$$|\partial^\beta P_\alpha(x)| \leq \eta^{1/2}\delta^{|\alpha|-|\beta|} \quad (\alpha \in \mathcal{A},\ \beta \in \mathcal{M},\ \beta > \alpha). \tag{2.44}$$

To compute such an η, we introduce the quadratic form

$$\begin{aligned} M^\delta((P_\alpha)_{\alpha\in\mathcal{A}}) &:= \sum_{\alpha\in\mathcal{A}} q(\delta^{m-n/p-|\alpha|}P_\alpha) + \sum_{\substack{\alpha\in\mathcal{A},\beta\in\mathcal{M}\\ \beta>\alpha}} (\delta^{|\beta|-|\alpha|}\partial^\beta P_\alpha(x))^2 \\ &= \sum_{\alpha\in\mathcal{A}} \sum_{\beta,\gamma\in\mathcal{M}} \delta^{2(m-n/p-|\alpha|)} q_{\beta\gamma} \cdot \partial^\beta P_\alpha(x) \cdot \partial^\gamma P_\alpha(x) \\ &\qquad + \sum_{\substack{\alpha\in\mathcal{A},\beta\in\mathcal{M}\\ \beta>\alpha}} (\delta^{|\beta|-|\alpha|}\partial^\beta P_\alpha(x))^2, \end{aligned} \tag{2.45}$$

which is defined on the affine subspace

$$H := \left\{\vec{P} = (P_\alpha)_{\alpha\in\mathcal{A}} : \partial^\beta P_\alpha(x) = \delta_{\beta\alpha} \text{ for } \alpha, \beta \in \mathcal{A}\right\}. \tag{2.46}$$

For fixed q, $\mathcal{A}$, x, we denote

$$\eta_{\min}(\delta) = \min_{\vec{P}\in H} M^\delta(\vec{P}),$$

which we regard as a function of $\delta \in (0,\infty)$.

The definition of $\eta_{\min}(\delta)$ shows that

$$\left\{\begin{array}{l}\text{we can satisfy (2.42), (2.43), (2.44) if } \eta > C(\Lambda)\cdot\eta_{\min}(\delta), \text{ but}\\ \text{we cannot satisfy (2.42), (2.43), (2.44) if } \eta < c(\Lambda)\cdot\eta_{\min}(\delta).\end{array}\right. \tag{2.47}$$

Hence,

(2.48) $\overline{\overline{\sigma}}$ has an $(\mathcal{A}, x, \eta^{1/2}, \delta)$-basis if $\eta > C(\Lambda)\cdot\eta_{\min}(\delta)$, but not if $\eta < c(\Lambda)\cdot\eta_{\min}(\delta)$.

Moreover,

$$\eta_{\min}(\delta_1) \leq \eta_{\min}(\delta_2) \leq \left(\frac{\delta_2}{\delta_1}\right)^{2m}\eta_{\min}(\delta_1) \quad \text{for } \delta_1 \leq \delta_2, \tag{2.49}$$

which follows at once from the definition of $\eta_{\min}$.

We now compute an expression that approximates the function $\eta_{\min}(\delta)$.

We identify the index set $\mathcal{I} = \{(\alpha, \beta) : \alpha \in \mathcal{A}, \beta \in \mathcal{M} \setminus \mathcal{A}\}$ with $\{1, \dots, J\}$ $(J = (\#\mathcal{A})\cdot(\#\mathcal{M} - \#\mathcal{A}))$ by fixing an enumeration of $\mathcal{I}$. We introduce coordinates $w = (w_j)_{1\leq j\leq J} = (w_{\alpha\beta})_{\alpha\in\mathcal{A},\beta\in\mathcal{M}\setminus\mathcal{A}} \in \mathbb{R}^J$ on the space H. We denote

$$P^w_\alpha(z) := \frac{1}{\alpha!}(z-x)^\alpha + \sum_{\beta\in\mathcal{M}\setminus\mathcal{A}} \frac{1}{\beta!} w_{\alpha\beta}(z-x)^\beta \quad \text{for } w \in \mathbb{R}^J. \tag{2.50}$$

We identify

$$\vec{P}^w = (P^w_\alpha)_{\alpha\in\mathcal{A}} \in H \ \text{ with } \ w = (w_j)_{1\leq j\leq J} = (w_{\alpha\beta})_{\alpha\in\mathcal{A},\beta\in\mathcal{M}\setminus\mathcal{A}} \in \mathbb{R}^J. \tag{2.51}$$

We wish to minimize the quadratic function $\widetilde{M}^\delta(w) := M^\delta(\vec{P}^w)$ over $w \in \mathbb{R}^J$. We write

$$\begin{aligned}\widetilde{M}^\delta(w) &= \sum_{i,j=1}^{J} A^\delta_{ij} w_i w_j - 2\sum_{j=1}^{J} b^\delta_j w_j + m^\delta \\ &= \langle A^\delta w, w\rangle - 2\langle b^\delta, w\rangle + m^\delta.\end{aligned} \tag{2.52}$$

Here, we specify a symmetric matrix $A^\delta = (A^\delta_{ij})$, vector $b^\delta = (b^\delta_j)$, and scalar m^δ – all functions of $\delta > 0$. Here, we write $\langle\cdot,\cdot\rangle$ to denote the standard Euclidean inner product on $\mathbb{R}^J$. We express

$$A^\delta_{ij} = \sum_{\mu,\nu} c^{ij}_{\mu\nu}\, \delta^{\mu+\nu/p}, \tag{2.53}$$

$$b^\delta_j = \sum_{\mu,\nu} c^j_{\mu\nu}\, \delta^{\mu+\nu/p}, \tag{2.54}$$

$$m^\delta = \sum_{\mu,\nu} c_{\mu\nu}\, \delta^{\mu+\nu/p}, \tag{2.55}$$

for computable coefficients $c^{ij}_{\mu\nu}$, $c^j_{\mu\nu}$, and $c_{\mu\nu}$; here, the sums on μ and ν are finite, and μ and ν are integers. The coefficient matrix $(c^{ij}_{\mu\nu})$ is symmetric with respect to (i, j). We compute these expressions by writing equation (2.45) in w-coordinates. The quadratic function $\widetilde{M}^\delta$ is nonnegative, hence $A^\delta \geq 0$.

Let $\epsilon > 0$. We eventually send ϵ to zero. We define

$$\widetilde{M}^{\epsilon,\delta}(w) := \langle A^{\epsilon,\delta}w, w\rangle - 2\langle b^{\delta}, w\rangle + m^{\delta}, \quad \text{where } A^{\epsilon,\delta}_{ij} := A^{\delta}_{ij} + \epsilon\delta_{ij}. \tag{2.56}$$

Note that $A^{\epsilon,\delta}$ is invertible because $A^{\delta} \geq 0$ and $\epsilon > 0$. Cramer's rule shows that

$$(A^{\epsilon,\delta})^{-1}_{ij} = \frac{[A^{\epsilon,\delta}]_{ij}}{\det(A^{\epsilon,\delta})} = \frac{\sum_{k,k'} a^{ij}_{kk'}\delta^{\lambda_k}\epsilon^{k'}}{\sum_{\ell,\ell'} b_{\ell\ell'}\delta^{\gamma_\ell}\epsilon^{\ell'}} \tag{2.57}$$

for computable numbers $a^{ij}_{kk'}$, $b_{\ell\ell'}$, λ_k, and γ_ℓ; here, the sums on k,k',ℓ,ℓ' are finite, and k,k',ℓ,ℓ' are nonnegative integers. We write $[A^{\epsilon,\delta}]_{ij}$ to denote the (i,j)-cofactor of the matrix $A^{\epsilon,\delta}$.

The minimum of the quadratic function $\widetilde{M}^{\epsilon,\delta}(w)$ is achieved when

$$\nabla\widetilde{M}^{\epsilon,\delta}(w) = 0,$$

namely, for $w = w^{\epsilon,\delta} := (A^{\epsilon,\delta})^{-1}b^{\delta}$. From (2.56) we see that the minimum value is

$$\begin{aligned}\widetilde{M}^{\epsilon,\delta}(w^{\epsilon,\delta}) &= \left\langle A^{\epsilon,\delta}\left(A^{\epsilon,\delta}\right)^{-1}b^{\delta}, \left(A^{\epsilon,\delta}\right)^{-1}b^{\delta}\right\rangle - 2\left\langle b^{\delta}, \left(A^{\epsilon,\delta}\right)^{-1}b^{\delta}\right\rangle + m^{\delta}\\ &= -\left\langle b^{\delta}, \left(A^{\epsilon,\delta}\right)^{-1}b^{\delta}\right\rangle + m^{\delta}.\end{aligned}$$

Therefore, based on (2.57) and based on the form of the vector b^{δ} and scalar m^{δ} written in (2.54), (2.55), we learn that

$$\min_{w\in\mathbb{R}^J}\widetilde{M}^{\epsilon,\delta}(w) = \frac{\sum_{k,k'} a_{kk'}\delta^{\lambda_k}\epsilon^{k'}}{\sum_{\ell,\ell'} b_{\ell\ell'}\delta^{\gamma_\ell}\epsilon^{\ell'}}. \tag{2.58}$$

for computable numbers $a_{kk'}$, $b_{kk'}$, λ_k, and γ_ℓ. We abuse notation, since the exponents λ_k in (2.58) might differ from the exponents λ_k in (2.57). However, note that the denominator of (2.58) matches the expression in the denominator of (2.57). Also note that all exponents in (2.58) have the form $\mu+\nu/p$ for $\mu,\nu\in\mathbb{Z}$.

The minimum value of $\widetilde{M}^{\epsilon,\delta}(w)$ converges to the minimum value of $\widetilde{M}^{\delta}(w)$ as $\epsilon \to 0^+$. Hence,

$$\eta_{\min}(\delta) = \min_{w\in\mathbb{R}^J}\widetilde{M}^{\delta}(w) = \lim_{\epsilon\to 0+}\frac{\sum_{k,k'} a_{kk'}\delta^{\lambda_k}\epsilon^{k'}}{\sum_{\ell,\ell'} b_{\ell\ell'}\delta^{\gamma_\ell}\epsilon^{\ell'}}.$$

Canceling the smallest powers of ϵ from the numerator and denominator above, we obtain the formula

$$\eta_{\min}(\delta) = \frac{\sum_{k=1}^{K} a_k\delta^{\lambda_k}}{\sum_{\ell=1}^{L} b_\ell\delta^{\gamma_\ell}} \tag{2.59}$$

for nonzero coefficients a_k, b_ℓ. All the coefficients and exponents in (2.59) can be computed using the numbers in (2.58). Both λ_k and γ_ℓ have the form $\mu+\nu/p$ with $\mu,\nu\in\mathbb{Z}$. Here, we abuse notation, since λ_k and γ_ℓ may be different from before. By collecting terms, we may assume that

$$|\lambda_k - \lambda_{k'}| \geq c \quad \text{and} \quad |\gamma_\ell - \gamma_{\ell'}| \geq c \quad \text{for } k \neq k',\ \ell \neq \ell'. \tag{2.60}$$

Here, both K and L are bounded by a universal constant, and $c > 0$ is a universal constant.

We have thus obtained a computable expression for $\eta_{\min}(\delta)$.

We approximate $\eta_{\min}(\delta)$ with a piecewise-monomial function using the following procedure.

PROCEDURE: APPROXIMATE RATIONAL FUNCTION

Given nonzero numbers a_k, b_ℓ and numbers λ_k, γ_ℓ satisfying (2.60), let

$$\eta_{\min}(\delta) = \frac{\sum_{k=1}^{K} a_k \delta^{\lambda_k}}{\sum_{\ell=1}^{L} b_\ell \delta^{\gamma_\ell}}.$$

We assume that K, L are bounded by a universal constant, and $\eta_{\min}(\delta) \geq 0$ for $\delta \in (0, \infty)$. We further assume that $\eta_{\min}(\delta)$ satisfies (2.49).

We compute a collection of pairwise disjoint intervals I_ν with $(0,\infty) = \cup_\nu I_\nu$, and we compute numbers c_ν, λ_ν associated to each I_ν, such that the function

$$\eta_*(\delta) := c_\nu \cdot \delta^{\lambda_\nu} \quad \text{for } \delta \in I_\nu$$

satisfies

$$c \cdot \eta_{\min}(\delta) \leq \eta_*(\delta) \leq C \cdot \eta_{\min}(\delta) \quad \text{for all } \delta \in (0, \infty).$$

The algorithm requires work and storage at most C. In particular, the number of distinct intervals I_ν is at most C.

Explanation. We will analyze separately the numerator and denominator in the rational function $\eta_{\min}(\delta)$. We define

$$\mathcal{B} := \bigcup_{k \neq k'} I_{kk'}\ , \ \text{ with } I_{kk'} := \left\{\delta \in (0,\infty) : 5^{-1} \cdot |a_k \delta^{\lambda_k}| \leq |a_{k'} \delta^{\lambda_{k'}}| \leq 5 \cdot |a_k \delta^{\lambda_k}|\right\},$$

and similarly

$$\mathcal{C} := \bigcup_{\ell \neq \ell'} J_{\ell\ell'}\ , \quad \text{with } J_{\ell\ell'} := \left\{\delta \in (0,\infty) : 5^{-1} \cdot |b_\ell \delta^{\gamma_\ell}| \leq |b_{\ell'} \delta^{\gamma_{\ell'}}| \leq 5 \cdot |b_\ell \delta^{\gamma_\ell}|\right\}.$$

Let $I \subset (0,\infty) \setminus \mathcal{B}$. For $\delta \in I$, all elements in the set $\left\{|a_k \delta^{\lambda_k}| : 1 \leq k \leq K\right\}$ are nonzero and are separated by at least a factor of 5. We choose k such that $|a_k \delta^{\lambda_k}|$ is maximized. By continuity, the same k must work for all $\delta \in I$. By summing a geometric series, we have

$$\sum_{k' \neq k} |a_{k'} \delta^{\lambda_{k'}}| < 2^{-1} \cdot |a_k \delta^{\lambda_k}| \quad \text{for all } \delta \in I.$$

We obtain the analogous estimate for $\mathcal{C}$ using a similar argument. Hence, for any interval $I \subset (0,\infty) \setminus (\mathcal{B} \cup \mathcal{C})$,

$$\text{(2.61)} \quad \begin{cases} \text{there exist unique } k = k(I) \in \{1,\dots,K\} \text{ and } \ell = \ell(I) \in \{1,\dots,L\} \\ \text{such that } |a_k \delta^{\lambda_k}| > 2 \sum_{k' \neq k} |a_{k'} \delta^{\lambda_{k'}}| \text{ and } |b_\ell \delta^{\gamma_\ell}| > 2 \sum_{\ell' \neq \ell} |b_{\ell'} \delta^{\gamma_{\ell'}}| \text{ for all } \delta \in I. \end{cases}$$

The fact that $k = k(I)$ and $\ell = \ell(I)$ are unique is obvious from the above statement.

The endpoints of each nonempty interval $I_{kk'} = [h^-_{kk'}, h^+_{kk'}]$ are solutions of the equations $|a_{k'}\delta^{\lambda_{k'}}| = 5 \cdot |a_k \delta^{\lambda_k}|$ and $|a_{k'}\delta^{\lambda_{k'}}| = 5^{-1} \cdot |a_k\delta^{\lambda_k}|$, namely $h^-_{kk'}$ and $h^+_{kk'}$ are among the numbers

$$\delta_1 = \left(5\left|\frac{a_k}{a_{k'}}\right|\right)^{1/(\lambda_{k'}-\lambda_k)} \quad \text{and} \quad \delta_2 = \left(5^{-1}\left|\frac{a_k}{a_{k'}}\right|\right)^{1/(\lambda_{k'}-\lambda_k)}.$$

Thus, the endpoints $h^-_{kk'}, h^+_{kk'}$ of the intervals $I_{kk'}$ are computable. Moreover, using (2.60) we see that

$$\begin{aligned}\int_{\mathcal{B}} \frac{dt}{t} \leq \sum_{k \neq k'} \int_{I_{kk'}} \frac{dt}{t} &= \sum_{k\neq k'} \log\left(\frac{h^+_{kk'}}{h^-_{kk'}}\right) \\ &= \sum_{k\neq k'} \frac{1}{|\lambda_{k'} - \lambda_k|} \log(25) \leq A, \quad \text{where } A = A(m,n,p),\end{aligned}$$

For a similar reason, the endpoints of the intervals $J_{\ell\ell'}$ are computable and

$$\int_{\mathcal{C}} \frac{dt}{t} \leq A.$$

We replace each pair of intersecting intervals among the $I_{kk'}$ and $J_{\ell\ell'}$ with their union. We continue until all the remaining intervals are pairwise disjoint. Thus, we can compute pairwise disjoint closed intervals I^{bad}_ν and pairwise disjoint open intervals I_μ such that

$$\mathcal{B} \cup \mathcal{C} = \bigcup_{\nu=1}^{\nu_{\max}} I^{\text{bad}}_\nu, \quad (0,\infty) \setminus (\mathcal{B} \cup \mathcal{C}) = \bigcup_{\mu=1}^{\mu_{\max}} I_\mu,$$

and

$$\int_{I^{\text{bad}}_\nu} \frac{dt}{t} \leq \int_{\mathcal{B}\cup\mathcal{C}} \frac{dt}{t} \leq 2A \quad \text{for each } \nu. \tag{2.62}$$

Note that $\nu_{\max} \leq \#\{I_{kk'}\} + \#\{J_{\ell\ell'}\} \leq K^2 + L^2$ and $\mu_{\max} = \nu_{\max} + 1$, hence $\nu_{\max}$ and $\mu_{\max}$ are bounded by a universal constant.

From (2.61), there exist indices $k = k(\mu) \in \{1,\dots,K\}$ and $\ell = \ell(\mu) \in \{1,\dots,L\}$ for each μ such that

$$|a_k\delta^{\lambda_k}| > 2\sum_{k'\neq k} |a_{k'}\delta^{\lambda_{k'}}| \quad \text{and} \quad |b_\ell \delta^{\gamma_\ell}| > 2 \sum_{\ell'\neq \ell} |b_{\ell'}\delta^{\gamma_{\ell'}}| \quad \text{for all } \delta \in I_\mu. \tag{2.63}$$

We can compute $k(\mu)$ and $\ell(\mu)$ for $\mu = 1,\dots,\mu_{\max}$, using a brute-force search.

Let μ be given, and set $k = k(\mu)$ and $\ell = \ell(\mu)$. We have $\eta_{\min}(\delta) = \frac{N(\delta)}{D(\delta)}$, with

$$N(\delta) = a_k\delta^{\lambda_k} + \sum_{k'\neq k} a_{k'}\delta^{\lambda_{k'}} \quad \text{and} \quad D(\delta) = b_\ell\delta^{\gamma_\ell} + \sum_{\ell'\neq\ell} b_{\ell'}\delta^{\gamma_{\ell'}}.$$

From (2.63), we have

$$\frac{1}{2} \leq \frac{N(\delta)}{a_k \delta^{\lambda_k}} \leq \frac{3}{2} \quad \text{and} \quad \frac{1}{2} \leq \frac{D(\delta)}{b_\ell \delta^{\gamma_\ell}} \leq \frac{3}{2} \quad \text{for all } \delta \in I_\mu.$$

Hence,

$$(1/4) \cdot \eta_{\min}(\delta) \leq \frac{a_k \delta^{\lambda_k}}{b_\ell \delta^{\gamma_\ell}} \leq (9/4) \cdot \eta_{\min}(\delta) \quad \text{for all } \delta \in I_\mu.$$

We fix $\delta_\nu \in I_\nu^{\text{bad}}$ for each ν. Note that $e^{-2A} \leq \delta/\delta_\nu \leq e^{2A}$ for all $\delta \in I_\nu^{\text{bad}}$, by (2.62). Hence, (2.49) implies that

$$c \cdot \eta_{\min}(\delta) \leq \eta_{\min}(\delta_\nu) \leq C \cdot \eta_{\min}(\delta) \quad \text{for all } \delta \in I_\nu^{\text{bad}}.$$

We define

$$\eta_*(\delta) = \begin{cases} \frac{a_{k(\mu)} \delta^{\lambda_{k(\mu)}}}{b_{\ell(\mu)} \delta^{\gamma_{\ell(\mu)}}} & \text{if } \delta \in I_\mu, \\ \eta_{\min}(\delta_\nu) & \text{if } \delta \in I_\nu^{\text{bad}}. \end{cases} \tag{2.64}$$

From the previous two paragraphs, we see that $\eta_{\min}(\delta)$ and $\eta_*(\delta)$ differ by at most a universal constant factor for all $\delta \in (0, \infty)$.

The above computations clearly require work at most C.

That completes our description of the procedure APPROXIMATE RATIONAL FUNCTION. $\square$

We have computed a piecewise-monomial function $\eta_*(\delta)$ differing from $\eta_{\min}(\delta)$ by at most a constant factor. Thus, we see that the properties (2.48) and (2.49) of $\eta_{\min}(\delta)$ imply the first and second bullet points below.

ALGORITHM: FIT BASIS TO CONVEX BODY

Given a nonnegative quadratic form q on $\mathcal{P}$, given a point $x \in \mathbb{R}^n$, and given a set $\mathcal{A} \subset \mathcal{M}$: We compute a partition of $(0, \infty)$ into at most C intervals I_ν, and for each I_ν we compute real numbers λ_ν, c_ν with $c_\nu \geq 0$, such that the function

$$\eta_*(\delta) := c_\nu \cdot \delta^{\lambda_\nu} \quad \text{for } \delta \in I_\nu$$

has the following properties.

- Let $\overline{\overline{\sigma}} \subset \mathcal{P}$ be a symmetric convex set that satisfies $\{q \leq \Lambda^{-1}\} \subset \overline{\overline{\sigma}} \subset \{q \leq \Lambda\}$ for a real number $\Lambda \geq 1$. Then, for any $\delta > 0$, $\overline{\overline{\sigma}}$ has an $(\mathcal{A}, x, \eta^{1/2}, \delta)$-basis if $\eta > C(\Lambda) \cdot \eta_*(\delta)$, but not if $\eta < c(\Lambda) \cdot \eta_*(\delta)$.
- Moreover, $c \cdot \eta_*(\delta_1) \leq \eta_*(\delta_2) \leq C \cdot \eta_*(\delta_1)$ whenever $\frac{1}{10}\delta_1 \leq \delta_2 \leq 10\delta_1$.
- The components of the piecewise-monomial function $\eta_*(\delta)$, i.e., the intervals I_ν and the numbers λ_ν, c_ν, can be computed using work and storage at most C.

Here, $c > 0$ and $C \geq 1$ are constants depending only on m,n, and p, while $c(\Lambda) > 0$ and $C(\Lambda) \geq 1$ are constants depending only on m,n,p, and Λ.

2.8. Algorithms for linear functionals

<u>Algorithm: Compress norms</u>

Fix $1 < p < \infty$ and $D \geq 1$. Given linear functionals $\mu_1, \dots, \mu_L : \mathbb{R}^D \to \mathbb{R}$ $(L \geq 1)$, we produce linear functionals $\mu_1^*, \dots, \mu_D^* : \mathbb{R}^D \to \mathbb{R}$ such that

$$c \cdot \sum_{i=1}^{D} |\mu_i^*(v)|^p \leq \sum_{i=1}^{L} |\mu_i(v)|^p \leq C \cdot \sum_{i=1}^{D} |\mu_i^*(v)|^p \quad \text{for all } v \in \mathbb{R}^D.$$

The work and storage used to do so are at most $C'L$. Here, c, C, C' depend only on D and p.

Explanation. In this explanation, c, C, C', etc., depend only on D and p.

We start with a few elementary estimates. Let $(\Omega, d\mu)$ be a probability space. Then, for $f : \Omega \to \mathbb{R}$ measurable, the mean $\overline{f} = \int_\Omega f d\mu$ satisfies

$$|\overline{f}| \leq \Big(\int_\Omega |f|^p d\mu \Big)^{1/p},$$

hence

$$|\overline{f}|^p + \int_\Omega |f - \overline{f}|^p d\mu \leq C \int_\Omega |f|^p d\mu.$$

Also,

$$\int_\Omega |f|^p d\mu \leq C |\overline{f}|^p + C \int_\Omega |f - \overline{f}|^p d\mu.$$

Applying the above to the function $f - b$ for a constant b, we find that

$$c \Big\{ |\overline{f} - b|^p + \int_\Omega |f - \overline{f}|^p d\mu \Big\} \leq \int_\Omega |f - b|^p d\mu \leq C \Big\{ |\overline{f} - b|^p + \int_\Omega |f - \overline{f}|^p d\mu \Big\} \tag{2.65}$$

with $c, C > 0$ depending only on p.

We now return to the task of constructing $\mu_1^*, \dots, \mu_D^*$.

We proceed by induction on D. In the base case $D = 1$, the construction of μ_1^* is trivial. For the induction step, fix $D \geq 2$, and assume we can carry out Compress Norms in dimension $D - 1$. We show how to carry out that algorithm in dimension D.

Let $\mu_1, \dots, \mu_L : \mathbb{R}^D \to \mathbb{R}$ be given linear functionals. We write

$$\mu_i(v_1, \dots, v_D) = \pm [\beta_i v_D - \widetilde{\mu}_i(v_1, \dots, v_{D-1})] \tag{2.66}$$

with $\beta_i \geq 0$, and we let $I = \{i : \beta_i \neq 0\}$. If I is empty, then we succeed simply by setting $\mu_D^* = 0$ and invoking Compress norms in dimension $D - 1$. Suppose I is non-empty. Let

$$\mathbf{B} := \sum_{j \in I} \beta_j^p.$$

We view I as a probability space, with

$$\text{Prob}(i) := \beta_i^p / \mathbf{B} \quad \text{for } i \in I.$$

Then

$$\sum_{i\in I} |\mu_i(v_1,\dots,v_D)|^p = \mathbf{B}\cdot \sum_{i\in I} \mathrm{Prob}(i)\cdot \left|v_D - \beta_i^{-1}\widetilde{\mu}_i(v_1,\dots,v_{D-1})\right|^p.$$

Invoking (2.65), with $b = v_D$ and $f(i) = \beta_i^{-1}\widetilde{\mu}_i(v_1,\dots,v_{D-1})$, we see that

$$\begin{aligned} c\sum_{i\in I}&|\mu_i(v_1,\dots,v_D)|^p \\ &\leq \mathbf{B}\cdot \Big\{|v_D - \overline{\mu}(v_1,\dots,v_{D-1})|^p \\ &\quad + \sum_{i\in I}\mathrm{Prob}(i)\cdot \left|\overline{\mu}(v_1,\dots,v_{D-1}) - \beta_i^{-1}\widetilde{\mu}_i(v_1,\dots,v_{D-1})\right|^p\Big\} \\ &\leq C\sum_{i\in I}|\mu_i(v_1,\dots,v_D)|^p, \end{aligned}$$

where

$$\overline{\mu}(v_1,\dots,v_{D-1}) := \sum_{i\in I}\mathrm{Prob}(i)\cdot\{\beta_i^{-1}\widetilde{\mu}_i(v_1,\dots,v_{D-1})\}. \tag{2.67}$$

Consequently, $\sum_{i=1}^{L}|\mu_i(v_1,\dots,v_D)|^p$ differs by at most a factor of C from

$$\begin{aligned} &\mathbf{B}\cdot|v_D - \overline{\mu}(v_1,\dots,v_{D-1})|^p \\ &+\Big\{\mathbf{B}\cdot\sum_{i\in I}\mathrm{Prob}(i)\cdot|\overline{\mu}(v_1,\dots,v_{D-1}) - \beta_i^{-1}\widetilde{\mu}_i(v_1,\dots,v_{D-1})|^p \\ &\qquad + \sum_{i\notin I}|\widetilde{\mu}_i(v_1,\dots,v_{D-1})|^p\Big\} \\ &= \mathbf{B}\cdot|v_D - \overline{\mu}(v_1,\dots,v_{D-1})|^p + \Big\{\sum_{i=1}^{L}|\beta_i\overline{\mu}(v_1,\dots,v_{D-1}) - \widetilde{\mu}_i(v_1,\dots,v_{D-1})|^p\Big\}, \end{aligned}$$

where

$$\overline{\mu}(v_1,\dots,v_{D-1}) := \mathbf{B}^{-1}\cdot\sum_{i=1}^{L}\beta_i^{p-1}\widetilde{\mu}_i(v_1,\dots,v_{D-1}).$$

Applying COMPRESS NORMS in dimension $D-1$ to the expression in curly brackets in the first box above, we obtain functionals $\mu_1^*,\dots,\mu_{D-1}^* : \mathbb{R}^{D-1}\to\mathbb{R}$ such that $\sum_{i=1}^{D-1}|\mu_i^*(v_1,\dots,v_{D-1})|^p$ differs by at most a factor of C from that expression in curly brackets.

Setting

$$\mu_D^*(v_1,\dots,v_D) := \mathbf{B}^{1/p}\cdot[v_D - \overline{\mu}(v_1,\dots,v_{D-1})], \tag{2.68}$$

we see that $\sum_{i=1}^{D-1}|\mu_i^*(v_1,\dots,v_{D-1})|^p + |\mu_D^*(v_1,\dots,v_D)|^p$ differs by at most a factor of C from $\sum_{i=1}^{L}|\mu_i(v_1,\dots,v_D)|^p$.

This completes our explanation of COMPRESS NORMS; note that the work and storage required are as promised. □

Algorithm: Optimize via matrix

Given $1 < p < \infty$ and given a matrix $(a_{\ell j})_{1 \le \ell \le L\,,\,1 \le j \le J}$, we compute a matrix $(b_{j\ell})_{1 \le j \le J\,,\,1 \le \ell \le L}$ for which the following holds.

Let $y_1, \dots, y_L$ be real numbers. Define

$$x_j^* = \sum_{\ell=1}^{L} b_{j\ell} y_\ell \quad \text{for } j = 1, \dots, J.$$

Then, for any real numbers $x_1, \dots, x_J$, we have

$$\sum_{\ell=1}^{L} \Bigl| y_\ell + \sum_{j=1}^{J} a_{\ell j} x_j^* \Bigr|^p \le C_1 \cdot \sum_{\ell=1}^{L} \Bigl| y_\ell + \sum_{j=1}^{J} a_{\ell j} x_j \Bigr|^p$$

with C_1 depending only on J and p.

The work and storage used to compute $(b_{j\ell})_{1 \le j \le J\,,\,1 \le \ell \le L}$ are at most CL, where C depends only on J.

Explanation. We write c, C, C', etc. to denote constants depending only on J, and $c(p)$, $C(p)$, etc. to denote constants depending only on J and p.

For the case $J = 1$ of our algorithm, we proceed as follows. Let $(a_{\ell 1})_{1 \le \ell \le L}$ be a given matrix.

If $(a_{\ell 1})_{1 \le \ell \le L}$ is identically zero, then the conclusion holds for any choice of $(b_{1\ell})_{1 \le \ell \le L}$ if we take $C_1 = 1$.

We suppose instead that $(a_{\ell 1}) \ne (0)$. Let

$$b_{1\ell} := -\Bigl(\sum_{\ell'=1}^{L} |a_{\ell' 1}|^p \Bigr)^{-1} \cdot |a_{\ell 1}|^{p-1} \operatorname{sgn}(a_{\ell 1}) \quad \text{for } 1 \le \ell \le L, \tag{2.69}$$

where sgn denotes the signum function: $\operatorname{sgn}(\eta) := 1$ for $\eta \ge 0$, $\operatorname{sgn}(\eta) := -1$ for $\eta < 0$. We compute the matrix $(b_{1\ell})_{1 \le \ell \le L}$ using work and storage at most CL.

For given real numbers $y_1, \dots, y_L$ we set

$$x^* = \sum_{\ell=1}^{L} b_{1\ell} y_\ell = - \sum_{\ell : a_{\ell 1} \ne 0} \frac{y_\ell \cdot |a_{\ell 1}|^p}{a_{\ell 1} \cdot \sum_{\ell'} |a_{\ell' 1}|^p}.$$

Define a probability measure $d\mu$ and function f on $\{1, \dots, L\}$ by setting $d\mu(\ell) = |a_{\ell 1}|^p / \sum_k |a_{k1}|^p$, and setting $f(\ell) = y_\ell / a_{\ell 1}$ if $a_{\ell 1} \ne 0$ and $f(\ell) = 0$ otherwise. We then have $x^* = -\int f d\mu$.

By applying (2.65) we see that $\int |f + x^*|^p d\mu \le C(p) \int |f + x|^p d\mu$ for any $x \in \mathbb{R}$. Therefore,

$$\sum_{\ell : a_{\ell 1} \ne 0} |y_\ell / a_{\ell 1} + x^*|^p |a_{\ell 1}|^p \le C(p) \sum_{\ell : a_{\ell 1} \ne 0} |y_\ell / a_{\ell 1} + x|^p |a_{\ell 1}|^p \quad \text{for any } x \in \mathbb{R}.$$

This gives the desired conclusion in the case $J = 1$.

For the general case, we use induction on J.

Let $J \geq 2$, and let $1 < p < \infty$ and $(a_{\ell j})_{1\leq \ell \leq L\,,\,1\leq j\leq J}$ be given. Then

$$\sum_{\ell=1}^{L} \left| y_\ell + \sum_{j=1}^{J} a_{\ell j} x_j \right|^p = \sum_{\ell=1}^{L} \left| \widehat{y}_\ell + \sum_{j=1}^{J-1} a_{\ell j} x_j \right|^p \tag{2.70}$$

with

$$\widehat{y}_\ell = y_\ell + a_{\ell J} \cdot x_J \quad \text{for } \ell = 1, \dots, L. \tag{2.71}$$

Applying our algorithm OPTIMIZE VIA MATRIX recursively to $1 < p < \infty$ and $(a_{\ell j})_{1\leq \ell\leq L\,,\,1\leq j\leq J-1}$, we produce a matrix $(\widehat{b}_{j\ell})_{1\leq j\leq J-1\,,\,1\leq \ell\leq L}$, for which the following holds.

- Let $\widehat{y}_1, \dots, \widehat{y}_L$ be real numbers, and set

$$\widehat{x}_j = \sum_{\ell=1}^{L} \widehat{b}_{j\ell} \widehat{y}_\ell \quad \text{for } j = 1, \dots, J-1. \tag{2.72}$$

  Then, for any real numbers $x_1, \dots, x_{J-1}$, we have

$$\sum_{\ell=1}^{L} \left| \widehat{y}_\ell + \sum_{j=1}^{J-1} a_{\ell j} \widehat{x}_j \right|^p \leq C(p) \sum_{\ell=1}^{L} \left| \widehat{y}_\ell + \sum_{j=1}^{J-1} a_{\ell j} x_j \right|^p. \tag{2.73}$$

From (2.70)–(2.73), we draw the following conclusion.

Let real numbers $y_1, \dots, y_L$, and $x_1, \dots, x_J$ be given. Define $\widehat{y}_1, \dots, \widehat{y}_L$ by (2.71), next define $\widehat{x}_1, \dots, \widehat{x}_{J-1}$ by (2.72), and finally set

$$\widehat{x}_J = x_J. \tag{2.74}$$

Then

$$\sum_{\ell=1}^{L} \left| y_\ell + \sum_{j=1}^{J} a_{\ell j} \widehat{x}_j \right|^p \leq C(p) \sum_{\ell=1}^{L} \left| y_\ell + \sum_{j=1}^{J} a_{\ell j} x_j \right|^p \tag{2.75}$$

and

$$\widehat{x}_j = \sum_{\ell=1}^{L} \widehat{b}_{j\ell} \cdot (y_\ell + a_{\ell J} \widehat{x}_J) \quad \text{for } j = 1, \dots, J-1. \tag{2.76}$$

Thus,

$$\widehat{x}_j = \sum_{\ell=1}^{L} \widehat{b}_{j\ell} y_\ell + g_j \widehat{x}_J \quad \text{for } j = 1, \dots, J-1, \text{ where} \tag{2.77}$$

$$g_j = \sum_{\ell=1}^{L} \widehat{b}_{j\ell} a_{\ell J} \quad \text{for } j = 1, \dots, J-1. \tag{2.78}$$

Next, note that

$$\begin{aligned} y_\ell + \sum_{j=1}^{J} a_{\ell j}\widehat{x}_j &= y_\ell + \sum_{j=1}^{J-1} a_{\ell j}\Big[\sum_{\ell'=1}^{L} \widehat{b}_{j\ell'}y_{\ell'} + g_j\widehat{x}_J\Big] + a_{\ell J}\widehat{x}_J \\ &= \Big\{y_\ell + \sum_{j=1}^{J-1} a_{\ell j}\sum_{\ell'=1}^{L}\widehat{b}_{j\ell'}y_{\ell'}\Big\} + \Big\{a_{\ell J} + \sum_{j=1}^{J-1} a_{\ell j}g_j\Big\}\widehat{x}_J. \end{aligned}$$

We set

$$y_\ell^{\text{ouch}} = y_\ell + \sum_{j=1}^{J-1} a_{\ell j}\sum_{\ell'=1}^{L}\widehat{b}_{j\ell'}y_{\ell'} \quad \text{for } \ell = 1,\dots,L \tag{2.79}$$

and

$$h_\ell = a_{\ell J} + \sum_{j=1}^{J-1} a_{\ell j}g_j \quad \text{for } \ell = 1,\dots,L. \tag{2.80}$$

Thus,

$$\sum_{\ell=1}^{L}\Big|y_\ell + \sum_{j=1}^{J} a_{\ell j}\widehat{x}_j\Big|^p = \sum_{\ell=1}^{L}|y_\ell^{\text{ouch}} + h_\ell\widehat{x}_J|^p. \tag{2.81}$$

Here, (2.81) holds whenever $\widehat{x}_1,\dots,\widehat{x}_{J-1}$ are determined from $\widehat{x}_J$ via (2.77).

Note that it is too expensive to compute y_ℓ^{ouch} for all ℓ ($1 \le \ell \le L$); that computation would require $\sim L^2J$ work. However, the y_ℓ^{ouch} are determined by (2.79); they are independent of our choice of $\widehat{x}_J$.

Applying the known case $J = 1$ of our algorithm Optimize via matrix, we compute from the h_ℓ a vector of coefficients γ_ℓ ($1 \le \ell \le L$), for which the following holds.

Let $\check{y}_1,\dots,\check{y}_L$ be real numbers. Set $\check{x} = \sum_{\ell=1}^{L}\gamma_\ell\check{y}_\ell$. Then

$$\sum_{\ell=1}^{L}|\check{y}_\ell + h_\ell\check{x}|^p \le C(p)\sum_{\ell=1}^{L}|\check{y}_\ell + h_\ell\widehat{x}|^p$$

for any real number $\widehat{x}$.

Taking $\check{y}_\ell = y_\ell^{\text{ouch}}$ for $\ell = 1,\dots,L$, we learn the following. Let

$$\check{x}_J = \sum_{\ell=1}^{L}\gamma_\ell\, y_\ell^{\text{ouch}}, \tag{2.82}$$

and then define $\check{x}_1,\dots,\check{x}_{J-1}$ from $\check{x}_J$ via (2.77), i.e.,

$$\check{x}_j = \sum_{\ell=1}^{L}\widehat{b}_{j\ell}y_\ell + g_j\check{x}_J \quad \text{for } j = 1,\dots,J-1. \tag{2.83}$$

Then

$$\sum_{\ell=1}^{L}\left|y_\ell+\sum_{j=1}^{J}a_{\ell j}\check{x}_j\right|^p \le C(p)\sum_{\ell=1}^{L}\left|y_\ell+\sum_{j=1}^{J}a_{\ell j}\widehat{x}_j\right|^p. \tag{2.84}$$

(See (2.81).)

From (2.75) and (2.84), we see that

$$\sum_{\ell=1}^{L}\left|y_\ell+\sum_{j=1}^{J}a_{\ell j}\check{x}_j\right|^p \le C(p)\sum_{\ell=1}^{L}\left|y_\ell+\sum_{j=1}^{J}a_{\ell j}x_j\right|^p. \tag{2.85}$$

Here, $\check{x}_1,\dots,\check{x}_J$ are computed from (2.82),(2.83); and $x_1,\dots,x_J$ are arbitrary.

We produce efficient formulas for the $\check{x}_j$. Putting (2.79) into (2.82), we find that

$$\check{x}_J=\sum_{\ell=1}^{L}\gamma_\ell\cdot\left\{y_\ell+\sum_{j=1}^{J-1}a_{\ell j}\sum_{\ell'=1}^{L}\widehat{b}_{j\ell'}y_{\ell'}\right\}=\sum_{\ell=1}^{L}\gamma_\ell\cdot y_\ell+\sum_{\ell'=1}^{L}\sum_{j=1}^{J-1}\left[\sum_{\ell=1}^{L}\gamma_\ell a_{\ell j}\right]\widehat{b}_{j\ell'}y_{\ell'}$$

$$=\sum_{\ell=1}^{L}\left\{\gamma_\ell+\sum_{j=1}^{J-1}\left[\sum_{\ell'=1}^{L}\gamma_{\ell'}a_{\ell' j}\right]\widehat{b}_{j\ell}\right\}\cdot y_\ell.$$

Therefore, setting

$$\Delta_j=\sum_{\ell=1}^{L}\gamma_\ell a_{\ell j}\quad \text{for } j=1,\dots,J-1 \tag{2.86}$$

and

$$b^{\#\#}_{J\ell}=\gamma_\ell+\sum_{j=1}^{J-1}\Delta_j\widehat{b}_{j\ell}\quad \text{for } \ell=1,\dots,L \tag{2.87}$$

we find that

$$\check{x}_J=\sum_{\ell=1}^{L}b^{\#\#}_{J\ell}y_\ell. \tag{2.88}$$

Substituting (2.88) into (2.83), we find that

$$\check{x}_j=\sum_{\ell=1}^{L}\{\widehat{b}_{j\ell}+g_jb^{\#\#}_{J\ell}\}y_\ell\quad \text{for } j=1,\dots,J-1.$$

Thus, setting

$$b^{\#\#}_{j\ell}=\widehat{b}_{j\ell}+g_jb^{\#\#}_{J\ell}\quad \text{for } j=1,\dots,J-1, \tag{2.89}$$

we have

$$\check{x}_j = \sum_{\ell=1}^{L} b^{\#\#}_{j\ell} y_\ell \quad \text{for } j = 1, \dots, J-1. \tag{2.90}$$

Recalling (2.88), we see that (2.90) holds for $j = 1, \dots, J$. Thus, with $\check{x}_1, \dots, \check{x}_J$ defined by (2.90), we have

$$\sum_{\ell=1}^{L} \left| y_\ell + \sum_{j=1}^{J} a_{\ell j} \check{x}_j \right|^p \le C(p) \sum_{\ell=1}^{L} \left| y_\ell + \sum_{j=1}^{J} a_{\ell j} x_j \right|^p$$

for any real numbers $x_1, \dots, x_J$. (See (2.85).)

So the matrix $(b^{\#\#}_{j\ell})_{1 \le j \le J\,,\, 1 \le \ell \le L}$ is as promised in our algorithm.

Let us review the computation of $(b^{\#\#}_{j\ell})$.

- Recursively, we apply the algorithm Optimize via matrix to the arguments $(a_{\ell j})_{1 \le \ell \le L\,,\, 1 \le j \le J-1}$, and p; this yields $(\widehat{b}_{j\ell})_{1 \le j \le J-1\,,\, 1 \le \ell \le L}$.
- Next, we compute from (2.78) the quantities
$$g_j = \sum_{\ell=1}^{L} \widehat{b}_{j\ell}\, a_{\ell J} \quad \text{for } j = 1, \dots, J-1.$$
- We then compute from (2.80) the numbers
$$h_\ell = a_{\ell J} + \sum_{j=1}^{J-1} a_{\ell j}\, g_j \quad \text{for } \ell = 1, \dots, L.$$
- We apply the case $J = 1$ of Optimize via matrix to the $L \times 1$ matrix (h_ℓ), to produce the numbers γ_ℓ $(\ell = 1, \dots, L)$.
- From (2.86) we then compute the numbers
$$\Delta_j = \sum_{\ell=1}^{L} \gamma_\ell\, a_{\ell j} \quad \text{for } j = 1, \dots, J-1.$$
- We set
$$b^{\#\#}_{J\ell} = \gamma_\ell + \sum_{j=1}^{J-1} \Delta_j\, \widehat{b}_{j\ell} \quad \text{for } \ell = 1, \dots, L.$$
(See (2.87).)
- Finally, we set
$$b^{\#\#}_{j\ell} = \widehat{b}_{j\ell} + g_j b^{\#\#}_{J\ell} \quad \text{for } j = 1, \dots, J-1 \text{ and } \ell = 1, \dots, L.$$
(See (2.89).)

One can now check easily that our algorithm uses work and storage at most CL, as promised.

This concludes our explanation of the algorithm Optimize via matrix. □

3. Statement of the main technical results

Suppose that $E \subset \frac{1}{32}Q^\circ$ is finite, where $Q^\circ = [0,1)^n$. We assume that $N = \#(E) \geq 2$.

If $\mathcal{A} \subsetneq \mathcal{M}$, then let $\mathcal{A}^- \subset \mathcal{M}$ denote the maximal subset less than $\mathcal{A}$. (See Section 2.6 for the definition of the order relation $<$ on $2^{\mathcal{M}}$.)

For each $\mathcal{A} \subset \mathcal{M}$, we will define the following.

- A decomposition $\mathrm{CZ}(\mathcal{A})$ of Q° into dyadic cubes. We guarantee the following.

(CZ1) If $Q, Q' \in \mathrm{CZ}(\mathcal{A})$ and $Q \leftrightarrow Q'$, then $\frac{1}{2}\delta_Q \leq \delta_{Q'} \leq 2\delta_Q$ ("good geometry").

(CZ2) If $Q \in \mathrm{CZ}(\mathcal{A})$ and $\delta_Q \leq c_*(\mathcal{A})$ then $S(\mathcal{A})Q$ is not tagged with $(\mathcal{A}, \epsilon_1(\mathcal{A}))$. Moreover, $S(\mathcal{A}) = 9$ for $\mathcal{A} = \mathcal{M}$.

(CZ3) In the case $\mathcal{A} \neq \mathcal{M}$:
If $Q \in \mathrm{CZ}(\mathcal{A})$, $Q' \in \mathrm{CZ}(\mathcal{A}^-)$, $Q' \subset Q$ and $\delta_{Q'} \leq c_*(\mathcal{A})\delta_Q$ then the cube $3Q$ is tagged with $(\mathcal{A}, \epsilon_2(\mathcal{A}))$.

(CZ4) In the case $\mathcal{A} = \mathcal{M}$:
If $Q \in \mathrm{CZ}(\mathcal{M})$, then $3Q$ is tagged with $(\mathcal{M}, \epsilon_2(\mathcal{M}))$.

(CZ5) In the case $\mathcal{A} \neq \mathcal{M}$:
$\mathrm{CZ}(\mathcal{A}^-)$ refines $\mathrm{CZ}(\mathcal{A})$.
(We do not exclude the possibility that $\mathrm{CZ}(\mathcal{A}^-) = \mathrm{CZ}(\mathcal{A})$.)

Moreover, $c_*(\mathcal{A}), \epsilon_1(\mathcal{A}), \epsilon_2(\mathcal{A}) \in (0,1)$ and $S(\mathcal{A}) \geq 1$.

- A collection $\mathrm{CZ}_{\mathrm{main}}(\mathcal{A})$ consisting of all cubes $Q \in \mathrm{CZ}(\mathcal{A})$ such that $\frac{65}{64}Q \cap E \neq \emptyset$.

- For each $Q \in \mathrm{CZ}_{\mathrm{main}}(\mathcal{A})$, a list of functionals in short form $\Omega(Q,\mathcal{A}) \subset \left[\mathbb{X}(\frac{65}{64}Q \cap E)\right]^*$ (the "assists") such that
$$\sum_{Q \in \mathrm{CZ}_{\mathrm{main}}(\mathcal{A})} \sum_{\omega \in \Omega(Q,\mathcal{A})} \mathrm{depth}(\omega) \leq CN.$$

- For each $Q \in \mathrm{CZ}_{\mathrm{main}}(\mathcal{A})$, a list of functionals $\Xi(Q,\mathcal{A}) \subset \left[\mathbb{X}(\frac{65}{64}Q \cap E) \oplus \mathcal{P}\right]^*$, each having $\Omega(Q,\mathcal{A})$-assisted depth at most C. We guarantee that
$$\sum_{Q \in \mathrm{CZ}_{\mathrm{main}}(\mathcal{A})} \#\left[\Xi(Q,\mathcal{A})\right] \leq CN.$$
We set
$$M_{(Q,\mathcal{A})}(f,P) = \left(\sum_{\xi \in \Xi(Q,\mathcal{A})} |\xi(f,P)|^p \right)^{1/p}.$$
For each $(f,P) \in \mathbb{X}(\frac{65}{64}Q \cap E) \oplus \mathcal{P}$, we guarantee that
$$c \cdot \|(f,P)\|_{(1+a(\mathcal{A}))Q} \leq M_{(Q,\mathcal{A})}(f,P) \leq C \cdot \|(f,P)\|_{\frac{65}{64}Q}.$$
Here, $0 < a(\mathcal{A}) \leq 1/64$.

- For each $Q \in CZ_{\text{main}}(\mathcal{A})$, a linear map $T_{(Q,\mathcal{A})} : \mathbb{X}(\frac{65}{64}Q \cap E) \oplus \mathcal{P} \to \mathbb{X}$ with the following properties.

  (E1) $T_{(Q,\mathcal{A})}(f, P) = f$ on $(1 + a(\mathcal{A}))Q \cap E$ for each (f, P).

  (E2) $\|T_{(Q,\mathcal{A})}(f, P)\|^p_{\mathbb{X}((1+a(\mathcal{A}))Q)} + \delta_Q^{-mp}\|T_{(Q,\mathcal{A})}(f, P) - P\|^p_{L^p((1+a(\mathcal{A}))Q)} \leq C\left[M_{(Q,\mathcal{A})}(f, P)\right]^p$ for each (f, P).

  (E3) $T_{(Q,\mathcal{A})}$ has $\Omega(Q, \mathcal{A})$-assisted depth at most C.

- The constants $c_*(\mathcal{A}), S(\mathcal{A}), \epsilon_1(\mathcal{A}), \epsilon_2(\mathcal{A}), a(\mathcal{A}), c, C$ depend only on m, n, p, and $\mathcal{A}$. The constant $S(\mathcal{A})$ is a positive integer.

Remark 29. Note that both $\Xi(Q, \mathcal{A})$ and $\Omega(Q, \mathcal{A})$ are lists, hence they may contain more than one copy of the same linear functional. In the sums in the third and fourth bullet points, we include separate summands for each occurrence of a given functional $\xi \in \Xi(Q, \mathcal{A})$ or $\omega \in \Omega(Q, \mathcal{A})$. See Section 2.1 for more information on our notation concerning lists.

To compute the objects defined above, we will produce the following algorithms.

- Algorithm: CZ-Oracle. We perform one-time work at most $CN \log N$ in space CN, after which we can answer queries. A query consists of a point $\underline{x} \in Q^\circ$. The response to the query $\underline{x}$ is the list of all cubes $Q \in CZ(\mathcal{A})$ such that $\underline{x} \in \frac{65}{64}Q$. To answer the query requires work and storage at most $C \log N$.

- Algorithm: Compute main-cubes. With work at most $CN \log N$ in space CN, we compute the collection of cubes $CZ_{\text{main}}(\mathcal{A})$.

- Algorithm: Compute functionals. With work at most $CN \log N$ in space CN, we compute the following.
  - For each cube $Q \in CZ_{\text{main}}(\mathcal{A})$, the list of functionals $\Omega(Q, \mathcal{A})$, with each functional written in short form.
  - For each cube $Q \in CZ_{\text{main}}(\mathcal{A})$, the list of functionals $\Xi(Q, \mathcal{A})$, with each functional written in short form (in terms of the assists $\Omega(Q, \mathcal{A})$).

- Algorithm: Compute extension operator. We perform one-time work at most $CN \log N$ in space CN, after which we can answer queries. A query consists of a cube $Q \in CZ_{\text{main}}(\mathcal{A})$ and a point $\underline{x} \in Q^\circ$. The response to the query $(Q, \underline{x})$ is a short form description of the $\Omega(Q, \mathcal{A})$-assisted bounded depth linear map

$$(f, P) \in \mathbb{X}\left(\frac{65}{64}Q \cap E\right) \oplus \mathcal{P} \mapsto J_{\underline{x}} T_{(Q,\mathcal{A})}(f, P).$$

  To answer the query requires work and storage at most $C \log N$.

4. Data structures

4.1. Algorithms for dyadic cubes

4.1.1. Dyadic cuboids. We define a *dyadic interval* to be a subinterval $[a, b) \subset [0, \infty)$ where

$$a = \sum_{\nu=-\infty}^{\infty} \delta_\nu 2^\nu, \quad b = \sum_{\nu=-\infty}^{\infty} \delta'_\nu 2^\nu, \quad \text{each } \delta_\nu, \delta'_\nu = 0 \text{ or } 1,$$

only finitely many δ_ν, δ'_ν are nonzero, and for some μ,

$$\delta_\nu = \delta'_\nu \text{ for } \nu > \mu; \ \delta_\mu = 0, \ \delta'_\mu = 1; \ \ \delta_\nu = \delta'_\nu = 0 \text{ for } \nu < \mu.$$

The *dyadic cuboids* to be defined in a moment, will be Cartesian products of dyadic intervals. Thus, by definition, a dyadic cuboid $Q \subset \mathbb{R}^n$ will be a subset of $[0, \infty)^n$.

Fix a dimension n. A *dyadic cuboid* Q is a Cartesian product of the form

$$[a_1, b_1) \times \cdots \times [a_n, b_n) \subset \mathbb{R}^n,$$

where each $[a_i, b_i)$ is a dyadic interval, and one of the following holds:

(1) All the $[a_i, b_i)$ have the same length,
 or for some j $(1 \le j < n)$,

(2) All the $[a_i, b_i)$ $(1 \le i \le j)$ have the same length, and each $[a_i, b_i)$ $(j < i \le n)$ has length $\frac{1}{2}(b_1 - a_1)$.

To *bisect* the cuboid Q means the following.

Case 1: Suppose Q is as in (1). Then we bisect $[a_n, b_n)$ into two dyadic intervals $I' = [a_n, \frac{a_n+b_n}{2})$ and $I'' = [\frac{a_n+b_n}{2}, b_n)$.

To *bisect* Q, we express Q as the disjoint union of the two dyadic cuboids

$$Q' = [a_1, b_1) \times \cdots \times [a_{n-1}, b_{n-1}) \times I'$$

and

$$Q'' = [a_1, b_1) \times \cdots \times [a_{n-1}, b_{n-1}) \times I''$$

We call Q' the *lesser dyadic child* of Q, and we call Q'' the *greater dyadic child* of Q.

Case 2: For some j $(1 \le j < n)$, suppose Q is as in (2). Then we bisect $[a_j, b_j)$ into two dyadic intervals $I' = [a_j, \frac{a_j+b_j}{2})$ and $I'' = [\frac{a_j+b_j}{2}, b_j)$.

To *bisect* Q is to express Q as a disjoint union of the dyadic cuboids

$$Q' = [a_1, b_1) \times \cdots \times [a_{j-1}, b_{j-1}) \times I' \times [a_{j+1}, b_{j+1}) \times \cdots \times [a_n, b_n)$$

and

$$Q'' = [a_1, b_1) \times \cdots \times [a_{j-1}, b_{j-1}) \times I'' \times [a_{j+1}, b_{j+1}) \times \cdots \times [a_n, b_n)$$

We call Q' the *lesser dyadic child* of Q, and we call Q'' the *greater dyadic child* of Q.

To understand dyadic cuboids and their dyadic children, it is convenient to think of base 2 expansions of real numbers. Let $\mathcal{DR}$ ("dyadic rationals") be the set of all sums of the form $\sum_{\nu=-\infty}^{\infty} \delta(\nu) 2^\nu$, where finitely many $\delta(\nu)$ are equal to 1, and all other $\delta(\nu)$ are equal to zero.

We define a map $\psi : \mathcal{DR}^n \to \mathcal{DR}$ as follows.

Let $x = (x_1, \dots, x_n) \in \mathcal{DR}^n$, with each $x_i = \sum_{\nu=-\infty}^{\infty} \delta_i(\nu) 2^\nu$ as in the definition of $\mathcal{DR}$. Then we define

$$\psi(x) = \sum_{\nu=-\infty}^{\infty} \sum_{i=1}^{n} \delta_i(\nu) 2^{\nu n + i} \in \mathcal{DR}.$$

Thus, ψ is a 1-1 correspondence between $\mathcal{DR}^n$ and $\mathcal{DR}$.

We define a 1-1 correspondence between dyadic cuboids in $\mathbb{R}^n$ and dyadic intervals in $\mathbb{R}$, by the following rule:

The dyadic cuboid $Q \subset \mathbb{R}^n$ corresponds to the dyadic interval $I \subset \mathbb{R}$ if and only if $\mathcal{DR} \cap I = \psi((\mathcal{DR})^n \cap Q)$. By thinking about base 2 expansions of numbers, one sees easily that this is indeed a 1-1 correspondence between dyadic cuboids in $\mathbb{R}^n$ and dyadic intervals in $\mathbb{R}$. Let us denote this 1-1 correspondence by $I = \Psi(Q)$.

Suppose that Q is a dyadic cuboid, with lesser dyadic child Q' and greater dyadic child Q''. Then $\Psi(Q')$ and $\Psi(Q'')$ are the two dyadic children of the dyadic interval $\Psi(Q)$; and $\Psi(Q')$ lies to the left of $\Psi(Q'')$. Again, we leave to the reader the verification of this fact.

We now define a binary relation on dyadic cuboids, and another binary relation on dyadic intervals. We will see that these two relations are both order relations, and moreover, the two order relations are equivalent via the 1-1 correspondence Ψ.

For cuboids: Let Q_1, Q_2 be distinct dyadic cuboids.

- If $Q_1 \subset Q_2$, then we say that $Q_2 < Q_1$.
- If $Q_2 \subset Q_1$, then we say that $Q_1 < Q_2$.
- Suppose Q_1 and Q_2 are disjoint. Let Q be the least common ancestor of Q_1 and Q_2 among dyadic cuboids. Let Q', Q'' be the lesser and greater dyadic children of Q, respectively. Then one of Q_1, Q_2 is contained in Q', and the other is contained in Q''.
  - If $Q_1 \subset Q'$ and $Q_2 \subset Q''$, then we say that $Q_1 < Q_2$.
  - If $Q_2 \subset Q'$ and $Q_1 \subset Q''$, then we say that $Q_2 < Q_1$.

For dyadic intervals: Let I_1, I_2 be distinct dyadic intervals.

- If $I_1 \subset I_2$, then we say that $I_2 < I_1$.
- If $I_2 \subset I_1$, then we say that $I_1 < I_2$.
- If I_1 and I_2 are disjoint, then let I be the smallest dyadic interval containing I_1 and I_2. We bisect I into I' and I'', with I' lying to the left of I''. Then one of I_1, I_2 is contained in I', and the other is contained in I''.
  - If $I_1 \subset I'$ and $I_2 \subset I''$, then we say that $I_1 < I_2$.
  - If $I_2 \subset I'$ and $I_1 \subset I''$, then we say that $I_2 < I_1$.

Thus, we have defined binary relations $<$ on dyadic cuboids, and on dyadic intervals. It is clear that these two relations correspond to each other via the 1-1 correspondence Ψ between dyadic cuboids and dyadic intervals.

Next, we check that $<$ is an order relation. To see this, it is most convenient to work with dyadic intervals. By examining each case mentioned above, we see that $[a_1, b_1) < [a_2, b_2)$ if and only if either $[a_1 < a_2]$ or $[a_1 = a_2 \text{ and } b_2 < b_1]$.

This makes it obvious that $<$ is an order relation.

We will make use of the following.

Proposition 30.

(1) *Let* I_1, I_2 *be dyadic intervals, and suppose* $I_1 < I_2$*. Then either* $I_2 \subset I_1$*, or* $I_1 \cap I_2 = \emptyset$*.*

(2) *Let* I_1, I_2, I_3 *be dyadic intervals, and suppose* $I_1 < I_2 < I_3$*. If* $I_1 \cap I_2 = \emptyset$ *then* $I_1 \cap I_3 = \emptyset$*.*

Proof. (1) holds simply because we cannot have $I_1 \subset I_2$ when $I_1 < I_2$.

To check (2), let $I_i = [a_i, b_i)$ for $i = 1, 2, 3$. We have $a_1 \leq a_2$, and $[a_1, b_1) \cap [a_2, b_2) = \emptyset$. Hence $a_2 \geq b_1$. Since $I_2 < I_3$, we have also $a_3 \geq a_2$. Therefore, $a_3 \geq b_1$, and thus $[a_1, b_1) \cap [a_3, b_3) = \emptyset$, as claimed. □

Corollary 31. *Let* $I_1 < I_2 < \cdots < I_N$ *for dyadic intervals* $I_1, \ldots, I_N$ $(N \geq 2)$*. Then one of the following holds:*

- *All of* $I_2, \cdots, I_N$ *are contained in* I_1*.*
- *All of* $I_2, \ldots, I_N$ *are disjoint from* I_1*.*
- *For some* j $(2 \leq j < N)$*, we find that* $I_2, \ldots, I_j \subset I_1$ *and* $I_{j+1}, \ldots, I_N$ *are disjoint from* I_1*.*

Corollary 32. *Let* $Q_1, \ldots, Q_N$ *be dyadic cuboids* $(N \geq 2)$*, and suppose* $Q_1 < Q_2 < \cdots < Q_N$*. Then one of the following holds:*

- *All of* $Q_2, \cdots, Q_N$ *are contained in* Q_1*.*
- *All of* $Q_2, \ldots, Q_N$ *are disjoint from* Q_1*.*
- *For some* j $(2 \leq j < N)$*, we find that* $Q_2, \ldots, Q_j \subset Q_1$ *and* $Q_{j+1}, \ldots, Q_N$ *are disjoint from* Q_1*.*

We briefly discuss the computer implementation of dyadic cuboids. In our infinite-precision model of computation (see Section 2.2), a dyadic cuboid Q can be stored using at most C memory locations[2]. We can compute the lesser and greater children of a given dyadic cuboid. We can determine whether two given dyadic cuboids are disjoint, and if not, then we can determine which one contains the other. We can also decide whether two given cuboids are equal. These operations

[2]In the finite-precision model of computation to be described in [21], we will deal only with dyadic cuboids whose sides have length between $2^{-C\overline{S}}$ and $2^{+C\overline{S}}$ for an integer $\overline{S} \geq 1$. We will assume that an $\overline{S}$-bit word can be stored in a single memory location.

require work and storage at most C. (In this paragraph C denotes a constant depending only on the dimension n.) We note also that we can compute the least common ancestor of two given dyadic cuboids using work and storage at most C. (Recall that in our model of computation it takes a single operation to compute the smallest dyadic interval containing two given dyadic intervals.) It follows that we can compare two given cuboids under the order relation $<$ using work and storage at most C.

4.1.2. Preliminary definitions. A *B-Tree* is a rooted finite tree T such that every node has zero, one or two children. We write $\mathrm{root}(T)$ to denote the root of T.

Let $x \in T$ be a node. Then $\mathrm{Descendants}(x, T)$ denotes the set of descendants of x in the tree T, and $\mathrm{Nondescendants}(x, T)$ denotes the set $T \setminus \mathrm{Descendants}(x, T)$.

If $x = \mathrm{root}(T)$, then of course $\mathrm{Nondescendants}(x, T)$ is empty.

If $x \neq \mathrm{root}(T)$, then $\mathrm{Nondescendants}(x, T)$ is again a B-Tree, with the same root as T.

In any case, $\mathrm{Descendants}(x, T)$ is a BTree with root x.

(Here, we adopt the convention that each node is a descendant of itself.)

For any BTree T, we write $\#(T)$ for the number of nodes in T.

When we implement a BTree T in the computer, we store the nodes of T, a pointer to the root of T, and a pointer from each node of T (except the root) to its parent. Also, we mark each node to indicate whether it is a leaf (recall that a leaf is a node with no children); and we mark each internal node (i.e. each non-leaf) with pointers to each of its children.

A *binary tree* is a BTree such that each node has either zero or two children.

DTrees and ADTrees

Fix $1 < p < \infty$, $n \geq 1$, $D \geq 1$.

A *DTree* is a BTree T each of whose nodes x is identified with a dyadic cuboid $Q_x \subset \mathbb{R}^n$, such that the following hold.

- Let y be a child of x in T. Then Q_y is a proper sub-cuboid of Q_x.
- Let y, z be distinct children of x in T. Then Q_y and Q_z are disjoint.

An *ADTree* is a DTree T each of whose nodes x is marked with D linear functionals $\mu_1^x, \dots, \mu_D^x$ on $\mathbb{R}^D$.

("D" in "DTree" stands for "dyadic"; "AD" in "ADTree" stands for "augmented dyadic".)

<u>Algorithm: BTree1</u>

Given a BTree T with $\#(T) \geq 2$, we produce a node $x_{\text{split}} \in T$, other than the root of T, such that

$$\#[\mathrm{Descendants}(x_{\text{split}}(T), T)] \leq \frac{9}{10}\#(T)$$

and

$$\#[\mathrm{Nondescendants}(x_{\text{split}}(T), T))] \leq \frac{9}{10}\#(T).$$

The work and storage used to do so are at most $C \cdot \#(T)$ for a universal constant C.

Explanation. We first mark each node of T with the number of its descendants. We then start with $\tilde{x} = \text{root}(T)$. Initially, $\#[\text{Descendants}(\tilde{x}, T)] = \#(T) > \frac{9}{10}\#(T)$.

```
While $\left(\#[\text{Descendants}(\tilde{x}, T)] > \frac{9}{10}\#(T)\right)$
    {/* Note that $\tilde{x}$ cannot be a leaf of $T$, hence there are one or two children
     of $\tilde{x}$. */
     We let $\tilde{y}$ be a child of $\tilde{x}$ having as many descendants as possible (among
     the children of $\tilde{x}$). We then set $\tilde{x} := \tilde{y}$. }
```

The above loop will terminate, since otherwise we would obtain an infinite descending sequence in the finite tree T.

When the loop terminates, we have

$$\#[\text{Descendants}(\tilde{x}, T)] \le \frac{9}{10}\#(T). \tag{4.1}$$

We will check also that $\tilde{x}$ is not the root of T, and that

$$\#[\text{Nondescendants}(\tilde{x}, T)] \le \frac{9}{10}\#(T). \tag{4.2}$$

Since the work and storage of the above procedures are at most $C \cdot \#(T)$, we can return $x_{\text{split}}(T) = \tilde{x}$, and our algorithm will perform as promised.

Thus, it remains only to check that $\tilde{x}$ is not the root of T, and that (4.2) holds.

That $\tilde{x}$ is not the root of T follows at once from (4.1).

To check (4.2), we note that $\tilde{x}$ arose from its parent $\tilde{x}^+$ by executing our loop for the last time. We have $\#[\text{Descendants}(\tilde{x}^+, T)] > \frac{9}{10}\#(T)$ since we executed the loop to produce $\tilde{x}$ from $\tilde{x}^+$.

Also,

$$\begin{aligned}
\#[\text{Descendants}(\tilde{x}^+, T)] &= 1 + \sum_{y \text{ children of } \tilde{x}^+} \#[\text{Descendants}(y, T)] \\
&\le 1 + 2 \cdot \max\{\#[\text{Descendants}(y, T)] : \ y \text{ children of } \tilde{x}^+\} \\
&= 1 + 2 \cdot \#[\text{Descendants}(\tilde{x}, T)]\},
\end{aligned}$$

since $\tilde{x}^+$ has at most 2 children, and since $\tilde{x}$ has at least as many descendants as any child of $\tilde{x}^+$.

Therefore, $1+2\cdot\#[\text{Descendants}(\tilde{x}, T)] > \frac{9}{10}\#(T)$, hence $\#[\text{Descendants}(\tilde{x}, T)] > \frac{9}{20}\#(T) - \frac{1}{2}$. Since $\#(T) \ge 2$, we find that

$$\#[\text{Descendants}(\tilde{x}, T)] > \frac{2}{20}\#(T) + \left(\frac{7}{20}\#(T) - \frac{1}{2}\right) > \frac{1}{10}\#(T).$$

Thus, $\#[\text{Descendants}(\tilde{x}, T)] > \frac{1}{10}\#(T)$, from which (4.2) follows at once.

This completes our explanation of the algorithm BTREE1. □

4.1.3. Control trees. Let T be a BTree. If $x, y \in T$, then we write $x \leq y$ if and only if x is a descendant of y in T.

Let $\widetilde{T}$ be a BTree. We call $\widetilde{T}$ a *sub-tree* of T if $\widetilde{T}$ consists of nodes in T and if the following condition holds: for any nodes $x \leq y \leq z$ in T, if $x, z \in \widetilde{T}$ then $y \in \widetilde{T}$.

A *control tree candidate* for T is a finite binary tree $\mathbb{T}$ (i.e., each internal node of $\mathbb{T}$ has exactly two children), whose nodes are marked as follows.

- *Let* ξ *be any node of* $\mathbb{T}$. Then ξ is marked by a pointer to a BTree called $\mathrm{BT}(\xi)$, which is a sub-tree of T. We mark ξ with a pointer to a node $x_{\text{root}}(\xi) \in T$ which is the root of $\mathrm{BT}(\xi)$.
- *Let* ξ *be any internal node of* $\mathbb{T}$. Then the two children of ξ in $\mathbb{T}$ are marked separately as $\mathrm{gochild}(\xi)$ and $\mathrm{staychild}(\xi)$; and the node ξ is marked by a node $x_{\text{split}}(\xi) \in \mathrm{BT}(\xi)$.
- *Let* ξ *be any leaf of* $\mathbb{T}$. Then ξ is marked by a node $x_{\text{indicated}}(\xi) \in T$.

Let T be a BTree. By induction on $\#(T)$, we define a particular control tree candidate for T, called the *control tree* for T, to be denoted $\mathrm{CT}(T)$. The inductive definition of $\mathrm{CT}(T)$ proceeds as follows.

Base case: Suppose $\#(T) = 1$. Thus, T consists of a single node x_0. We then take $\mathrm{CT}(T)$ to consist of a single node ξ_0, marked with the nodes $x_{\text{indicated}}(\xi_0) = x_0$ and $x_{\text{root}}(\xi_0) = x_0$, and also marked with a pointer to the BTree $\mathrm{BT}(\xi_0) = T$. Note that $\mathrm{CT}(T)$ is a control tree candidate for T. Thus we have defined $\mathrm{CT}(T)$ in the base case.

Induction step: Suppose $\#(T) \geq 2$, and suppose we have already defined $\mathrm{CT}(T')$ for any BTree T' with fewer nodes than T. We define $\mathrm{CT}(T)$ as follows.

We apply to the BTree T the algorithm BTREE1, to produce a node $x_{\text{split}}(T) \in T$. We know that $x_{\text{split}}(T)$ is not the root of T; and that

$$\#[\mathrm{Descendants}(x_{\text{split}}(T), T)] \leq \frac{9}{10}\#(T), \text{ and}$$
$$\#[\mathrm{Nondescendants}(x_{\text{split}}(T), T)] \leq \frac{9}{10}\#(T).$$

Let $T_{\text{go}} := \mathrm{Descendants}(x_{\text{split}}(T), T)$ and $T_{\text{stay}} := \mathrm{Nondescendants}(x_{\text{split}}(T), T)$. Then T_{go}, T_{stay} are BTrees, with fewer nodes than T. By induction hypothesis, we have already defined the control trees $\mathrm{CT}(T_{\text{go}})$, $\mathrm{CT}(T_{\text{stay}})$.

We define the tree $\mathbb{T}$ to consist of a root ξ_0, together with the two trees $\mathrm{CT}(T_{\text{go}})$, $\mathrm{CT}(T_{\text{stay}})$, where we take the two children of ξ_0 to be the roots of $\mathrm{CT}(T_{\text{go}})$ and of $\mathrm{CT}(T_{\text{stay}})$. Thus, $\mathbb{T}$ is a binary tree. We mark the nodes of $\mathbb{T}$ to form a control tree candidate for T, as follows:

- We keep the markings of the nodes of $\mathrm{CT}(T_{\text{go}})$ and $\mathrm{CT}(T_{\text{stay}})$, without change.
- We mark the root of $\mathrm{CT}(T_{\text{go}})$ as $\mathrm{gochild}(\xi_0)$, and we mark the root of $\mathrm{CT}(T_{\text{stay}})$ as $\mathrm{staychild}(\xi_0)$.
- We mark the root ξ_0 with the node $x_{\text{split}}(\xi_0) = x_{\text{split}}(T)$.
- We mark the root ξ_0 with a pointer to the tree T, i.e., we take $\mathrm{BT}(\xi_0) = T$ and $x_{\text{root}}(\xi_0) =$ the root of T.

We define $\mathrm{CT}(\mathsf{T})$ to be the marked tree $\mathbb{T}$. This concludes our inductive definition of $\mathrm{CT}(\mathsf{T})$. To show that $\mathrm{CT}(\mathsf{T})$ is a control-tree candidate for T, we just need to establish the following lemma.

Lemma 33. *Let* T *be a BTree. Then* $\mathrm{BT}(\xi)$ *is a sub-tree of* T *for each* $\xi \in \mathrm{CT}(\mathsf{T})$.

Proof. The proof is by induction on $\#(\mathsf{T})$.

If $\#(\mathsf{T}) = 1$ then $\mathrm{BT}(\xi) = \mathsf{T}$ for the single node ξ in $\mathrm{CT}(\mathsf{T})$. The result is immediate.

Suppose that $\#(\mathsf{T}) \geq 2$, and suppose that lemma holds for all trees with fewer nodes than T.

If ξ is the root ξ_0 of $\mathrm{CT}(\mathsf{T})$, then $\mathrm{BT}(\xi) = \mathsf{T}$ by definition, hence the conclusion of the lemma is obvious.

If ξ is not the root of $\mathrm{CT}(\mathsf{T})$ then ξ is a node in either $\mathrm{CT}(\mathsf{T}_{\text{go}})$ or $\mathrm{CT}(\mathsf{T}_{\text{stay}})$.

Thus, by the inductive hypothesis, $\mathrm{BT}(\xi)$ is a sub-tree of either T_{go} or T_{stay}. Since T_{go} and T_{stay} are sub-trees of T, we conclude that $\mathrm{BT}(\xi)$ is a sub-tree of T.

This concludes the proof of the lemma by induction. □

Lemma 34. *The following properties of* $\mathrm{CT}(\mathsf{T})$ *hold.*

(A) *The number of nodes of* $\mathrm{CT}(\mathsf{T})$ *is* $2\#(\mathsf{T}) - 1$.

(B) *Moreover, any descending sequence in* $\mathrm{CT}(\mathsf{T})$ *has length at most* $1+\mathsf{C}\log(\#(\mathsf{T}))$ *for a universal constant* C.

(C) *Finally,*

$$\sum_{\xi \in \mathrm{CT}(\mathsf{T})} \#\left(\mathrm{BT}(\xi)\right) \leq \mathsf{C} \cdot \#(\mathsf{T}) \cdot \{\log_2(\#(\mathsf{T})) + 1\}$$

for a universal constant C.

Proof. We prove (A) by induction on $\#(\mathsf{T})$. If $\#(\mathsf{T}) = 1$, then by definition $\#(\mathrm{CT}(\mathsf{T})) = 1$, so (A) holds in this case.

For the induction step, suppose we know (A) for all trees with fewer nodes than T ($\#(\mathsf{T}) \geq 2$). We establish (A) for T. Indeed, with $x_{\text{split}}(\mathsf{T})$ as in the definition of $\mathrm{CT}(\mathsf{T})$, we know that $\mathrm{CT}(\mathsf{T})$ consists of the root ξ_0, the control tree $\mathrm{CT}(\mathrm{Descendants}(x_{\text{split}}(\mathsf{T}),\mathsf{T}))$, and the control tree $\mathrm{CT}(\mathrm{Nondescendants}(x_{\text{split}}(\mathsf{T}),\mathsf{T}))$. Hence,

$$\begin{aligned}\#\,\mathrm{CT}(\mathsf{T}) = 1 + \#\,\mathrm{CT}(\mathrm{Descendants}(x_{\text{split}}(\mathsf{T}), \mathsf{T}))\\ + \#\,\mathrm{CT}(\mathrm{Nondescendants}(x_{\text{split}}(\mathsf{T}), \mathsf{T})),\end{aligned}$$

whereas

$$\#\mathsf{T} = \#\,\mathrm{Descendants}(x_{\text{split}}(\mathsf{T}), \mathsf{T}) + \#\,\mathrm{Nondescendants}(x_{\text{split}}(\mathsf{T}), \mathsf{T}).$$

Since $\#\,\mathrm{Descendants}(x_{\text{split}}(\mathsf{T}), \mathsf{T})$, $\#\,\mathrm{Nondescendants}(x_{\text{split}}(\mathsf{T}), \mathsf{T})$ are strictly less than $\#(\mathsf{T})$, the induction hypothesis gives

$$\#\,\mathrm{CT}(\mathrm{Descendants}(x_{\text{split}}(\mathsf{T}), \mathsf{T})) = 2 \cdot \#\,\mathrm{Descendants}(x_{\text{split}}(\mathsf{T}), \mathsf{T}) - 1$$

and

$$\#\,\mathrm{CT}(\mathrm{Nondescendants}(x_{\text{split}}(\mathsf{T}), \mathsf{T})) = 2 \cdot \#\,\mathrm{Nondescendants}(x_{\text{split}}(\mathsf{T}), \mathsf{T}) - 1.$$

Adding the above, we find that $\#\,\mathrm{CT}(T) - 1 = 2 \cdot \#T - 2$, proving (A) for the BTree T. This completes our induction and proves (A).

To prove (B), we check that $\#\mathrm{BT}(\xi') \le \frac{9}{10} \cdot \#\,\mathrm{BT}(\xi)$ whenever ξ' is a child of ξ in $\mathrm{CT}(T)$. Indeed, this follows from the definition of $\mathrm{BT}(\xi)$ and the defining property of $x_{\mathrm{split}}(T)$ by an obvious induction on $\#(T)$.

For a descending chain $\xi_0, \xi_1, \xi_2, \dots, \xi_\ell$ in $\mathrm{CT}(T)$, we therefore have

$$1 \le \#\,\mathrm{BT}(\xi_\ell) \le (9/10)^\ell \#\,\mathrm{BT}(\xi_0) \le (9/10)^\ell \cdot \#[T].$$

Thus, $\ell \le \frac{\log(\#T)}{\log(10/9)}$, proving (B).

To prove (C), we prove by induction on $\#(T)$ that

$$(*) \qquad \sum_{\xi \in \mathrm{CT}(T)} \#\,\mathrm{BT}(\xi) \le \#T \cdot \left\{\log_{10/9}(\#T) + 1\right\}.$$

For $\#(T) = 1$, this holds because $\mathrm{BT}(\xi_0) = T$ where ξ_0 is the one and only node of $\mathrm{CT}(T)$.

Assume $(*)$ holds for all BTrees with fewer nodes than T, where T is a given BTree with $\#(T) \ge 2$. Then by our induction hypothesis we have

$$\begin{aligned}(\dagger) \quad \sum_{\xi \in \mathrm{CT}(T)} \#\,\mathrm{BT}(\xi) &= \#\,\mathrm{BT}(\mathrm{root}(\mathrm{CT}(T))) + \sum_{\xi \in \mathrm{CT}(\mathrm{Descendants}(x_{\mathrm{split}}(T),T))} \#\,\mathrm{BT}(\xi) \\ &\quad + \sum_{\xi \in \mathrm{CT}(\mathrm{Nondescendants}(x_{\mathrm{split}}(T),T))} \#\,\mathrm{BT}(\xi) \\ &\le \#(T) + \#\,\mathrm{Descendants}(x_{\mathrm{split}}(T),T) \cdot \left\{1 + \log_{10/9}[\#\,\mathrm{Descendants}(x_{\mathrm{split}}(T),T)]\right\} \\ &\quad + \#\,\mathrm{Nondescendants}(x_{\mathrm{split}}(T),T) \cdot \left\{1 + \log_{10/9}[\#\,\mathrm{Nondescendants}(x_{\mathrm{split}}(T),T)]\right\}.\end{aligned}$$

We know that

$$1 + \log_{10/9}[\#\,\mathrm{Descendants}(x_{\mathrm{split}}(T), T)] \le \log_{10/9} \#T \quad \text{and that}$$
$$1 + \log_{10/9}[\#\,\mathrm{Nondescendants}(x_{\mathrm{split}}(T), T)] \le \log_{10/9} \#T.$$

Hence, $(\dagger)$ yields the estimate

$$\begin{aligned}\sum_{\xi \in \mathrm{CT}(T)} \#\,\mathrm{BT}(\xi) \le \#T + \log_{10/9}(\#T) \cdot \#\,\mathrm{Descendants}(x_{\mathrm{split}}(T), T) \\ + \log_{10/9}(\#T) \cdot \#\,\mathrm{Nondescendants}(x_{\mathrm{split}}(T), T),\end{aligned}$$

thus proving $(*)$. The proof of our lemma is complete. □

ALGORITHM: MAKE CONTROL TREE (DELUXE EDITION)

Given a BTree T, we produce the control tree $\mathrm{CT}(T)$. The work and storage used to do so are at most $C \cdot \#(T) \cdot (1 + \log \#(T))$ for a universal constant C.

Explanation. We simply follow the definition in the obvious way. Where the definition proceeds by induction, the algorithm calls itself recursively. The assertion about the work and storage follows from Lemma 34, assertion (C), and also the bound on the running time of the algorithm BTREE1, which is used as a subroutine. □

We will not use the Deluxe edition explained above, because it uses too much storage.

<u>ALGORITHM: MAKE CONTROL TREE (PAPERBACK EDITION)</u>

Given a BTree T, we produce the tree CT(T) with all its markings *except* for the BTrees BT(ξ) ($\xi \in$ CT(T)). For each $\xi \in$ CT(T) we indicate whether BT(ξ) is a singleton.

The work used to do so is at most $C \cdot \#(T) \cdot (1 + \log \#(T))$, and the storage used is at most $C \cdot \#(T)$. Here, C is a universal constant.

Explanation. We proceed as in the deluxe edition of the algorithm MAKE CONTROL TREE, except that we delete T when we are finished using it.

We spell out the details.

If $\#(T) = 1$, then we take CT(T) to consist of a single node ξ_0, marked with $x_{\text{indicated}}(\xi_0) = x_{\text{root}}(\xi_0) =$ the one and only node of T. We indicate that the BTree BT(ξ_0) is a singleton.

If $\#(T) > 1$, then we execute the algorithm BTREE1 to produce the node $x_{\text{split}}(T)$.

/* In a later variant of this algorithm, we insert code here */

We compute the trees $T' =$ Descendants($x_{\text{split}}(T), T$) with root $x_{\text{split}}(T)$, and $T'' =$ Nondescendants($x_{\text{split}}(T), T$) with root = root(T).

To produce the trees T', T'' efficiently, we can simply erase the marking indicating $x_{\text{split}}(T)$ as a child of its parent in T, and then produce pointers to the roots of T', T''. This destroys the tree T after we no longer need it.

Recursively, we apply the paperback edition of MAKE CONTROL TREE to T' and T''. Thus, we obtain CT(T') and CT(T'') with all their markings, except for the markings BT(ξ') ($\xi' \in$ CT(T')) and BT(ξ'') ($\xi'' \in$ CT(T'')). These latter markings have not been computed (or rather, they were computed and then deleted).

The tree CT(T) then consists of the two trees CT(T') and CT(T''), together with a root ξ_0. The children of ξ_0 are the roots of the two trees CT(T'), CT(T''). We mark the root of CT(T') as gochild(ξ_0), and we mark the root of CT(T'') as staychild(ξ_0). Also, we mark the root ξ_0 of CT(T) with the node $x_{\text{split}}(\xi_0) = x_{\text{split}}(T) \in T$.

We mark the root ξ_0 of CT(T) with the node $x_{\text{root}}(\xi_0) =$ the root of T. (Recall that BT(ξ_0) $= T$.) We indicate that the BTree BT(ξ_0) is not a singleton. We do not mark the root ξ_0 with anything else to tell us what the tree T was before we destroyed it.

This completes our description of the algorithm.

Let us check how much time and space are used.

Let Time(T) be the number of operations needed to execute the paperback algorithm for the tree T. Recalling that the algorithm BTREE1 uses work $C\#(T)$ to produce $x_{\text{split}}(T)$, we see that

$$\text{Time}(T) \leq C\#(T) + \text{Time}(T') + \text{Time}(T''),$$

and we recall that $\#(T'), \#(T'') \leq \frac{9}{10}\#(T)$ and that $\#(T') + \#(T'') = \#(T)$. Hence, it follows by induction on $\#(T)$ that

$$\text{Time}(T) \leq C\#(T) \cdot [1 + \log_{10/9} \#T].$$

Thus, the work required to execute our paperback algorithm is as promised.

Next, we study the storage used by our paperback algorithm, which we denote by $\mathfrak{S}(T)$.

Since we erase T, we see easily that

$$\mathfrak{S}(T) \leq \max\{C \cdot \#(T), \mathfrak{S}(T') + \mathfrak{S}(T'') + C\},$$

i.e.,

$$[\mathfrak{S}(T) + C] \leq \max\{C'\#(T), [\mathfrak{S}(T') + C] + [\mathfrak{S}(T'') + C].$$

Since $\#(T) = \#(T') + \#(T'')$, it follows by induction on $\#(T)$ that

$$[\mathfrak{S}(T) + C] \leq C''\#(T).$$

This proves that the storage used by our paperback algorithm is as promised. □

ALGORITHM: MAKE CONTROL TREE (HYBRID VERSION)

Given an ADTree T, with each node $x \in T$ marked by functionals $\mu_1^x, \dots, \mu_D^x : \mathbb{R}^D \to \mathbb{R}$, we produce the control tree CT(T) with all its markings *except* for the trees BT(ξ) ($\xi \in$ CT(T)). For each node $\xi \in$ CT(T), we produce functionals $\mu_1^\xi, \dots, \mu_D^\xi : \mathbb{R}^D \to \mathbb{R}$ such that

$$\sum_{x \in BT(\xi)} \sum_{i=1}^{D} |\mu_i^x(v)|^p \quad \text{and} \quad \sum_{i=1}^{D} |\mu_i^\xi(v)|^p$$

differ by at most a factor $C(D, p)$ for any $v \in \mathbb{R}^D$. (This makes sense because BT(ξ) is a sub-tree of T for each ξ.)

We mark each node $\xi \in$ CT(T) with such functionals $\mu_1^\xi, \dots, \mu_D^\xi$.

We mark each node $\xi \in$ CT(T) to indicate whether BT(ξ) is a singleton.

The work and storage needed to execute this algorithm are at most $C\#(T) \cdot [\log(\#T) + 1]$ and $C\#(T)$, respectively.

Explanation. We proceed as in the explanation of the paperback edition of MAKE CONTROL TREE, with the following changes.

If $\#(T) = 1$, then the tree CT(T) contains only the root node ξ_0. We set $\mu_i^{\xi_0} = \mu_i^x$ for $i = 1, \dots, D$, for the one and only one node $x \in T$.

If $\#(T) \geq 2$, then we proceed as follows. Where we wrote

/* In a later variant of this algorithm, we insert code here */

we now insert a call to COMPRESS NORMS (Section 2.8). Thus, with work and storage at most $C\#(T)$, we produce functionals $\mu_1^*, \dots, \mu_D^* : \mathbb{R}^D \to \mathbb{R}$ such that

$$\sum_{x\in T}\sum_{i=1}^{D}|\mu_i^x(v)|^p \quad \text{and} \quad \sum_{i=1}^{D}|\mu_i^*(v)|^p$$

differ by at most a factor $C(D,p)$ for any $v \in \mathbb{R}^D$.

Instead of recursively applying the paperback edition of MAKE CONTROL TREE, we now recursively apply the hybrid version.

Just before the sentence "This completes the description of the algorithm" we set $\mu_i^{\xi_0} = \mu_i^*$ for $i = 1, \dots, D$.

Since $\mathrm{BT}(\xi_0) = T$, our $\mu_i^{\xi_0}$ behave as we ask.

The work and storage needed to execute this algorithm are as promised. □

4.1.4. Encapsulations. Let T be a DTree, and let $\mathrm{CT}(T)$ be its control tree. Recall that each node $\xi \in \mathrm{CT}(T)$ is marked with a BTree $\mathrm{BT}(\xi)$ consisting of nodes of T. Also, each node $x \in T$ is marked with a dyadic cuboid Q_x.

Let Q be a dyadic cuboid. An *encapsulation* of Q is a set S of nodes of $\mathrm{CT}(T)$, such that $\{x \in T : Q_x \subset Q\}$ is the disjoint union of the sets $\mathrm{BT}(\xi)$ as ξ varies over S.

ALGORITHM: ENCAPSULATE

Let T be a DTree with N nodes. After $CN(1+\log N)$ one-time work in space CN, we can answer queries as follows:

A query consists of a dyadic cuboid Q.

The response to a query Q is an encapsulation S of Q, consisting of at most $C + C\log N$ nodes of $\mathrm{CT}(T)$.

The work and storage used to answer a query are at most $C + C\log N$. Here, C denotes a constant depending only on the dimension n.

Explanation. Suppose T is not a singleton.

Let $x_{\text{split}}(T)$ be the node produced by the algorithm BTREE1.

Write $T' = \mathrm{Descendants}(x_{\text{split}}(T), T)$ and $T'' = \mathrm{Nondescendants}(x_{\text{split}}(T), T)$.

We ask: For which nodes $x \in T$ do we have $Q_x \subset Q$? To answer this question, we compare the dyadic cuboids Q and $Q_{x_{\text{split}}(T)}$. There are three cases:

Case 1: $Q \subset Q_{x_{\text{split}}(T)}$. In this case, we never have $Q_x \subset Q$ for an $x \in T''$.

Hence, in this case, $\{x \in T : Q_x \subset Q\} = \{x \in T' : Q_x \subset Q\}$. Thus, we have reduced matters from T to T'.

Case 2: $Q_{x_{\text{split}}(T)} \subsetneq Q$. In this case, all $x \in T'$ satisfy $Q_x \subset Q$.

Therefore, in this case,

$$\begin{aligned}\{x \in T : Q_x \subset Q\} &= \{x \in T'' : Q_x \subset Q\} \cup T' \\ &= \{x \in T'' : Q_x \subset Q\} \cup \mathrm{BT}(\text{gochild}(\text{root of } \mathrm{CT}(T))).\end{aligned}$$

Thus, we have reduced matters from T to T''.

Case 3: $Q_{x_{\text{split}}(T)} \cap Q = \emptyset$. In this case, no $x \in T'$ satisfy $Q_x \subset Q$, hence $\{x \in T : Q_x \subset Q\} = \{x \in T'' : Q_x \subset Q\}$. Again, we have reduced matters from T to T''.

Cases 1, 2, 3 are the only possibilities, since Q and $Q_{x_{\text{split}}(T)}$ are dyadic cuboids.

Thanks to the above remarks, the following procedure produces an encapsulation, when applied to $\widehat{\xi} = \text{root}(\text{CT}(T))$.

- One-time work: Paperback edition of Make control tree.
- Procedure Encap$(Q, \widehat{\xi})$:
  /* Produces an encapsulation S of Q for the BTree $\text{BT}(\widehat{\xi})$. */
  - If $\widehat{\xi}$ is a leaf of $\text{CT}(T)$, then Encap$(Q, \widehat{\xi})$ returns $\{\widehat{\xi}\}$ if $Q_{x_{\text{indicated}}(\widehat{\xi})} \subset Q$, and returns $\emptyset$ otherwise.
  - If $\widehat{\xi}$ is an internal node of $\text{CT}(T)$, then let $\widehat{x} = x_{\text{split}}(\widehat{\xi})$, $\xi' = \text{gochild}(\widehat{\xi})$, $\xi'' = \text{staychild}(\widehat{\xi})$.
    - ∗ *If* $Q \subset Q_{\widehat{x}}$, then return the set produced by (recursively) executing Encap(Q, ξ').
    - ∗ *If* $Q_{\widehat{x}} \subsetneq Q$, then return the union of $\{\xi'\}$ with the set produced by (recursively) executing Encap(Q, ξ'').
    - ∗ *If* $Q_{\widehat{x}} \cap Q = \emptyset$, then return the set produced by (recursively) executing Encap(Q, ξ'').

Note that the one-time work here is simply that of the paperback edition of Make control tree; hence, we perform one-time work at most $CN(1 + \log N)$ in space CN.

Regarding the query work, note that the "depth" of the recursion (i.e., the number of recursive calls to Encap$(\cdot, \cdot)$) is at most $1 + C \log N$, since $\#(\text{BT}(\xi))$ decreases by at least a factor of $\frac{9}{10}$ each time we pass from a node ξ to gochild(ξ) or staychild(ξ).

Therefore, the work and storage used to answer a query are at most $C + C \log N$. In particular, $\#(S) \leq C + C \log N$, since it takes work at most $C + C \log N$ to write down S.

This completes our explanation of the algorithm Encapsulate. □

<u>Algorithm: ADProcess</u>

Given an ADTree T, (recall that each node $x \in T$ is marked with a dyadic cuboid Q_x and with linear functionals $\mu_1^x, \dots, \mu_D^x : \mathbb{R}^D \to \mathbb{R}$) with N nodes, we perform one-time work at most $CN(1 + \log N)$ in space CN, after which we can answer queries as follows:

A query consists of a dyadic cuboid Q.

The response to a query consists of linear functionals $\mu_1^Q, \dots, \mu_D^Q : \mathbb{R}^D \to \mathbb{R}$ such that

$$\sum_{i=1}^{D} |\mu_i^Q(\nu)|^p \quad \text{and} \quad \sum_{x \in T,\ Q_x \subset Q} \ \sum_{i=1}^{D} |\mu_i^x(\nu)|^p$$

differ by at most a factor C for any $\nu \in \mathbb{R}^D$.

The work and storage needed to respond to a query are at most $C \cdot (\log N + 1)$. Here, C depends only on p, n, D (n = dimension of the cuboids).

Explanation. We perform the one-time work for the algorithm ENCAPSULATE, and we perform the hybrid version of the algorithm MAKE CONTROL TREE. Thus, we produce the control tree $\mathrm{CT}(T)$; each node ξ is marked with linear functionals $\mu_1^\xi, \dots, \mu_D^\xi : \mathbb{R}^D \to \mathbb{R}$ such that

$$\sum_{i=1}^{D} |\mu_i^\xi(\nu)|^p \quad \text{and} \quad \sum_{x \in \mathrm{BT}(\xi)} \sum_{i=1}^{D} |\mu_i^x(\nu)|^p$$

differ by at most a factor of C for any $\nu \in \mathbb{R}^D$.

Moreover, thanks to the query algorithm in ENCAPSULATE, we can answer queries as follows.

A query consists of a dyadic cuboid Q. The response to a query Q consists of a set S of at most $C + C \log N$ nodes in $\mathrm{CT}(T)$ such that $\{x \in T : Q_x \subset Q\}$ is the disjoint union over $\xi \in S$ of $\mathrm{BT}(\xi) \subset T$.

Given $\nu \in \mathbb{R}^D$, we have therefore

$$\sum_{x \in T : Q_x \subset Q} \sum_{i=1}^{D} |\mu_i^x(\nu)|^p = \sum_{\xi \in S} \left\{ \sum_{x \in \mathrm{BT}(\xi)} \sum_{i=1}^{D} |\mu_i^x(\nu)|^p \right\}$$

which differs by at most a factor of C from

$$(*) \qquad \sum_{\xi \in S} \sum_{i=1}^{D} |\mu_i^\xi(\nu)|^p.$$

Applying the algorithm COMPRESS NORMS (Section 2.8) to the linear functionals μ_i^ξ ($\xi \in S$; $i = 1, \dots, D$), we obtain linear functionals $\mu_1^Q, \dots, \mu_D^Q$, such that for any $\nu \in \mathbb{R}^D$, the quantity $(*)$ differs from $\sum_{i=1}^{D} |\mu_i^Q(\nu)|^p$ by at most a factor of C. Thus, $\mu_1^Q, \dots, \mu_D^Q$ satisfy the desired condition.

The one-time work of the above algorithm is at most $CN(1 + \log N)$, in space CN, thanks to our estimates of the one-time work of ENCAPSULATE, and the work and storage of the hybrid version of MAKE CONTROL TREE.

The query work of our algorithm is at most that of the algorithm ENCAPSULATE, together with the work of applying COMPRESS NORMS to the μ_i^ξ ($\xi \in S$; $i = 1, \dots, D$).

Since $\#(S) \le C(\log N + 1)$, the work of applying COMPRESS NORMS is at most $C(\log N + 1)$. The query work of ENCAPSULATE is also at most $C(\log N + 1)$.

Therefore, the total query work of our present algorithm is as promised.

This completes our explanation of the algorithm ADPROCESS. □

4.1.5. Making a tree from a list of cuboids. Fix a dimension n, and let $Q_1, \dots, Q_N$ be a sequence of distinct dyadic cuboids in $\mathbb{R}^n$. We assume that

$$(4.3) \qquad Q_1 < Q_2 < \cdots < Q_N.$$

(See Section 4.1.1 for the definition of the order relation $<$.)

ALGORITHM: MAKE FOREST

Given $1 \le i_{\text{start}} \le i_{\text{end}} \le N$, we produce the following:

- A sorted list $i_1 < i_2 < \cdots < i_L$ consisting precisely of all the i such that $i_{\text{start}} \le i \le i_{\text{end}}$, and such that there exists no i' with $i_{\text{start}} \le i' \le i_{\text{end}}$ and $Q_i \subsetneq Q_{i'}$. The list is computed as a linked list: We do not compute an array $i_1, i_2, \dots, i_L$. Rather, we compute the initial entry i_1, and we mark each i_ν with a pointer to its successor $i_{\nu+1}$ $(1 \le \nu < L)$. The last entry i_L is marked with a NULL pointer.
- For each i $(i_{\text{start}} \le i \le i_{\text{end}})$ we produce a pointer which is NULL if i appears in the list $i_1, \dots, i_L$, and otherwise indicates i'' such that among all i' such that $i_{\text{start}} \le i' \le i_{\text{end}}$ and $Q_i \subsetneq Q_{i'}$, the cuboid $Q_{i''}$ is the smallest with respect to inclusion.

To do so requires at most $C \cdot (i_{\text{end}} - i_{\text{start}} + 1) \cdot (1 + \log N)$ work and at most $C \cdot (i_{\text{end}} - i_{\text{start}} + 1)$ storage, aside from that used to hold the list (4.3). Here, C depends only on the dimension n.

Explanation. We proceed recursively, by induction on $i_{\text{end}} - i_{\text{start}}$.

In the base case: $i_{\text{end}} = i_{\text{start}}$. Then our task is trivial, and it takes work and storage at most C.

The induction step: Suppose $i_{\text{end}} > i_{\text{start}}$. Then, by Corollary 32, we have one of the following cases.

Case 1: $Q_{i_{\text{start}}+1}, \dots, Q_{i_{\text{end}}} \subset Q_{i_{\text{start}}}$.

Case 2: $Q_{i_{\text{start}}+1}, \dots, Q_{i_{\text{end}}}$ are all disjoint from $Q_{i_{\text{start}}}$.

Case 3: For some j $(i_{\text{start}} + 1 \le j < i_{\text{end}})$, we have $Q_{i_{\text{start}}+1}, \dots, Q_j \subset Q_{i_{\text{start}}}$, and $Q_{j+1}, \dots, Q_{i_{\text{end}}}$ are disjoint from $Q_{i_{\text{start}}}$.

We can determine which of these cases holds, simply by checking $Q_{i_{\text{start}}+1} \cap Q_{i_{\text{start}}}$ and $Q_{i_{\text{end}}} \cap Q_{i_{\text{start}}}$. Moreover, if Case 3 holds, then we can find j by doing a binary search. This requires work at most $C \cdot (1 + \log N)$ and storage at most C, aside from the storage used to hold the given cuboids (4.3).

We describe how to proceed in Case 3. Later, we explain the modifications needed for Cases 1 and 2.

Suppose we are in Case 3, with j known. Recursively, we apply the algorithm MAKE FOREST to indices $i_{\text{start}} + 1$ and j (in place of i_{start} and i_{end}) and also to indices $j + 1$ and i_{end} (in place of i_{start} and i_{end}).

Thus we obtain the following:

(4.4) $\tilde{i}_1 < \tilde{i}_2 < \cdots < \tilde{i}_{\tilde{L}}$, a linked list of all $\tilde{i} \in \{i_{\text{start}} + 1, \dots, j\}$ such that $Q_{\tilde{i}}$ is maximal (under inclusion) among $Q_{i_{\text{start}}+1}, \dots, Q_j$.

(4.5) For each $\tilde{i} \in \{i_{\text{start}} + 1, \dots, j\}$, either a NULL pointer indicating that $Q_{\tilde{i}}$ is maximal as in (4.4), or else a pointer to the $\tilde{i}_+ \in \{i_{\text{start}} + 1, \dots, j\}$ such that $Q_{\tilde{i}_+} \supsetneq Q_{\tilde{i}}$ with $Q_{\tilde{i}_+}$ as small as possible under inclusion.

(4.6) $i_1^\# < i_2^\# < \cdots < i_{L^\#}^\#$, a linked list of all $i^\# \in \{j+1, \dots, i_{\text{end}}\}$ such that $Q_{i^\#}$ is maximal (under inclusion) among $Q_{j+1}, \dots, Q_{i_{\text{end}}}$.

(4.7) For each $i^\# \in \{j+1, \dots, i_{\text{end}}\}$, either a NULL pointer indicating that $i^\#$ appears in the list (4.6), or else a pointer to the $i_+^\# \in \{j+1, \dots, i_{\text{end}}\}$ such that $Q_{i_+^\#} \supsetneq Q_{i^\#}$ with $Q_{i_+^\#}$ as small as possible under inclusion.

We now produce the desired output for i_{start}, i_{end}.

- Our linked list $i_1 < i_2 < \cdots < i_L$ consists of the list $i_1^\# < i_2^\# < \cdots < i_{L^\#}^\#$, with $i_0^\# := i_{\text{start}}$ added to the beginning of the list. (Since the lists in question are implemented here as linked lists, it takes work at most C to add $i_0^\#$ to the list.)

- Our pointers are as follows.

  For $i \in \{j+1, \dots, i_{\text{end}}\}$, the pointers are precisely those produced in (4.7).

  For $i \in \{i_{\text{start}}+1, \dots, j\}$, we take the pointers produced in (4.5). However, for each $\tilde{i}_\ell$ in the linked list (4.4), we set the pointer associated to $\tilde{i}_\ell$ (which was NULL in (4.5)) so that it indicates i_{start}.

  For $i = i_{\text{start}}$, we take a NULL pointer.

The work needed to implement the above bullet points is at most

$$C + C \cdot \#\{\text{Maximal cuboids under inclusion among } Q_{i_{\text{start}}+1}, \dots, Q_j\}.$$

This concludes our description of the algorithm in Case 3. It produces the desired information thanks to the inclusions and disjointness conditions that hold in Case 3.

Cases 1 and 2 are similar to Case 3, but easier.

In Case 1, there are no $\{j+1, \dots, i_{\text{end}}\}$ to deal with, so we omit all steps relevant to $\{j+1, \dots, i_{\text{end}}\}$.

Similarly, in Case 2, there are no $\{i_{\text{start}}+1, \dots, j\}$ to deal with, so we omit all steps relevant to $\{i_{\text{start}}+1, \dots, j\}$.

Thus, in all cases, our recursive algorithm works as promised, except that we have not yet estimated the work and storage needed to carry it out.

Regarding the work, which we call $\mathfrak{W}(i_{\text{start}}, i_{\text{end}})$, we note that (in Case 3) we have

$$\begin{aligned}\mathfrak{W}(i_{\text{start}}, i_{\text{end}}) &\leq C \cdot (1 + \log N) + \mathfrak{W}(i_{\text{start}}+1, j) + \mathfrak{W}(j+1, i_{\text{end}}) \\ &\quad + C \cdot \#\big\{\text{Maximal } Q_i \text{ (under inclusion) among } Q_{i_{\text{start}}+1}, \dots, Q_j\big\}.\end{aligned} \tag{4.8}$$

In Case 1 we have instead

$$\begin{aligned}\mathfrak{W}(i_{\text{start}}, i_{\text{end}}) &\leq C \cdot (1 + \log N) + \mathfrak{W}(i_{\text{start}}+1, i_{\text{end}}) \\ &\quad + C \cdot \#\big\{\text{Maximal } Q_i \text{ (under inclusion) among } Q_{i_{\text{start}}+1}, \dots, Q_{i_{\text{end}}}\big\}.\end{aligned} \tag{4.9}$$

In Case 2 we have

$$\mathfrak{W}(i_{\text{start}}, i_{\text{end}}) \leq C \cdot (1 + \log N) + \mathfrak{W}(i_{\text{start}} + 1, i_{\text{end}}) \tag{4.10}$$

since there are no pointers to modify in Case 2.

To analyze (4.8),(4.9),(4.10), we introduce $\mathrm{NR}(i_{\text{start}}, i_{\text{end}})$:= the number of cuboids Q_i among $Q_{i_{\text{start}}}, \dots, Q_{i_{\text{end}}}$ that are not contained in any other $Q_{i'}$ among $Q_{i_{\text{start}}}, \dots, Q_{i_{\text{end}}}$. ("NR" stands for "Number of Roots".)

Then (4.8),(4.9),(4.10) together with the known inclusions that follow from Corollary 32 tell us the following.

In Case 3, with j as given in that case, we have

$$\begin{aligned}\mathfrak{W}(i_{\text{start}}, i_{\text{end}}) \leq{}& C \cdot (1 + \log N) + \mathfrak{W}(i_{\text{start}} + 1, j) \\ &+ \mathfrak{W}(j + 1, i_{\text{end}}) + C \cdot \mathrm{NR}(i_{\text{start}} + 1, j)\end{aligned} \tag{4.11}$$

and

$$\mathrm{NR}(i_{\text{start}}, i_{\text{end}}) = 1 + \mathrm{NR}(j + 1, i_{\text{end}}). \tag{4.12}$$

In Case 1, we have instead

$$\mathfrak{W}(i_{\text{start}}, i_{\text{end}}) \leq C \cdot (1 + \log N) + \mathfrak{W}(i_{\text{start}} + 1, i_{\text{end}}) + C \cdot \mathrm{NR}(i_{\text{start}} + 1, i_{\text{end}}) \tag{4.13}$$

and

$$\mathrm{NR}(i_{\text{start}}, i_{\text{end}}) = 1. \tag{4.14}$$

In Case 2, we have

$$\mathfrak{W}(i_{\text{start}}, i_{\text{end}}) \leq C \cdot (1 + \log N) + \mathfrak{W}(i_{\text{start}} + 1, i_{\text{end}}) \tag{4.15}$$

and

$$\mathrm{NR}(i_{\text{start}}, i_{\text{end}}) = 1 + \mathrm{NR}(i_{\text{start}} + 1, i_{\text{end}}). \tag{4.16}$$

Consequently, in Case 3 we have

$$\begin{aligned}\left[\mathfrak{W}(i_{\text{start}}, i_{\text{end}}) + \underline{C}' \mathrm{NR}(i_{\text{start}}, i_{\text{end}})\right] \leq{}& \underline{C}'' \cdot (1 + \log N) \\ &+ \left[\mathfrak{W}(i_{\text{start}} + 1, j) + \underline{C}' \mathrm{NR}(i_{\text{start}} + 1, j)\right] \\ &+ \left[\mathfrak{W}(j + 1, i_{\text{end}}) + \underline{C}' \mathrm{NR}(j + 1, i_{\text{end}})\right].\end{aligned} \tag{4.17}$$

(Here, we pick $\underline{C}'$ big, and then pick $\underline{C}''$ much bigger.)

In Cases 1 and 2, we have instead

$$\begin{aligned}\left[\mathfrak{W}(i_{\text{start}}, i_{\text{end}}) + \underline{C}' \mathrm{NR}(i_{\text{start}}, i_{\text{end}})\right] \leq{}& \underline{C}'' \cdot (1 + \log N) \\ &+ \left[\mathfrak{W}(i_{\text{start}} + 1, i_{\text{end}}) + \underline{C}' \mathrm{NR}(i_{\text{start}} + 1, i_{\text{end}})\right].\end{aligned} \tag{4.18}$$

Also, when $\mathfrak{i}_{\text{start}} = \mathfrak{i}_{\text{end}}$ we have

$$\mathfrak{W}(\mathfrak{i}_{\text{start}}, \mathfrak{i}_{\text{end}}) + \underline{C}' \,\mathrm{NR}(\mathfrak{i}_{\text{start}}, \mathfrak{i}_{\text{end}}) \le \underline{C}''. \tag{4.19}$$

Thanks to (4.17)$\cdots$(4.19), induction on $\mathfrak{i}_{\text{end}} - \mathfrak{i}_{\text{start}}$ yields the estimate

$$\mathfrak{W}(\mathfrak{i}_{\text{start}}, \mathfrak{i}_{\text{end}}) + \underline{C}' \,\mathrm{NR}(\mathfrak{i}_{\text{start}}, \mathfrak{i}_{\text{end}}) \le \underline{C}''' \cdot (1 + \log N) \cdot (\mathfrak{i}_{\text{end}} - \mathfrak{i}_{\text{start}} + 1).$$

In particular,

$$\mathfrak{W}(\mathfrak{i}_{\text{start}}, \mathfrak{i}_{\text{end}}) \le C \cdot (1 + \log N) \cdot (\mathfrak{i}_{\text{end}} - \mathfrak{i}_{\text{start}} + 1)$$

as promised in the statement of MAKE FOREST.

Next, we analyze the storage needed to execute MAKE FOREST.

We implement the pointers as global data (see the second bullet point in the statement of the algorithm); this requires space at most $C \cdot (\mathfrak{i}_{\text{end}} - \mathfrak{i}_{\text{start}} + 1)$. Let $\mathfrak{S}(\mathfrak{i}_{\text{start}}, \mathfrak{i}_{\text{end}})$ be the space in which we can execute our algorithm MAKE FOREST$(\mathfrak{i}_{\text{start}}, \mathfrak{i}_{\text{end}})$, not counting the space needed for the pointers, but including the space needed to compute and display the linked list; see the first bullet point in the statement of the algorithm.

In Case 3, we see that

$$\mathfrak{S}(\mathfrak{i}_{\text{start}}, \mathfrak{i}_{\text{end}}) \le C + \mathfrak{S}(\mathfrak{i}_{\text{start}} + 1, \mathfrak{j}) + \mathfrak{S}(\mathfrak{j}, \mathfrak{i}_{\text{end}}).$$

In Case 1 and in Case 2 we have instead

$$\mathfrak{S}(\mathfrak{i}_{\text{start}}, \mathfrak{i}_{\text{end}}) \le C + \mathfrak{S}(\mathfrak{i}_{\text{start}} + 1, \mathfrak{i}_{\text{end}}).$$

Since also $\mathfrak{S}(\mathfrak{i}_{\text{start}}, \mathfrak{i}_{\text{start}}) \le C$, it follows by induction that $\mathfrak{S}(\mathfrak{i}_{\text{start}}, \mathfrak{i}_{\text{end}}) \le C(\mathfrak{i}_{\text{end}} - \mathfrak{i}_{\text{start}} + 1)$.

Thus, the storage used in executing MAKE FOREST$(\mathfrak{i}_{\text{start}}, \mathfrak{i}_{\text{end}})$ is as promised.

This completes our explanation of that algorithm. □

<u>ALGORITHM: FILL IN GAPS</u>

Suppose we are given a cuboid $\widehat{Q}$ and a list $Q_{\mathfrak{i}_{\text{start}}}, \dots, Q_{\mathfrak{i}_{\text{end}}}$ of pairwise disjoint cuboids, sorted so that $Q_{\mathfrak{i}_{\text{start}}} < Q_{\mathfrak{i}_{\text{start}}+1} < \cdots < Q_{\mathfrak{i}_{\text{end}}}$. Assume that each $Q_\mathfrak{i} \subset \widehat{Q}$.

We produce a DTree T consisting of cuboids, with root $\widehat{Q}$, and with leaves $Q_{\mathfrak{i}_{\text{start}}}, \dots, Q_{\mathfrak{i}_{\text{end}}}$ (and with no other leaves). Each node of T is either a leaf, the parent of a leaf, or has precisely two children.

The work and storage used to do so are at most $C \cdot (\mathfrak{i}_{\text{end}} - \mathfrak{i}_{\text{start}} + 1) \cdot \log(\mathfrak{i}_{\text{end}} - \mathfrak{i}_{\text{start}} + 2)$ and $C \cdot (\mathfrak{i}_{\text{end}} - \mathfrak{i}_{\text{start}} + 1)$, respectively. Here, C depends only on the dimension n.

Explanation. If any $Q_\mathfrak{i} = \widehat{Q}$, then there is only one $Q_\mathfrak{i}$, so we take our DTree to consist only of $\widehat{Q}$. Suppose otherwise.

If $\mathfrak{i}_{\text{end}} - \mathfrak{i}_{\text{start}} + 1 \le 2$, then we can take our DTree to have root $\widehat{Q}$, and take $\widehat{Q}$ to have children $Q_{\mathfrak{i}_{\text{start}}}, \dots, Q_{\mathfrak{i}_{\text{end}}}$. Thus, our present algorithm is trivial in this case.

Suppose instead $\mathfrak{i}_{\text{end}} - \mathfrak{i}_{\text{start}} + 1 \ge 3$. We take $Q^{\#}$ to be the least dyadic cuboid (under inclusion) that contains $Q_{\mathfrak{i}_{\text{start}}}$ and $Q_{\mathfrak{i}_{\text{end}}}$. Let Q', Q'' be, respectively, the lesser and greater child of $Q^{\#}$ (as dyadic cuboids).

Since $Q_{\mathfrak{i}_{\text{start}}} < \cdots < Q_{\mathfrak{i}_{\text{end}}}$ and the $Q_\mathfrak{j}$ are pairwise disjoint, it follows that

- $Q^{\#} \subset \widehat{Q}$.
- $Q_i \subset Q^{\#}$ for each $i = i_{\text{start}}, \dots, i_{\text{end}}$.
- There exists j ($i_{\text{start}} \le j < i_{\text{end}}$) such that

$$Q_{i_{\text{start}}}, \dots, Q_j \subset Q' \quad \text{and} \quad Q_{j+1}, \dots, Q_{i_{\text{end}}} \subset Q''.$$

(These properties are obvious for dyadic cuboids, since they are obvious for dyadic intervals; see our discussion of the 1-1 correspondence Ψ in Section 4.1.1.)

We can recursively apply the present algorithm to produce DTrees T',T'' with roots Q',Q'', respectively. The leaves of T' are precisely $Q_{i_{\text{start}}}, \dots, Q_j$; and the leaves of T'' are precisely $Q_{j+1}, \dots, Q_{i_{\text{end}}}$.

Our DTree T will consist of a root $\widehat{Q}$, together with T' and T''. The two children of $\widehat{Q}$ will be Q' and Q''. This T is obviously as promised.

Aside from recursively calling on itself, the above algorithm uses work at most $C \log(\#(T) + 1)$ and storage at most C (not counting the storage used to hold $Q_{i_{\text{start}}}, \dots, Q_{i_{\text{end}}}$). The factor of $\log(\#(T)+1)$ comes from a binary search used to find j.

Therefore, altogether, our algorithm uses work and storage at most $C \cdot \#(T) \log(\#(T)+1)$ and $C \cdot \#(T)$, respectively, where T is the DTree arising from the algorithm. Since the leaves of T are precisely $Q_{i_{\text{start}}}, \dots, Q_{i_{\text{end}}}$, and since each node of T (other than the leaves and parents of leaves) has 2 children, we see that

$$\#(T) \le C \cdot \#(\text{leaves of } T) = C \cdot (i_{\text{end}} - i_{\text{start}} + 1).$$

Thus, the work and storage of the algorithm are as promised. $\square$

<u>Algorithm: Make DTree</u>

Suppose we are given a list of distinct cuboids $Q_1, \dots, Q_N$. With work $\le CN(1 + \log N)$ in space CN we produce a DTree T with the following properties.

- Each of the Q_j is a node of T.
- Each node of T is marked as *original* if and only if it is one of the Q_j.
- Furthermore, we mark each node Q of T to indicate either the smallest (under inclusion) original node containing Q, or else to indicate that no such original node exists.
- The number of nodes of T is at most CN.

Explanation. We pick a cuboid Q^{00} that strictly contains the cuboids $Q_1, \dots, Q_N$.

Applying the algorithm Make forest, we make a tree $T^{(1)}$ with nodes $Q^{00}, Q_1, \dots, Q_N$, such that Q is a descendant of Q' in $T^{(1)}$ if and only if $Q \subset Q'$.

Note that Q^{00} is the root of $T^{(1)}$.

Recall that each non-root node in $T^{(1)}$ is marked with a pointer to its parent. We mark each internal node in $T^{(1)}$ with pointers to its children, using an obvious algorithm. We also mark each non-root node in $T^{(1)}$ to indicate that it is original.

By repeatedly applying the algorithm Fill in gaps to a node Q of $T^{(1)}$ together with a list of its children (sorted under $<$), we imbed the tree $T^{(1)}$ in a DTree $T^{(2)}$

that has at most CN nodes. The nodes in $T^{(1)}$ retain their markings in $T^{(2)}$. At each stage, we indicate the smallest original node containing each of the newly generated nodes (if such an original node exists). Indeed, if $Q = Q^{00}$, then we mark each of the new nodes Q' generated by FILL IN GAPS to indicate that Q' is not contained in any original node. If $Q \neq Q^{00}$, then we mark each of the new nodes Q' generated by FILL IN GAPS to indicate that the smallest original node containing Q' is the node Q.

This completes the construction of the marked DTree $T^{(2)}$.

One can easily check that the algorithm satisfies the desired work and storage bounds. □

<u>ALGORITHM: COMPUTE NORMS FROM MARKED CUBOIDS</u>

Suppose we are given a list $Q_1, \dots, Q_N$ of distinct dyadic cuboids in $\mathbb{R}^n$, with each cuboid Q_i marked with linear functionals $\mu_1^{Q_i}, \dots, \mu_{L_i}^{Q_i} : \mathbb{R}^D \to \mathbb{R}$. Let $\widehat{N} = \sum_{i=1}^N (L_i + 1)$.

Given $1 < p < \infty$, we perform one-time work at most $C\widehat{N}(1 + \log \widehat{N})$ in space $C\widehat{N}$, after which we can answer queries as follows:

A query consists of a dyadic cuboid Q in $\mathbb{R}^n$.

The response to the query Q is a list of linear functionals $\widehat{\mu}_1^Q, \dots, \widehat{\mu}_D^Q : \mathbb{R}^D \to \mathbb{R}$, for which we guarantee the estimate

$$c \cdot \sum_{j=1}^{D} |\widehat{\mu}_j^Q(v)|^p \le \sum_{Q_i \subset Q} \sum_{j=1}^{L_i} |\mu_j^{Q_i}(v)|^p \le C \sum_{j=1}^{D} |\widehat{\mu}_j^Q(v)|^p \quad \text{for all } v \in \mathbb{R}^D.$$

The work and storage used to answer a query are at most $C \cdot (1 + \log N)$. Here, c and C depend only on n, p, and D.

Explanation. For each $i = 1, \dots, N$, we apply COMPRESS NORMS (see Section 2.8) to the functionals $\mu_j^{Q_i}$ $(1 \le j \le L_i)$. We obtain linear functionals $\overline{\mu}_1^{Q_i}, \dots, \overline{\mu}_D^{Q_i} : \mathbb{R}^D \to \mathbb{R}$ such that

$$c \cdot \sum_{j=1}^{D} |\overline{\mu}_j^{Q_i}(v)|^p \le \sum_{j=1}^{L_i} |\mu_j^{Q_i}(v)|^p \le C \cdot \sum_{j=1}^{D} |\overline{\mu}_j^{Q_i}(v)|^p \quad \text{for all } v \in \mathbb{R}^D. \tag{4.20}$$

We then construct a Dtree T such that each cuboid Q_j is a node of T. This requires an application of the algorithm MAKE DTREE. We guarantee that T has at most CN nodes.

We mark each node Q_i of T $(1 \le i \le N)$ with the linear functionals $\overline{\mu}_1^{Q_i}, \dots, \overline{\mu}_D^{Q_i} : \mathbb{R}^D \to \mathbb{R}$. We mark each node $\widetilde{Q}$ of T that is not among $Q_1, \dots, Q_N$ with linear functionals $\overline{\mu}_1^{\widetilde{Q}}, \dots, \overline{\mu}_D^{\widetilde{Q}} : \mathbb{R}^D \to \mathbb{R}$, all of which are simply zero.

Equipped with these markings, T becomes an ADTree. (See Section 4.1.2 for the definition of an ADTree.)

Applying the algorithm ADPROCESS to the ADTree T, we perform one-time work, after which we can answer queries. A query consists of a dyadic cuboid Q

in $\mathbb{R}^n$. The response to the query Q is a set of linear functionals $\widehat{\mu}_1^Q, \dots, \widehat{\mu}_D^Q : \mathbb{R}^D \to \mathbb{R}$ such that

$$c\sum_{j=1}^{D} |\widehat{\mu}_j^Q(v)|^p \leq \sum_{Q_i \subset Q} \sum_{j=1}^{D} |\overline{\mu}_j^{Q_i}(v)|^p \leq C\sum_{j=1}^{D} |\widehat{\mu}_j^Q(v)|^p \qquad \text{for all } v \in \mathbb{R}^D.$$

This estimate and (4.20) show that the functionals $\widehat{\mu}_1^Q, \dots, \widehat{\mu}_D^Q$ satisfy the conclusion of the present algorithm. The work and storage used are easily seen to be as promised. □

<u>Algorithm: Placing a point inside target cuboids</u>

Given a list of dyadic cuboids $Q_1, \dots, Q_N \subset Q^\circ$ (not necessarily pairwise disjoint), we perform one-time work $\leq CN(1 + \log N)$ in space $\leq CN$, after which we can answer queries as follows:

A query consists of a point $\underline{x} \in \mathbb{R}^n$.

The response to a query $\underline{x}$ is either one of the Q_i containing $\underline{x}$, or else a promise that no such Q_i exists.

The work to answer a query is at most $C \cdot (1 + \log N)$.

Here, C depends only on the dimension n.

Explanation. We construct a DTree T and its control tree $\mathbb{T} = \mathrm{CT}(T)$ with at most CN nodes (using the paperback edition of Make control tree), such that each of the Q_j is a node of T. We mark each node of T as *original* if and only if it is one of the Q_j. Furthermore, we mark each node Q of T to indicate either the smallest (under inclusion) original node containing Q, or else to indicate that no such original node exists. (See Make DTree.)

Recall that each internal node $\xi \in \mathbb{T}$ has two children, marked as gochild(ξ) and staychild(ξ). Each internal node $\xi \in \mathbb{T}$ is also marked with a node $Q_{\text{split}}(\xi) \in T$. Each node $\xi \in \mathbb{T}$ is marked with a node $Q_{\text{root}}(\xi) =$ the root of the DTree BT(ξ).

Recall that we've marked each ξ to say whether BT(ξ) is a singleton.

Recall that BT(root($\mathbb{T}$)) $= T$. Also recall that

$$\begin{aligned} \mathrm{BT}(\mathrm{gochild}(\xi)) &= \mathrm{Descendants}(Q_{\text{split}}(\xi), \mathrm{BT}(\xi)), \text{ and} \\ \mathrm{BT}(\mathrm{staychild}(\xi)) &= \mathrm{Nondescendants}(Q_{\text{split}}(\xi), \mathrm{BT}(\xi)) \end{aligned}$$

for each internal node $\xi \in \mathbb{T}$.

Let $\underline{x} \in \mathbb{R}^n$ and $\xi \in \mathbb{T}$ be given. We consider the following procedure.

<u>Procedure find-original-node $(\underline{x}, \xi)$</u>

We determine whether $\underline{x}$ *is contained in an original node in* BT(ξ). *If such an original node exists, then we exhibit one.*

The above procedure answers our query when applied to $\xi = \mathrm{root}(\mathbb{T})$.

We now assume that $\xi \in \mathbb{T}$ is arbitrary. We ask whether $\underline{x}$ is contained in an original node in BT(ξ).

To study our question, we compare $\underline{x}$ with the root $Q_{\text{root}}(\xi)$ of BT(ξ). If $\underline{x} \notin Q_{\text{root}}(\xi)$, the answer is obviously <u>NO</u>.

(No original node $Q \in \mathrm{BT}(\xi)$ contains $\underline{x}$.)

Suppose $\underline{x} \in Q_{\text{root}}(\xi)$. If $Q_{\text{root}}(\xi)$ is *original*, the answer is obviously YES, and we exhibit $Q_{\text{root}}(\xi)$ as an original node containing $\underline{x}$.

Suppose $Q_{\text{root}}(\xi)$ is *not original*. If $Q_{\text{root}}(\xi)$ is a leaf of $\text{BT}(\xi)$ (i.e., $\text{BT}(\xi)$ is a singleton), then the answer is obviously NO.

Suppose $Q_{\text{root}}(\xi)$ is not a leaf of $\text{BT}(\xi)$ (i.e., $\text{BT}(\xi)$ is not a singleton). We then compare $\underline{x}$ with $Q_{\text{split}}(\xi) \in \text{BT}(\xi)$.

If $\underline{x} \notin Q_{\text{split}}(\xi)$, then all the descendants of $Q_{\text{split}}(\xi)$ in $\text{BT}(\xi)$ are irrelevant for our discussion, i.e., they can't possibly contain $\underline{x}$. Therefore, in this case it's enough to ask whether $\underline{x}$ is contained in an original cuboid of $\text{Nondescendants}(Q_{\text{split}}(\xi), \text{BT}(\xi))$.

Thus, in this case, we can pass from the root ξ to $\xi^+ = \text{staychild}(\xi) \in \mathbb{T}$, and we can answer our question by recursion.

On the other hand, suppose $\underline{x} \in Q_{\text{split}}(\xi)$. We examine the following two situations.

- Suppose $Q_{\text{split}}(\xi)$ *is contained* in an original node of T. Furthermore, suppose that the smallest original node $Q \in T$ containing $Q_{\text{split}}(\xi)$ is contained in $Q_{\text{root}}(\xi)$. Note that $Q_{\text{split}}(\xi)$ and $Q_{\text{root}}(\xi)$ are nodes in $\text{BT}(\xi)$, and $Q_{\text{split}}(\xi) \leq Q \leq Q_{\text{root}}(\xi)$ (inclusion), hence Q is a node of $\text{BT}(\xi)$ because $\text{BT}(\xi)$ is a sub-tree of T (see Lemma 33). Thus, the answer to our question is YES. We exhibit the original node $Q \in \text{BT}(\xi)$ containing $\underline{x}$.
- Suppose that either $Q_{\text{split}}(\xi)$ *is not contained* in an original node of T, or that the smallest original node containing $Q_{\text{split}}(\xi)$ *is not contained* in $Q_{\text{root}}(\xi)$. This means that none of the original nodes in $\text{BT}(\xi)$ contain $Q_{\text{split}}(\xi)$. Then any original node in $\text{BT}(\xi)$ that contains $\underline{x}$ must be a descendant of $Q_{\text{split}}(\xi)$.

  Therefore, we may pass from ξ to $\xi^- = \text{gochild}(\xi) \in \mathbb{T}$, and we can answer our question by recursion.

So, in all cases, we can answer our question.

The one-time work is at most $CN(1 + \log N)$. The query work, apart from recursing, is at most C. We recurse at most $C \cdot (1 + \log N)$ times, since $\mathbb{T}$ has depth at most $C \cdot (1 + \log N)$. So, the query work is at most $C \cdot (1 + \log N)$, as desired.

This completes the description of the procedure FIND-ORIGINAL-NODE. As mentioned before, this yields the algorithm PLACING A POINT INSIDE TARGET CUBOIDS. □

4.2. The Callahan–Kosaraju decomposition

Let $E \subset \mathbb{R}^n$ with $\#(E) = N \geq 2$. A *well-separated pairs decomposition* (WSPD) of E is a finite sequence of Cartesian products $E_1' \times E_1'', \dots, E_L' \times E_L''$ contained in $E \times E$, with the following properties.

(WSPD1) Each pair $(x', x'') \in E \times E$ with $x' \neq x''$ belongs to exactly one of the sets $E_\ell' \times E_\ell''$ (for $\ell = 1, \dots, L$). Moreover, $E_\ell' \cap E_\ell'' = \emptyset$ for $\ell = 1, \dots, L$.

(WSPD2) For each $\ell = 1, \dots, L$, we have $\text{diam}(E_\ell') + \text{diam}(E_\ell'') \leq 10^{-10} \, \text{dist}(E_\ell', E_\ell'')$.

(WSPD3) We have $L \leq CN$, for some constant C depending only on the dimension n.

The next algorithm arises in the work of Callahan and Kosaraju in [9] (see also [17]).

<u>Algorithm: Make WSPD</u>

With work at most $CN \log N$ in space at most CN, we compute a WSPD for E and we compute representative pairs $(x'_\ell, x''_\ell) \in E'_\ell \times E''_\ell$ for $\ell = 1, \dots, L$.

We do not explain here what it means to "compute" a WSPD. This does not matter, however, since we will only need the representative pairs (x'_ℓ, x''_ℓ).

4.3. The BBD tree

We recall a few of the results of Arya, Mount, Netanyahu, Silverman and Wu in [1].

Given $E \subset \mathbb{R}^n$ such that $\#(E) = N \geq 2$, and given $x \in \mathbb{R}^n$, we can enumerate the points of E as $y_1, \dots, y_N$ so that

$$|x - y_1| \leq |x - y_2| \leq \cdots \leq |x - y_N|.$$

We define $d_k(x, E) = |x - y_k|$. The definition of $d_k(x, E)$ is clearly independent of the chosen enumeration.

The following result is contained in [1] (see also [17]).

Theorem 35. *There exists an algorithm with the following properties:*

- *The algorithm receives as input a subset E with $\#(E) = N \geq 2$. The algorithm performs one-time work at most $CN \log N$ using storage CN, after which the algorithm is prepared to answer queries.*
- *A query consists of a point $x \in \mathbb{R}^n$.*
- *The answer to a query x consists of two distinct points $\tilde{x}_1, \tilde{x}_2 \in E$ with $|x - \tilde{x}_1| \leq 2d_1(x, E)$ and $|x - \tilde{x}_2| \leq 2d_2(x, E)$.*
- *The work required to answer a query is at most $C \log N$.*
- *Here, C depends only on the dimension n.*

The proof of Theorem 35 relies on a data structure called a BBD Tree, which is associated to E. As another application of the BBD Tree, we have the following algorithm (see [17]).

<u>Algorithm RCZ</u>

Given real numbers $\lambda(x)$ $(x \in E)$, we perform one-time work at most $CN \log N$ in space CN, after which we can do the following.

Given a dyadic cuboid Q, we can compute the following numbers and points.

- $\#(E \cap Q)$.
- $\min\{\lambda(x) : x \in E \cap Q\}$ (or a promise that $E \cap Q$ is empty).
- A representative point $x(Q) \in E \cap Q$ (if $E \cap Q \neq \emptyset$).
- $\mathrm{diam}(Q \cap E)$.

This computation requires work at most $C \log N$.

Explanation. The computation follows directly from ALGORITHM RCZ1 in Section 25 of [17] and ALGORITHM REP1 in Section 27 of [17]. These algorithms explain how to compute the quantities $\min\{\delta(x, \mathcal{A}) : x \in E \cap Q\}$ and $\#(E \cap Q)$, and how to compute a representative point $x(Q) \in E \cap Q$ when $E \cap Q \neq \emptyset$. The numbers $\delta(x, \mathcal{A})$ $(x \in E)$ in Section 25 of [17] are treated as arbitrary given real numbers (except in Lemma 1 in Section 25 of [17], which is not used elsewhere in the relevant algorithms). See Section 4.6 of this paper for a related discussion.

For each $1 \leq i \leq n$ we define coordinate functions $\lambda_i(x) = x_i$ for $x = (x_1, \dots, x_n) \in E$. Applying the computation in the first bullet point, we compute the quantities

$$r_i := \min\{x_i : x = (x_1, \dots, x_n) \in E \cap Q\}, \text{ and}$$
$$s_i := \max\{x_i : x = (x_1, \dots, x_n) \in E \cap Q\} \quad (1 \leq i \leq n).$$

Thus, we can compute $\operatorname{diam}(E \cap Q) = \max\{|s_i - r_i| : i = 1, \dots, n\}$. (Recall that diameters are measured using the ℓ^∞ norm.) □

Remark 36. Let Q be a dyadic cube. We can decide whether $3Q \cap E$ is nonempty, and if so we compute $\min\{\lambda(x) : x \in E \cap 3Q\}$ and $\#(E \cap 3Q)$. We use a divide and conquer strategy. We write $3Q$ as the disjoint union of 3^n dyadic cubes Q_ν of sidelength δ_Q. We apply ALGORITHM RCZ to each Q_ν. We can tell whether $E \cap 3Q$ is nonempty by checking whether $E \cap Q_\nu$ is nonempty for some ν. We complete the computation using the formulas

$$\min\{\lambda(x) : x \in E \cap 3Q\} = \min_\nu \min\{\lambda(x) : x \in E \cap Q_\nu\}$$
$$\#(E \cap 3Q) = \sum_\nu \#(E \cap Q_\nu).$$

Similarly, we can compute $\min\left\{\lambda(x) : x \in E \cap \frac{65}{64}Q\right\}$ and $\#(E \cap \frac{65}{64}Q)$ (if $E \cap \frac{65}{64}Q \neq \emptyset$). Here, we use the fact that $\frac{65}{64}Q$ is the disjoint union of 130^n dyadic cubes of sidelength $\frac{1}{128}\delta_Q$.

Hence, by replicating the argument in ALGORITHM RCZ, we can compute $\operatorname{diam}(E \cap 3Q)$ and $\operatorname{diam}(E \cap \frac{65}{64}Q)$.

All the above computations requires work at most $C \log N$ after the one-time work of ALGORITHM RCZ has been carried out.

4.4. Clusters

Suppose we are given $E \subset \mathbb{R}^n$ with $\#(E) = N \geq 2$.

Suppose we are given $A \geq 1$. Assume A exceeds a large enough constant determined by n. Assume also that A is an integer power of 2.

In this section, let C, c, C', etc. denote constants determined by n, and let $C(A)$, $c(A)$, $C'(A)$, etc. denote constants determined by A and n.

These symbols may denote different constants in different occurrences.

Recall that we use the l^∞-norm and l^∞-metric on $\mathbb{R}^n$: for $x = (x_1, \dots, x_n) \in \mathbb{R}^n$, we have $|x| = \max_{1 \leq i \leq n} |x_i|$.

A subset $S \subset E$ is called a *cluster* if

$$\#(S) \geq 2 \quad \text{and} \quad \operatorname{dist}(S, E \setminus S) \geq A^3 \cdot \operatorname{diam}(S),$$

where $\operatorname{dist}(S, E \setminus S) = \infty$ if $E \setminus S = \emptyset$.

A cluster S is called a *strong cluster* if

$$\operatorname{dist}(S, E \setminus S) \geq A^5 \cdot \operatorname{diam}(S),$$

where $\operatorname{dist}(S, E \setminus S) = \infty$ if $E \setminus S = \emptyset$.

A cluster that is not a strong cluster is called a *weak cluster.*

Let S be any finite non-empty subset of $\mathbb{R}^n$. The lower left corner of S, denoted by $\operatorname{LLC}(S)$, is defined by

$$\begin{aligned} &\operatorname{LLC}(S) = (x_1, \dots, x_n) \in \mathbb{R}^n, \\ &\qquad \text{where } x_i = \min\{y_i : y = (y_1, \dots, y_n) \in S\} \text{ for } 1 \leq i \leq n. \end{aligned}$$

The upper right corner of S, denoted by $\operatorname{URC}(S)$, is defined by

$$\begin{aligned} &\operatorname{URC}(S) = (x_1, \dots, x_n) \in \mathbb{R}^n, \\ &\qquad \text{where } x_i = \max\{y_i : y = (y_1, \dots, y_n) \in S\} \text{ for } 1 \leq i \leq n. \end{aligned}$$

Note that $\operatorname{diam}(S) = |\operatorname{LLC}(S) - \operatorname{URC}(S)|$, since we are using the l^∞ metric on $\mathbb{R}^n$.

If $S \subset \mathbb{R}^n$ is finite and $\#(S) \geq 2$, then we define the *descriptor cube* of S to be the smallest dyadic cube Q such that Q contains $\operatorname{LLC}(S)$ and $3Q$ contains $\operatorname{URC}(S)$. We note that S has one and only one descriptor cube.

We write $\operatorname{DC}(S)$ for the descriptor cube of S.

If $\operatorname{LLC}(S) = (x_1^\downarrow, \dots, x_n^\downarrow)$ and $\operatorname{URC}(S) = (x_1^\uparrow, \dots, x_n^\uparrow)$, then every $(x_1, \dots, x_n) \in S$ satisfies $x_i^\downarrow \leq x_i \leq x_i^\uparrow$ for $1 \leq i \leq n$.

Hence, if $Q = I_1 \times \cdots \times I_n$ is the descriptor cube of S, we have $x_i^\downarrow, x_i^\uparrow \in 3I_i$ for $1 \leq i \leq n$, consequently, we have $x_i \in 3I_i$ for $1 \leq i \leq n$. It follows that $x \in 3Q$.

Thus, if $Q = \operatorname{DC}(S)$, then $S \subset 3Q$. Moreover, $\operatorname{diam}(S) \geq c\delta_Q$, by the minimal property of Q.

<u>Algorithm: Find descriptor cube</u>

We perform one-time work $\leq CN \log N$ in space CN, after which we answer queries as follows. A query consists of a dyadic cube $\underline{Q}$. The response to a query is as follows:

> Either we guarantee that $\#(3\underline{Q} \cap E) \leq 1$, or we guarantee that $\#(3\underline{Q} \cap E) \geq 2$ and we compute the descriptor cube $\operatorname{DC}(3\underline{Q} \cap E)$ together with the points $\operatorname{LLC}(3\underline{Q} \cap E)$ and $\operatorname{URC}(3\underline{Q} \cap E)$. The query work is at most $C \log N$.

Explanation. We perform the one-time work of the BBD tree, after which we can do the following:

We compute $\#(3\underline{Q} \cap E)$ using Algorithm RCZ (see Remark 36). If $\#(3\underline{Q} \cap E) \leq 1$ then we indicate as such and terminate the computation. Otherwise, we guarantee that $\#(3\underline{Q} \cap E) \geq 2$ and proceed as follows.

Suppose we assign to each $x \in E$ a label $\lambda(x) \in \mathbb{R}$. After one-time work at most $CN \log(N)$ we can answer queries, as follows.

A query is a dyadic cube $\underline{Q}$ and a response to a query is $\max_{x \in E \cap 3\underline{Q}} \lambda(x)$ and $\min_{x \in E \cap 3\underline{Q}} \lambda(x)$. The query work is at most $C \log N$. (See Remark 36.)

Taking $\lambda(x)$ to be the i^{th} coordinate of x for each $x \in E$ and looping over all i, we see that we can perform one-time work at most $CN \log(N)$, after which, given any dyadic query cube $\underline{Q}$, we can compute $\mathrm{LLC}(3\underline{Q} \cap E)$ and $\mathrm{URC}(3\underline{Q} \cap E)$ with work at most $C \log N$.

After we obtain $\mathrm{LLC}(3\underline{Q} \cap E)$ and $\mathrm{URC}(3\underline{Q} \cap E)$, we can compute $\mathrm{DC}(3\underline{Q} \cap E)$ with work at most C.

This completes the explanation of the algorithm FIND DESCRIPTOR CUBE. □

ALGORITHM: MAKE CLUSTER DESCRIPTORS

We produce a list of dyadic cubes $Q_1^{CD}, \dots, Q_L^{CD}$, with the following properties:

- For each l, the set $S_l = 3Q_l^{CD} \cap E$ is a cluster.
- Every strong cluster is one of the the S_l above.
- For each l, the cube Q_l^{CD} is the descriptor cube of S_l.
- $L \le CN$.
- The cubes $Q_1^{CD}, \dots, Q_L^{CD}$ are all distinct.

The algorithm uses work at most $CN \log N$ in space CN.

Explanation. We perform the one-time work to make representatives (x'_ν, x''_ν) $(1 \le \nu \le \nu_{\max})$ of the well-separated pairs decomposition of E. Thus, $\nu_{\max} \le CN$ and for any $x', x'' \in E$ with $x' \ne x''$, there exists ν such that

$$|x' - x'_\nu| + |x'' - x''_\nu| \le 10^{-10}\, |x' - x''| \quad \text{and}$$
$$|x' - x'_\nu| + |x'' - x''_\nu| \le 10^{-10}\, |x'_\nu - x''_\nu| .$$

(See Section 4.2.)

We perform the one-time work of the algorithm FIND DESCRIPTOR CUBE.

We perform the one-time work of the BBD tree. After that, given a dyadic cube $\underline{Q}$, we can compute $\#(E \cap 3\underline{Q})$ in time $C \log N$. (See the algorithm RCZ in Section 4.3.)

For each ν, we let $S_\nu^{\text{cand}} {=} 3Q_\nu^{\text{cand}} {\cap} E$, where Q_ν^{cand} is a dyadic cube containing x'_ν with $2\,|x'_\nu - x''_\nu| \le \delta_{Q_\nu^{\text{cand}}} \le 8\,|x'_\nu - x''_\nu|$.

Instead of computing S_ν^{cand} (which will take too much work), we compute Q_ν^{cand}.

To test whether we "like" S_ν^{cand}, we test whether

$$\#(E \cap 3Q_\nu^{\text{cand}}) = \#(E \cap 3\hat{Q}_\nu^{\text{cand}}), \tag{4.21}$$

where $\hat{Q}_\nu^{\text{cand}} \supset Q_\nu^{\text{cand}}$ is a dyadic cube with sidelength $A^4 \delta_{Q_\nu^{\text{cand}}}$. (This test takes work $C \log N$ after we perform the one-time work of the BBD tree. Recall that A is a power of 2.)

If we like S_ν^{cand} (i.e., (4.21) holds), then we apply the query algorithm within the algorithm Find descriptor cube to find

$$\mathrm{DC}(S_\nu^{\text{cand}}) = \mathrm{DC}(3Q_\nu^{\text{cand}} \cap E).$$

We then add $\mathrm{DC}(S_\nu^{\text{cand}})$ to the list of the cubes $\{Q_l^{\text{CD}}\}$.

If (4.21) does not hold, we do nothing further regarding Q_ν^{cand}.

Thus, we produce a list of cubes

$$Q_1^{\text{CD}}, \dots, Q_L^{\text{CD}}.$$

Note that $L \le \nu_{\max} \le CN$, since each Q_ℓ^{CD} arises from Q_ν^{cand} for some $1 \le \nu \le \nu_{\max}$.

If we like S_ν^{cand}, then $S_\nu^{\text{cand}} = 3Q_\nu^{\text{cand}} \cap E$ is a cluster. Indeed, since

$$\#(E \cap 3Q_\nu^{\text{cand}}) = \#(E \cap 3\hat{Q}_\nu^{\text{cand}}),$$

we have

$$\begin{aligned} \mathrm{dist}(S_\nu^{\text{cand}}, E \setminus S_\nu^{\text{cand}}) &\ge \mathrm{dist}(3Q_\nu^{\text{cand}}, \mathbb{R}^n \setminus 3\hat{Q}_\nu^{\text{cand}}) \\ &\ge cA^4 \delta_{Q_\nu^{\text{cand}}} \ge c'A^4 \cdot \mathrm{diam}(S_\nu^{\text{cand}}); \end{aligned} \tag{4.22}$$

in obtaining inequality (4.22), we used $S_\nu^{\text{cand}} = 3Q_\nu^{\text{cand}} \cap E \subset 3Q_\nu^{\text{cand}}$ and

$$\#\left\{(E \setminus S_\nu^{\text{cand}}) \cap 3\hat{Q}_\nu^{\text{cand}}\right\} = \#\left\{(E \cap 3\hat{Q}_\nu^{\text{cand}}) \setminus (E \cap 3Q_\nu^{\text{cand}})\right\} = 0. \tag{4.23}$$

This completes the proof that S_ν^{cand} is a cluster whenever (4.21) holds.

Next, we show that whenever we like $S_\nu^{\text{cand}} = 3Q_\nu^{\text{cand}} \cap E$ (i.e., (4.21) holds), then

$$\left\{Q_l^{\text{CD}} := \mathrm{DC}(S_\nu^{\text{cand}}), S_l := S_\nu^{\text{cand}}\right\}$$

satisfies

$$S_l = 3Q_l^{\text{CD}} \cap E. \tag{4.24}$$

Indeed, since $Q_l^{\text{CD}} = \mathrm{DC}(S_\nu^{\text{cand}})$, we have

$$S_l = S_\nu^{\text{cand}} \subset 3Q_l^{\text{CD}} \cap E. \tag{4.25}$$

On the other hand, $\delta_{Q_l^{\text{CD}}}$ and $\delta_{Q_\nu^{\text{cand}}}$ are both comparable to $|x'_\nu - x''_\nu|$. Indeed, for Q_ν^{cand}, this follows from the defining condition. For Q_l^{CD}, we have $x'_\nu, x''_\nu \in 3Q_\nu^{\text{cand}} \cap E = S_\nu^{\text{cand}} \subset 3Q_l^{\text{CD}}$, hence

$$|x'_\nu - x''_\nu| \le C\delta_{Q_l^{\text{CD}}}.$$

Also, since $S_\nu^{\text{cand}} = 3Q_\nu^{\text{cand}} \cap E$, we have

$$\mathrm{diam}(S_\nu^{\text{cand}}) \le 3\delta_{Q_\nu^{\text{cand}}} \le C\,|x'_\nu - x''_\nu|,$$

hence $Q_l^{\text{CD}} = \mathrm{DC}(S_\nu^{\text{cand}})$ satisfies

$$\delta_{Q_l^{\text{CD}}} \le C \cdot \mathrm{diam}(S_\nu^{\text{cand}}) \le C'\,|x'_\nu - x''_\nu|.$$

Thus, as claimed, $\delta_{Q_\nu^{\text{cand}}}$ and $\delta_{Q_l^{\text{CD}}}$ are comparable to $|x'_\nu - x''_\nu|$.

Furthermore, since $x'_\nu \in 3Q_\nu^{\text{cand}} \cap 3Q_l^{\text{CD}}$, it follows that

$$3Q_l^{\text{CD}} \cap E \subset AQ_\nu^{\text{cand}} \cap E = 3Q_\nu^{\text{cand}} \cap E;$$

the last equality holds since we like S_ν^{cand}.

Therefore,

$$3Q_l^{\text{CD}} \cap E \subset 3Q_\nu^{\text{cand}} \cap E = S_\nu^{\text{cand}} = S_l. \tag{4.26}$$

From (4.25) and (4.26), we obtain (4.24).

Since $Q_l^{\text{CD}} = \text{DC}(S_\nu^{\text{cand}})$, it now follows that $Q_l^{\text{CD}} = \text{DC}(S_l)$, and $S_l = 3Q_l^{\text{CD}} \cap E$.

Since $S_l = S_\nu^{\text{cand}}$ and S_ν^{cand} is a cluster, we have shown that S_l is a cluster.

We have now proven the first, third, and fourth bullet points asserted in the specification of our algorithm MAKE CLUSTER DESCRIPTORS.

Next, we show the second bullet point: every strong cluster is one of the S_l.

Indeed, let S be a strong cluster. Thus, $\#(S) \geq 2$ and

$$\text{dist}(S, E \setminus S) \geq A^5 \cdot \text{diam}(S).$$

Fix $x', x'' \in S$ such that $|x' - x''| = \text{diam}(S)$. Then we can find ν such that

$$|x'_\nu - x'| + |x''_\nu - x''| \leq 10^{-10}\, |x' - x''| = 10^{-10}\, \text{diam}(S).$$

We have

$$\text{dist}(x'_\nu, S) \leq |x'_\nu - x'| \leq 10^{-10}\, \text{diam}(S) \quad \text{and} \tag{4.27}$$

$$\text{dist}(x''_\nu, S) \leq |x''_\nu - x''| \leq 10^{-10}\, \text{diam}(S). \tag{4.28}$$

Since $x'_\nu, x''_\nu \in E$ and $\text{dist}(S, E \setminus S) \geq A^5 \cdot \text{diam}(S)$, from (4.27) and (4.28), we conclude that $x'_\nu, x''_\nu \in S$.

Next, we show that $S = S_\nu^{\text{cand}} = 3Q_\nu^{\text{cand}} \cap E$.

By definition of Q_ν^{cand}, we have $x'_\nu \in Q_\nu^{\text{cand}}$ and

$$2\,|x'_\nu - x''_\nu| \leq \delta_{Q_\nu^{\text{cand}}} \leq 8\,|x'_\nu - x''_\nu|. \tag{4.29}$$

Therefore, every point $z \in \mathbb{R}^n$ such that $|z - x'_\nu| \leq \frac{3}{2}\,|x'_\nu - x''_\nu|$ belongs to $3Q_\nu^{\text{cand}}$. On the other hand, since $x'_\nu \in S$, we know that every $x \in S$ satisfies

$$|x - x'_\nu| \leq \text{diam}(S) = |x' - x''| \leq \frac{3}{2}\,|x'_\nu - x''_\nu|.$$

Therefore, $S \subset 3Q_\nu^{\text{cand}}$. Since S is a cluster, $S \subset E$. Thus,

$$S \subset 3Q_\nu^{\text{cand}} \cap E.$$

If $S \neq 3Q_\nu^{\text{cand}} \cap E$, then there would exist $\hat{x} \in (3Q_\nu^{\text{cand}} \cap E) \setminus S$. We would then have

$$\text{dist}(E \setminus S, S) \leq |\hat{x} - x'_\nu| \leq C\delta_{Q_\nu^{\text{cand}}} \leq C'\,|x'_\nu - x''_\nu| \leq C'\,\text{diam}(S),$$

contradicting our assumption that S is a strong cluster.

This completes the proof that $S = S_\nu^{\text{cand}} = 3Q_\nu^{\text{cand}} \cap E$, where the last equality follows by definition.

Next, we check that we like S_ν^{cand}, i.e., that

$$\#(3Q_\nu^{\text{cand}} \cap E) = \#(3\hat{Q}_\nu^{\text{cand}} \cap E),$$

where $\hat{Q}_\nu^{\text{cand}} \supset Q_\nu$ is a dyadic cube with $\delta_{\hat{Q}_\nu^{\text{cand}}} = A^4 \delta_{Q_\nu}$.

Indeed, suppose not. Then there would exist $\hat{x} \in (3\hat{Q}_\nu^{\text{cand}} \cap E) \setminus (3Q_\nu^{\text{cand}} \cap E) \subset E \setminus S$, where the last inclusion holds because $S = 3Q_\nu^{\text{cand}} \cap E$. We have

$$\begin{aligned} \operatorname{dist}(E \setminus S, S) &\le |\hat{x} - x'_\nu| \le C\delta_{\hat{Q}_\nu^{\text{cand}}} = CA^4 \delta_{Q_\nu^{\text{cand}}} \\ &\le C'A^4 \, |x'_\nu - x''_\nu| \qquad \text{(see (4.29))} \\ &\le C'A^4 \cdot \operatorname{diam}(S) \qquad \text{(since } x'_\nu, x''_\nu \in S\text{)}, \end{aligned}$$

contradicting our assumption that S is a strong cluster. This completes the proof that we like S_ν^{cand}.

We now know that $\mathrm{DC}(S_\nu^{\text{cand}})$ is one of the Q_l^{CD}, and that (for the same l), we have $S_l = S_\nu^{\text{cand}} = S$.

This proves the second bullet point asserted in our specification of the algorithm MAKE CLUSTER DESCRIPTORS.

It remains to verify the last bullet point of the algorithm MAKE CLUSTER DESCRIPTORS, i.e., the cubes $Q_1^{CD}, \cdots, Q_L^{CD}$ are all distinct. Since $L \le CN$, we can sort the Q_l^{CD}'s and remove the duplicates with work $CN \log N$.

Now all bullet points asserted in the specification of our algorithm hold.

The reader can easily check that the work and storage of our algorithm are as promised. □

Remark 37. Note that the clusters S_l produced (implicitly) by the above algorithm are all distinct, since their descriptor cubes Q_l^{CD} are all distinct, and any cluster has one and only one descriptor cube.

ALGORITHM: LOCATE RELEVANT CLUSTER

After performing the algorithm MAKE CLUSTER DESCRIPTORS and other one-time work, we can answer queries as follows: A query consists of a point $\underline{x} \in \mathbb{R}^n$, for which there exist a strong cluster S and a point $x(S) \in S$ such that

$$A \operatorname{diam}(S) \le |\underline{x} - x(S)| \le A^{-1} \operatorname{dist}(E \setminus S, S). \tag{4.30}$$

We do not assume that S or $x(S)$ is known.

The response to a query $\underline{x}$ is one of the descriptor cubes Q_l^{CD} produced by the algorithm MAKE CLUSTER DESCRIPTORS such that (4.30) holds for $S = S_l := 3Q_l^{CD} \cap E$ and for some $x(S) \in S$.

The one-time work is at most $CN \log N$ in space CN; the query work is at most $C \log N$.

Explanation. Suppose (4.30) holds for some S and for some $x(S) \in S$.

Assume that $\underline{x} \in S$. Then $A \cdot \operatorname{diam}(S) \le |\underline{x} - x(S)| \le \operatorname{diam}(S)$. This gives a contradiction if $A > 1$. Hence, we have shown that $\underline{x} \notin S$.

Assume next that $\underline{x} \in E \setminus S$. Then $\operatorname{dist}(E \setminus S, S) \le |\underline{x} - x(S)| \le A^{-1} \operatorname{dist}(E \setminus S, S)$. This gives a contradiction if $A > 1$. Hence, we have shown that $\underline{x} \notin E \setminus S$.

We have proven that $\underline{x} \notin E$. Now,

$$\operatorname{dist}(\underline{x}, S) \leq |\underline{x} - x(S)| \leq A^{-1} \operatorname{dist}(E \setminus S, S),$$

and if $E \setminus S \neq \emptyset$ then

$$\begin{aligned}\operatorname{dist}(\underline{x}, E \setminus S) &\geq \operatorname{dist}(E \setminus S, S) - \operatorname{dist}(\underline{x}, S) \geq \operatorname{dist}(E \setminus S, S) - |\underline{x} - x(S)| \\ &\geq \operatorname{dist}(E \setminus S, S) - A^{-1} \operatorname{dist}(E \setminus S, S) \geq (1/2) \operatorname{dist}(E \setminus S, S).\end{aligned}$$

If $E \setminus S = \emptyset$, then by definition $\operatorname{dist}(\underline{x}, E \setminus S) = \infty$.

Therefore,

$$\operatorname{dist}(\underline{x}, E \setminus S) \geq cA \cdot \operatorname{dist}(\underline{x}, S),$$

which yields

$$\operatorname{dist}(\underline{x}, S) = \operatorname{dist}(\underline{x}, E).$$

Using the BBD tree, we compute a number $\Delta > 0$ such that

$$8 \cdot \operatorname{dist}(\underline{x}, E) \leq \Delta \leq 32 \cdot \operatorname{dist}(\underline{x}, E),$$

and such that Δ is a power of 2.

We then produce the dyadic cube $Q^\#$ of sidelength Δ containing $\underline{x}$.

We claim that $S = 3Q^\# \cap E$.

To see this, note that $\operatorname{dist}(\underline{x}, E \setminus S) \geq cA \cdot \operatorname{dist}(\underline{x}, S) = cA \cdot \operatorname{dist}(\underline{x}, E) \geq c'A\Delta$. On the other hand, $\underline{x} \in Q^\#$ and $\delta_{Q^\#} = \Delta$.

Therefore, $3Q^\# \cap E \subset S$. If $3Q^\# \cap E \neq S$, then there exists $\hat{x} \in S \setminus (3Q^\# \cap E) \subset S$. On the other hand, since $\underline{x} \in Q^\#$ and $\delta_{Q^\#} = \Delta \geq 8\operatorname{dist}(\underline{x}, E)$, we know that $\frac{3}{2}Q^\#$ contains a point of E; say $\check{x} \in \frac{3}{2}Q^\# \cap E$. Note that $\check{x} \in 3Q^\# \cap E \subset S$.

Thus, $\hat{x}, \check{x} \in S$, with $\check{x} \in \frac{3}{2}Q^\#$ and $\hat{x} \notin 3Q^\#$. Therefore,

$$\begin{aligned}\operatorname{diam}(S) &\geq |\hat{x} - \check{x}| \geq c\delta_{Q^\#} = c\Delta \geq c \cdot \operatorname{dist}(\underline{x}, E) = c \cdot \operatorname{dist}(\underline{x}, S) \\ &\geq c\left[|\underline{x} - x(S)| - \operatorname{diam}(S)\right], \text{ since } x(S) \in S, \\ &\geq c\left[A \operatorname{diam}(S) - \operatorname{diam}(S)\right], \text{ by (4.30)}.\end{aligned}$$

Thus, $\operatorname{diam}(S) \geq c \cdot (A-1) \cdot \operatorname{diam}(S)$. Since S contains at least two points (because it is a cluster), we have reached a contradiction. This completes the proof of our claim that $S = 3Q^\# \cap E$.

Since S is a strong cluster, its descriptor cube $DC(S)$ is among the cubes Q_l^{CD} produced by the algorithm Make cluster descriptors.

We can compute $DC(S)$ by applying the algorithm Find descriptor cube to the query cube $Q^\#$; this produces $DC(S)$ because $S = 3Q^\# \cap E$.

Accordingly, our algorithm proceeds as follows:

- Compute Δ, a power of 2, such that $8\operatorname{dist}(\underline{x}, E) \leq \Delta \leq 32\operatorname{dist}(\underline{x}, E)$, using the BBD tree.
- Produce the dyadic cube $Q^\#$ of sidelength Δ, containing $\underline{x}$.
- Apply the query algorithm in Find descriptor cube to the query cube $Q^\#$. This produces the cube $DC(S)$.

- By a binary search, locate $\mathrm{DC}(S)$ among the cubes $Q_1^{CD}, \dots, Q_L^{CD}$ produced previously by the algorithm MAKE CLUSTER DESCRIPTORS. This produces one of the Q_l^{CD}, which is the descriptor cube for the cluster S in (4.30); in particular $S = S_l = 3Q_l^{CD} \cap E$ satisfies (4.30), for some $x(S) \in S$.

Thus, our algorithm does what we promised. The work and storage of the algorithm are easily seen to be as promised also. □

ALGORITHM: MAKE CLUSTER REPRESENTATIVES

For each of the cubes $Q_1^{CD}, \dots, Q_L^{CD}$ produced by the algorithm MAKE CLUSTER DESCRIPTORS, we compute a point

$$x(S_l) \in S_l = 3Q_l^{CD} \cap E.$$

The algorithm uses work $\leq CN \log N$ in space CN.

Explanation. For each Q_l^{CD} (a dyadic cube), we use the BBD tree to compute a point $x(S_l) \in 3Q_l^{CD} \cap E$ (which we know to be non-empty, since it is a cluster). We use the fact than $3Q_l^{CD}$ is the union of 3^n dyadic cubes; the required algorithm can be found in Section 4.3.

The work and storage are as promised. □

Since every strong cluster is one of the S_l, we have computed a representative point of every strong cluster and possibly also of some weak clusters.

For each of the clusters $S = S_l := 3Q_l^{CD} \cap E$ we define the *halo* $H(S)$ by

$$H(S) = \left\{ y \in \mathbb{R}^n : A \cdot \mathrm{diam}(S) < |y - x(S)| < A^{-1} \cdot \mathrm{dist}(E \setminus S, S) \right\} \tag{4.31}$$

where $x(S)$ is the cluster representative produced by the algorithm MAKE CLUSTER REPRESENTATIVE.

Finally, we recall Lemma 6.7 from [18].

Lemma 38. *Fix a cluster $S = 3Q_l^{CD} \cap E$. Suppose that $x \in H(S)$ and $x' \in H(S)$ satisfy $|x - x(S)| \geq |x' - x(S)|$. Assume furthermore that x and x' belong to the same connected component of $H(S)$. Then there exist a finite sequence of points $x_1, \dots, x_{\underline{L}} \in H(S)$, and a positive integer L_*, with the following properties:*

- $x_1 = x$ *and* $x_{\underline{L}} = x'$.
- $|x_{\ell+1} - x(S)| \leq |x_\ell - x(S)|$ *for* $\ell = 1, \dots, \underline{L} - 1$.
- $|x_\ell - x_{\ell+1}| \leq A^{-2} |x_\ell - x(S)|$ *for* $\ell = 1, \dots, \underline{L} - 1$.
- $|x_{\ell+L_*} - x(S)| \leq (1 - A^{-3}) |x_\ell - x(S)|$ *for* $1 \leq \ell \leq \underline{L} - L_*$.
- $L_* \leq A^3$.

Remark 39. Lemma 6.7 in [18] was stated incorrectly in dimension $n = 1$. Here, we include the minor yet necessary modifications. The additional assumption that x and x' belong to the same connected component of $H(S)$ is required, since in dimension $n = 1$ the halo $H(S)$ consists of two connected components. Clearly, if x and x' belong to distinct connected components there can be no finite sequence as in the statement of the lemma, and so this extra hypothesis is necessary. We shall not prove this lemma in the case $n = 1$, as the argument is quite obvious. The proof of Lemma 6.7 in [18] remains valid when $n \geq 2$, since then the halos are connected.

4.5. Paths to keystone cubes

We assume we are given a finite subset $E \subset \mathbb{R}^n$, with $N = \#(E) \geq 2$.

We are also given constants $K \geq 10$, $A \geq 10$. We assume that A is greater than a large enough constant determined by n. We further assume that A is a power of 2 and that K is an odd integer.

We write c, C, C', etc. to denote constants that depend only on the dimension n; we write $c(K)$, $C(K)$, and $C'(K)$, etc. to denote constants that depend only on K and n; we write $c(A)$, $c(A)$, $C'(A)$, etc. to denote constants that depend only on A, K, n. These symbols may denote different constants in different occurrences.

We suppose we are given a locally finite collection CZ consisting of dyadic cubes that form a partition of $\mathbb{R}^n$. We do *not* assume that any list of CZ cubes is given; in fact, there are infinitely many CZ cubes. Rather, we assume that we have access to a CZ-Oracle. Given a query point $\underline{x} \in \mathbb{R}^n$, the CZ-Oracle returns the one and only $Q \in$ CZ that contains $\underline{x}$. We do not count any calls to the CZ-Oracle in the one-time work or the query work of any of the algorithms presented here; we will instead keep track of the *number* of calls to the CZ-Oracle.

We make the following assumptions on the decomposition CZ.

- (Good geometry) If $Q, Q' \in$ CZ and $Q \leftrightarrow Q'$, then $\frac{1}{64}\delta_Q \leq \delta_{Q'} \leq 64\delta_Q$.
- (E is nearby) For each $Q \in$ CZ, we have $\#(9Q \cap E) \geq 2$.

Due to good geometry, we see that there exists a constant $c_G > 0$, which is an integer power of 2 depending only on the dimension n, such that, for any $Q, Q' \in$ CZ we have

$$(1 + 8c_G)Q \cap (1 + 8c_G)Q' \neq \emptyset \implies Q \leftrightarrow Q'. \tag{4.32}$$

We next make a few definitions.

A finite sequence $\mathcal{S} = (Q_1, Q_2, \ldots, Q_{\underline{L}})$ consisting of CZ cubes is called a *path* provided that any two consecutive cubes in the sequence touch. This property may be equivalently written as

$$Q_1 \leftrightarrow Q_2 \leftrightarrow Q_3 \leftrightarrow \cdots \leftrightarrow Q_{\underline{L}}.$$

We sometimes say that the path $\mathcal{S}$ *joins* Q_1 to $Q_{\underline{L}}$.

A path $\mathcal{S} = (Q_1, \ldots, Q_{\underline{L}})$ is called *exponentially decreasing* if there exist constants $0 < c(A) < 1$ and $C(A) \geq 1$ such that

$$\delta_{Q_{\ell'}} \leq C(A) \cdot (1 - c(A))^{\ell' - \ell} \delta_{Q_\ell} \quad \text{for } 1 \leq \ell \leq \ell' \leq \underline{L}.$$

In particular, the constants $c(A)$ and $C(A)$ are assumed to be independent of the length $\underline{L}$ of the sequence $\mathcal{S}$.

A cube $Q \in$ CZ is called *keystone* provided $\delta_{Q'} \geq \delta_Q$ for each $Q' \in$ CZ that meets KQ. (Obviously, this definition depends on the choice of K, which will always be clear from the context.)

Recall that a subset $S \subset E$ is called a *cluster* if $\#(S) \geq 2$ and $\mathrm{dist}(S, E \setminus S) \geq A^3 \cdot \mathrm{diam}(S)$.

There is a special collection of clusters $S = 3Q_l^{CD} \cap E$ $(1 \leq \ell \leq L)$ arising in the algorithm MAKE CLUSTER DESCRIPTORS. We compute the representative point $x(S) \in S$ associated to each such cluster S using the algorithm MAKE CLUSTER REPRESENTATIVE. For each such cluster, we let the halo $H(S)$ be defined as in (4.31).

From this point onward in the section, until further notice, we shall assume that $n \geq 2$. We make use of this assumption in several of the results that follow. Toward the end of this section we sketch the modifications required in dimension $n = 1$.

Now suppose we are given $x \in (1 + c_G)Q \cap H(S)$, where $Q \in CZ$ and $S = 3Q_l^{CD} \cap E$ for some fixed $1 \leq \ell \leq L$.

If $\delta_Q < \hat{c}\,|x - x(S)|$ for a small enough constant $\hat{c}$, then since also $x \in (1+c_G)Q$, we have

$$\mathrm{dist}(10Q, x(S)) \geq |x - x(S)| - 10\delta_Q \geq (1/2) \cdot |x - x(S)| \geq (A/2) \cdot \mathrm{diam}(S),$$

hence

$$\mathrm{dist}(10Q, S) \geq \mathrm{dist}(10Q, x(S)) - \mathrm{diam}(S) \geq (A/4) \cdot \mathrm{diam}(S),$$

which implies $10Q \cap S = \emptyset$.

Additionally, note that

$$10Q \subset B(x(S), |x - x(S)| + 10\delta_Q) \subset B(x(S), 2\,|x - x(S)|), \tag{4.33}$$

since we assume $\delta_Q < \hat{c}\,|x - x(S)|$.

On the other hand,

$$\mathrm{dist}(x(S), E \setminus S) \geq \mathrm{dist}(S, E \setminus S) \geq A\,|x - x(S)|$$

since we assume that $x \in H(S)$.

Together with (4.33), the above estimate tells us that $10Q \cap (E \setminus S) = \emptyset$.

Since also $10Q \cap S = \emptyset$, we now know that $10Q \cap E = \emptyset$, contradicting our assumption "E is nearby" for the cube $Q \in CZ$.

Thus, our assumption $\delta_Q < \hat{c}\,|x - x(S)|$ must be false. We have proven the following.

$$\left[\begin{array}{c}\text{Suppose } Q \in CZ\,,\ x \in (1 + c_G)Q \cap H(S), \text{ where } S = 3Q_l^{CD} \cap E. \\ \text{Then } \delta_Q \geq c\,|x - x(S)|\,.\end{array}\right] \tag{4.34}$$

Remark 40. Suppose $Q, Q' \in CZ$ and $x \in (1+c_G)Q$, $x' \in (1+c_G)Q'$. From (4.32) and the good geometry of CZ we see that

$$\delta_{Q'} \leq C \cdot [\delta_Q + |x - x'|]\,.$$

This estimate and its analogue with Q and Q' interchanged tell us that

$$c \cdot \left[\delta_Q + |x - x'|\right] \leq \left[\delta_{Q'} + |x - x'|\right] \leq C \cdot \left[\delta_{Q'} + |x - x'|\right]. \tag{4.35}$$

Lemma 41. *Let* $S = 3Q_l^{CD} \cap E$ *be a cluster produced by the algorithm* MAKE CLUSTER DESCRIPTORS, *and let* $x(S) \in S$ *be its associated representative. Recall that*

$$H(S) = \left\{ y \in \mathbb{R}^n : A \cdot \mathrm{diam}(S) < |y - x(S)| < A^{-1} \cdot \mathrm{dist}(E \setminus S, S) \right\}. \tag{4.36}$$

Let $x \in (1+c_G)Q \cap H(S)$, *with* $Q \in CZ$. *Finally let* $Q^{CZ}(S)$ *be the* CZ *cube containing* $x(S)$. *Then* δ_Q *and* $\delta_{Q^{CZ}(S)} + |x - x(S)|$ *differ by at most a factor* C.

Proof. From (4.35) we deduce that $\delta_{Q^{CZ}(S)} + |x - x(S)|$ and $\delta_Q + |x - x(S)|$ differ by at most a factor of C. From (4.34), we have $\delta_Q \geq c|x - x(S)|$. Combining these two estimates, we obtain the conclusion of the lemma. □

Lemma 42. *Let* $S = 3Q_l^{CD} \cap E$ *be a cluster produced by the algorithm* MAKE CLUSTER DESCRIPTORS. *Let* $x, x' \in H(S)$, *and let* $Q, Q' \in CZ$, *with* $x \in Q$ *and* $x' \in Q'$. *If* $|x - x'| \leq A^{-2}|x - x(S)|$, *then* $Q \leftrightarrow Q'$.

Proof. By Lemma 41, we have

$$|x' - x| \leq A^{-2}\left[|x - x(S)| + \delta_{Q^{CZ}(S)}\right] \leq CA^{-2}\delta_Q.$$

Since $x \in Q$, we have $x' \in (1+c_G)Q$. Also $x' \in Q' \subset (1+c_G)Q'$. Thus $(1+c_G)Q \cap (1+c_G)Q' \neq \emptyset$, which implies that $Q \leftrightarrow Q'$. Here, we use (4.32). □

We assume that we have done all the one-time work for the algorithms in Section 4.4. Thus, the query algorithms from Section 4.4 are at our disposal in the present section. Recall that the one-time work just mentioned consists of work at most $CN \log N$ in space at most CN.

<u>ALGORITHM: KEYSTONE-OR-NOT</u>

Given a cube $Q \in CZ$, we produce one of the following outcomes:

(KEY 1) We guarantee that Q is a keystone cube.

(KEY 2) We produce a cube $Q' \in CZ$ such that

$$\delta_{Q'} \leq \frac{1}{2}\delta_Q \quad \text{and} \quad Q' \cap KQ \neq \emptyset,$$

and such that there exists an exponentially decreasing path of CZ cubes

$$Q = Q_1 \leftrightarrow Q_2 \leftrightarrow \cdots \leftrightarrow Q_L = Q',$$

with $L \leq C(K)$, and

$$\delta_{Q_l} \leq C(K) \cdot (1 - c(K))^{l - l'} \cdot \delta_{Q_{l'}} \quad \text{for } 1 \leq l' \leq l \leq L. \tag{4.37}$$

The work, space and number of calls to the CZ-ORACLE required by the algorithm are bounded by a constant $C(K)$.

Explanation. Since K is an odd integer, we can partition KQ into K^n many dyadic cubes $\widetilde{Q}_\nu$, each of sidelength δ_Q. For each $\widetilde{Q}_\nu$, we apply the CZ-ORACLE to determine Q_ν^{CZ}, the CZ cube containing the center of $\widetilde{Q}_\nu$.

The cube Q is keystone if and only if $Q_\nu^{CZ} \supseteq \widetilde{Q}_\nu$ for each ν. Thus, we can test whether Q is a keystone, using work at most $C(K)$ and using at most $C(K)$ calls to the CZ-ORACLE. If Q is a keystone, then we are done.

Suppose Q is not a keystone cube. We claim that there exists a path $Q = Q_1 \leftrightarrow Q_2 \leftrightarrow \cdots \leftrightarrow Q_L$ as in (KEY 2). This claim was essentially proven in Lemma 6.12 of [18]. The main difference is that the keystone cubes in [18] are defined with $K = 100$, whereas here K is an odd integer of size at least 10 (to be fixed later). By making superficial modifications to the argument in Lemma 6.12 of [18] we prove our claim in the present setting.

To find an exponentially decreasing path as in (KEY 2), we first enumerate all the paths of CZ cubes $Q = Q_1 \leftrightarrow Q_2 \leftrightarrow \cdots \leftrightarrow Q_L$ with $L \leq C(K)$, and we then "test" each path to see whether it satisfies the necessary conditions: $\delta_{Q_L} \leq (1/2)\delta_Q$, $Q_L \cap KQ \neq \emptyset$, and (4.37).

There are at most $C(K)$ such paths, and we can generate them using work, space and calls to the CZ-ORACLE at most $C(K)$, because, given a cube $\widetilde{Q} \in CZ$, we can determine all the cubes $\widetilde{Q}' \in CZ$ such that $\widetilde{Q} \leftrightarrow \widetilde{Q}'$, by using at most C calls to the CZ-ORACLE. (Just query the CZ-ORACLE using as $\underline{x}$ the center of each dyadic cube $\widetilde{Q}'$ such that $\widetilde{Q} \leftrightarrow \widetilde{Q}'$ and $\frac{1}{64}\delta_{\widetilde{Q}} \leq \delta_{\widetilde{Q}'} \leq 64\delta_{\widetilde{Q}}$.)

We can "test" a given path using work and storage at most $C(K)$, and using no calls to the CZ-ORACLE.

This concludes the explanation of the algorithm KEYSTONE-OR-NOT. $\square$

ALGORITHM: LIST ALL KEYSTONE CUBES

We produce a list $Q_1^\#, \dots, Q_{L^\#}^\#$, consisting of all the keystone cubes in CZ. Each keystone cube appears once and only once in our list. We have $L^\# \leq C(K)N$. The algorithm uses work at most $C(K)N \log N$ in space $C(K)N$, and at most $C(K)N$ calls to the CZ-ORACLE.

Explanation. Let $Q^\# \in CZ$ be a keystone cube. Since $Q^\# \in CZ$, there exists a point $x \in 9Q^\# \cap E$ due to our assumption that "E is nearby". Let Q_x be the CZ cube containing x. Then $\delta_{Q_x} \geq \delta_{Q^\#}$ because $Q^\#$ is keystone; moreover, $\delta_{Q_x} \leq C\delta_{Q^\#}$ because of good geometry. Hence, for each keystone cube $Q^\#$ there exists $x \in E$ such that

$$x \in 9Q^\# \quad \text{and} \quad c\delta_{Q_x} \leq \delta_{Q^\#} \leq \delta_{Q_x}. \tag{4.38}$$

To generate all the keystone cubes we may therefore proceed as follows.

We loop over all $x \in E$. For each $x \in E$, we produce the unique CZ cube Q_x containing x, using the CZ-ORACLE. We then list all the dyadic cubes $Q^\#$ satisfying (4.38) (there are at most C such $Q^\#$ for a fixed x). Finally, we apply the algorithm KEYSTONE-OR-NOT to test each $Q^\#$ to see whether it is a keystone cube. We discard the cubes $Q^\#$ that are not keystone and retain the remaining cubes. Clearly, a single iteration of the loop requires at most $C(K)$ calls to the CZ-ORACLE and additional work at most $C(K)$.

Thus, with work and storage at most $C(K)N$, and with at most $C(K)N$ calls to the CZ-ORACLE, we produce a list $Q_1^\#, \dots, Q_{L^\#}^\#$, with $L^\# \le C(K)N$, consisting of all the keystone cubes, but possibly containing multiple copies of the same cube.

With work at most $C(K)N \log N$ in space $C(K)N$, we can sort the list $Q_1^\#, \dots, Q_{L^\#}^\#$ and remove duplicates. This completes our explanation of the algorithm LIST ALL KEYSTONE CUBES. □

ALGORITHM: MAKE AUXILIARY CUBES

For each Q_l^{CD}, $S_l = 3Q_l^{CD} \cap E$, produced by the algorithm MAKE CLUSTER DESCRIPTORS, we compute a point $x_l^{\text{extra}} \in H(S_l)$ such that

$$2A \cdot \operatorname{diam}(S_l) \le \left|x_l^{\text{extra}} - x(S_l)\right| \le 8A \cdot \operatorname{diam}(S_l) \tag{4.39}$$

and we compute Q_l^{extra}, the CZ cube containing x_l^{extra}.

The algorithm uses work $\le C(A)N \log N$ in space $C(A)N$, and makes at most $C(A)N$ calls to the CZ-ORACLE.

Explanation. For each $1 \le l \le L$ we compute $\operatorname{diam}(S_l)$; see Remark 36. We choose $x \in \mathbb{R}^n$ satisfying

$$2A \cdot \operatorname{diam}(S_l) \le |x - x(S_l)| \le 8A \cdot \operatorname{diam}(S_l).$$

Note that we necessarily have $x \in H(S_l)$. After picking such a point $x = x_l^{\text{extra}}$, we call the CZ-ORACLE to determine Q_l^{extra}.

Recall that there are at most CN distinct indices l; see the algorithm MAKE CLUSTER DESCRIPTORS. Thus, in the present algorithm, the work, storage and number of calls to the CZ-ORACLE are bounded as required. □

From Lemma 41 and the definition of Q_l^{extra} $(1 \le l \le L)$, we obtain the following:

Lemma 43. *Assume that* $n \ge 2$. *Let* $Q \in CZ$ *and* $l \in \{1, \dots, L\}$, *and suppose that*

$$cA^{10} \cdot \operatorname{diam}(S_l) < |x - x(S_l)| < CA^{-10} \cdot \operatorname{dist}(S_l, E \setminus S_l) \quad \textit{for all } x \in (1 + c_G)Q. \tag{4.40}$$

Then there exists an exponentially decreasing path $\mathcal{S} = (Q_1, \dots, Q_{\bar{J}})$ *joining* Q *to* Q_l^{extra}.

Proof. Denote the point $x^{\text{extra}} = x_l^{\text{extra}}$, the cube $Q^{\text{extra}} = Q_l^{\text{extra}}$, and the cluster $S = S_l$. By the conditions in the algorithm MAKE AUXILIARY CUBES, we have $x^{\text{extra}} \in Q^{\text{extra}} \cap H(S)$.

We will construct an exponentially decreasing path $\mathcal{S} = (\widehat{Q}_1, \dots, \widehat{Q}_{\bar{J}})$ that joins Q to Q^{extra}.

Let $x \in Q$. From (4.40), we see that $x \in Q \cap H(S)$. From (4.39) and (4.40) we see that

$$|x^{\text{extra}} - x(S)| \le 8A \operatorname{diam}(S) \le cA^{10} \operatorname{diam}(S) \le |x - x(S)|.$$

Lemma 38 implies that there exists a sequence of points $x_1, \dots, x_J \in H(S)$ and an integer $J_* \ge 1$, satisfying the following bullet points.

- $x_1 = x$ and $x_J = x^{\text{extra}}$.

- $|x_{j+1} - x(S)| \le |x_j - x(S)|$ for $j = 1, \dots, J-1$.
- $|x_j - x_{j+1}| \le A^{-2}|x_j - x(S)|$ for $j = 1, \dots, J-1$.
- $|x_{j+J_*} - x(S)| \le (1 - A^{-3})|x_j - x(S)|$ for $1 \le j \le J - J_*$.
- $J_* \le A^3$.

(Our assumption that $n \ge 2$ implies that the halo $H(S_l)$ has a single connected component. When $n = 1$ we cannot use Lemma 38, hence we will have to modify our approach.)

Since $|x_j - x(S)|$ is a non-increasing sequence, (4.39) and (4.40) imply that

$$\begin{aligned} 2A\,\mathrm{diam}(S) &\le |x^{\mathrm{extra}} - x(S)| = |x_J - x(S)| \le |x_j - x(S)| \le |x_1 - x(S)| \\ &= |x - x(S)| \le CA^{-10}\,\mathrm{dist}(S, E \setminus S) \le A^{-1}\,\mathrm{dist}(S, E\setminus S) \quad \text{for } 1 \le j \le J. \end{aligned}$$

Hence, $x_j \in H(S)$ for each $j = 1, \dots, J$.

Let Q_j $(1 \le j \le J)$ denote the CZ cube containing x_j. Thus $Q_1 = Q$ and $Q_J = Q^{\mathrm{extra}}$. (Recall that $x_1 = x \in Q$ and $x_J = x^{\mathrm{extra}} \in Q^{\mathrm{extra}}$.)

By the third bullet point and by Lemma 42 we have

$$Q_j \leftrightarrow Q_{j+1} \quad \text{for } j = 1, \dots, J-1. \tag{4.41}$$

Hence, $\delta_{Q_{j+1}}$ and δ_{Q_j} differ by at most a factor of 64, thanks to good geometry.

Recall that $c_G > 0$ is a universal constant satisfying (4.32). We now prove the following:

Claim. If $1 \le J_0 \le J$ satisfies

$$|x_{J_0} - x(S)| \ge c_G \cdot \delta_{Q^{CZ}(S)}, \tag{4.42}$$

then

$$\delta_{Q_{j'}} \le C(A) \cdot (1 - c(A))^{j'-j} \delta_{Q_j} \quad \text{for all } 1 \le j \le j' \le J_0. \tag{4.43}$$

Proof of Claim. Since the sequence $|x_j - x(S)|$ $(1 \le j \le J)$ is non-increasing, (4.42) implies that $|x_j - x(S)| \ge c_G \cdot \delta_{Q^{CZ}(S)}$ for all $1 \le j \le J_0$. Thus, Lemma 41 implies that

$$c \cdot |x_j - x(S)| \le \delta_{Q_j} \le C \cdot |x_j - x(S)| \quad \text{for } 1 \le j \le J_0.$$

This estimate and the fourth bullet point written above imply that

$$\delta_{Q_{J_* k}} \le C \cdot (1 - A^{-3})^{k-\ell} \cdot \delta_{Q_{J_* \ell}} \quad \text{for } 1 \le J_*\ell \le J_* k \le J_0.$$

Since $\delta_{Q_{j+1}}$ and δ_{Q_j} differ by at most a factor of 64, and since $1 \le J_* \le A^3$, we obtain (4.43). This completes the proof of the claim. □

Since $|x_j - x(S)|$ $(1 \le j \le J)$ is non-increasing, the following cases are exhaustive.

Case 1. $|x_J - x(S)| \ge c_G \cdot \delta_{Q^{CZ}(S)}$.

Case 2. There exists $1 \le \bar{J} \le J-1$ such that

- $|x_{\bar{J}+1} - x(S)| < c_G \cdot \delta_{Q^{CZ}(S)}$, and
- $|x_{\bar{J}} - x(S)| \ge c_G \cdot \delta_{Q^{CZ}(S)}$.

Case 3. $|x_1 - x(S)| < c_G \cdot \delta_{Q^{CZ}(S)}$.

First, we consider Case 1. In this case, (4.42) holds with $J_0 = J$. Hence, (4.43) implies that

$$\delta_{Q_{j'}} \leq C(A) \cdot (1 - c(A))^{j'-j} \delta_{Q_j} \quad \text{for } 1 \leq j \leq j' \leq J.$$

We define the sequence $\mathcal{S} := (Q_1, \dots, Q_J)$. The above estimate and (4.41) show that $\mathcal{S}$ is an exponentially decreasing path joining Q to Q^{extra}.

Next, we consider Case 2. In this case, (4.42) holds with $J_0 = \overline{J}$. Hence, (4.43) implies that

$$\delta_{Q_{j'}} \leq C(A) \cdot (1 - c(A))^{j'-j} \delta_{Q_j} \quad \text{for } 1 \leq j \leq j' \leq \overline{J}. \tag{4.44}$$

Recall that $x(S) \in Q^{CZ}(S)$, and $|x_{\overline{J}+1} - x(S)| < c_G \cdot \delta_{Q^{CZ}(S)}$. Hence, $x_{\overline{J}+1} \in (1 + 8c_G)Q^{CZ}(S)$. Since also $x_{\overline{J}+1} \in Q_{\overline{J}+1}$ we see that $(1 + 8c_G)Q_{\overline{J}+1} \cap (1 + 8c_G)Q^{CZ}(S) \neq \emptyset$, hence $Q_{\overline{J}+1} \leftrightarrow Q^{CZ}(S)$ thanks to (4.32).

Moreover, since $|x_j - x(S)|$ is non-increasing, we have

$$|x^{\text{extra}} - x(S)| = |x_J - x(S)| \leq |x_{\overline{J}+1} - x(S)| \leq c_G \cdot \delta_{Q^{CZ}(S)}.$$

Recall that $x(S) \in Q^{CZ}(S)$. Hence, the above estimate shows that $x^{\text{extra}} \in (1 + 8c_G)Q^{CZ}(S)$. Since also $x^{\text{extra}} \in Q^{\text{extra}}$ we see that $(1 + 8c_G)Q^{CZ}(S) \cap (1 + 8c_G)Q^{\text{extra}} \neq \emptyset$, hence $Q^{CZ}(S) \leftrightarrow Q^{\text{extra}}$ thanks to (4.32).

We define the sequence

$$\mathcal{S} := (\widehat{Q}_1, \widehat{Q}_2, \dots, \widehat{Q}_{\overline{\overline{J}}}) := (Q_1, Q_2, \dots, Q_{\overline{J}}, Q_{\overline{J}+1}, Q^{CZ}(S), Q^{\text{extra}}).$$

From (4.41), we see that $\widehat{Q}_j \leftrightarrow \widehat{Q}_{j+1}$ $(1 \leq j \leq \overline{\overline{J}} - 1)$, hence $\delta_{\widehat{Q}_j}$ and $\delta_{\widehat{Q}_{j+1}}$ differ by at most a factor of 64, thanks to good geometry. Combined with (4.44), this shows that $\mathcal{S}$ is an exponentially decreasing path joining Q to Q^{extra}.

Lastly, we consider Case 3. In this case, $Q_1 \leftrightarrow Q^{CZ}(S)$ and $Q^{CZ}(S) \leftrightarrow Q^{\text{extra}}$ as in the discussion of Case 2. Hence, the exponentially decreasing path $\mathcal{S} := (Q_1, Q^{CZ}(S), Q^{\text{extra}})$ joins Q to Q^{extra}.

This completes the proof of Lemma 43. □

A cube $Q \in CZ$ is called *interstellar* provided that

$$(1 + 3c_G)Q \cap E = \emptyset \ \text{ and } \ \operatorname{diam}(A^{10}Q \cap E) \leq A^{-10}\delta_Q.$$

Any cube $Q \in CZ$ that is not interstellar will be called *non-interstellar.*

<u>ALGORITHM: TEST AN INTERSTELLAR CUBE</u>

We perform one-time work at most $C(A)N \log N$ in space $C(A)N$, after which we can answer queries.

A query consists of a cube $Q \in CZ$.

The response to the query Q is as follows. We first determine whether Q is interstellar. If it is, then we find an index $1 \leq l \leq L$ such that $S = S_l = 3Q_l^{CD} \cap E$ satisfies

$$cA^{10} \cdot \operatorname{diam}(S) < |x - x(S)| < CA^{-10} \cdot \operatorname{dist}(S, E \setminus S), \tag{4.45}$$

for all $x \in (1 + c_G)Q$. The query work is at most $C(A) \log N$.

Explanation. We determine whether Q is interstellar by computing $\mathrm{diam}(A^{10}Q \cap E)$ and by testing whether $(1+3c_G)Q \cap E = \emptyset$ using the BBD tree. We compute $\mathrm{diam}(A^{10}Q \cap E)$ using the algorithm RCZ in Section 4.3. We use that $A^{10}Q$ can be expressed as the disjoint union of finitely many dyadic cubes of sidelength $\delta_Q/2$; see Remark 36. Similarly, we test whether $(1+3c_G)Q \cap E = \emptyset$ using the BBD tree and the fact that $(1+3c_G)Q$ can be expressed as the disjoint union of finitely many dyadic cubes of sidelength $c_G\delta_Q/2$; again, see Remark 36. This computation requires work and storage at most $C(A)\log N$.

The proof of Lemma 6.3 in [18] shows that if Q is interstellar, then for some cluster S we have (4.45) for all $x \in (1+c_G)Q$.

It follows that S is a strong cluster and therefore S is among the clusters S_l produced by the algorithm Make cluster descriptors. Hence, for any $x \in (1+c_G)Q$ we have $S = 3\hat{Q} \cap E$, where $\hat{Q}$ is a dyadic cube such that $\delta_{\hat{Q}} \in [8\cdot \mathrm{dist}(x,E), 64\cdot \mathrm{dist}(x,E)]$ and $x \in \widehat{Q}$.

Therefore, given an interstellar Q, we can learn which S_l satisfies (4.45) as follows:

- Let x be the center of Q.
- Compute $\mathrm{dist}(x,E)$ up to a factor of 2, using the BBD tree.
- Compute a dyadic cube $\hat{Q}$ with $x \in \hat{Q}$ and $8\,\mathrm{dist}(x,E) \le \delta_{\hat{Q}} \le 64\,\mathrm{dist}(x,E)$.
- Using the algorithm Find descriptor cube, compute the descriptor cube $Q^{CD} = \mathrm{DC}(3\hat{Q} \cap E)$.
- We know that Q^{CD} will be one of our cubes Q_l^{CD} produced by the algorithm Make cluster descriptors. By a binary search, we find l such that $Q^{CD} = Q_l^{CD}$.
- Thus, we find the l for which $S = S_l = 3Q_l^{CD} \cap E$ satisfies (4.45).

This process takes work $\le C(A)\log N$ and uses at most $C(A)$ calls to the CZ-Oracle. $\square$

Algorithm: List all non-interstellar cubes

Using work at most $C(A)N\log N$ in space $C(A)N$ and making at most $C(A)N$ calls to the CZ-Oracle, we produce all the non-interstellar cubes $Q \in \mathrm{CZ}$.

Explanation. We compute representatives $(x'_\nu, x''_\nu) \in E \times E$ $(\nu = 1,\dots,\nu_{\max})$ arising in the WSPD. These representatives have the property that, for each $(x',x'') \in E\times E \setminus \{(x,x) : x \in E\}$, there exists ν such that

$$|x'_\nu - x'| + |x''_\nu - x''| \le \frac{1}{100}|x' - x''|,$$

and $\nu_{\max} \le CN$.

Let $Q \in \mathrm{CZ}$ be non-interstellar. Then either $(1+3c_G)Q \cap E \ne \emptyset$ or $\mathrm{diam}(A^{10}Q \cap E) > A^{-10}\delta_Q$. In the second case, there exist two points $x', x'' \in A^{10}Q \cap E$ with $|x'-x''| > A^{-10}\delta_Q$. Hence, in the second case there exists ν such that $x'_\nu, x''_\nu \in A^{11}Q \cap E$ and $|x'_\nu - x''_\nu| > A^{-11}\delta_Q$.

We have shown the following: If $Q \in CZ$ is non-interstellar, then

$$\exists\, x \in E \text{ such that } x \in (1+3c_G)Q, \quad \text{or}$$
$$\exists\, \nu \text{ such that } x'_\nu, x''_\nu \in A^{11}Q \cap E \text{ and } |x'_\nu - x''_\nu| > A^{-11}\delta_Q.$$

Thus, to list all the non-interstellar cubes we can proceed as follows:

For each $x \in E$, find all the cubes $Q \in CZ$ such that $(1+3c_G)Q$ contains x. There are at most C such cubes for each fixed x. We produce these cubes by making at most C calls to the CZ-ORACLE.

For each $\nu = 1, \dots, \nu_{\max}$, find all the dyadic cubes Q such that $x'_\nu, x''_\nu \in A^{11}Q \cap E$ and $\delta_Q \le A^{11}|x'_\nu - x''_\nu|$. There are at most $C(A)$ such cubes for each fixed ν.

We have produced a list that contains all the non-interstellar cubes, and consists of at most $C(A)N$ dyadic cubes. We now pass through this list and remove any cubes that are interstellar. We then sort the remaining cubes and remove duplicates.

Thus we have computed the list of all non-interstellar cubes.

The reader may easily check that our algorithm performs as promised in terms of work, storage, and calls to the CZ-ORACLE. □

We now create a list USUAL-SUSPECTS, consisting of all keystone cubes and all non-interstellar CZ cubes, and all the cubes Q_l^{extra} produced by the algorithm MAKE AUXILIARY CUBES.

There are at most $C(A)N$ cubes in the list USUAL-SUSPECTS, and we can produce the list using work $\le C(A)N \log N$, storage $\le C(A)N$, and at most $C(A)N$ calls to the CZ-ORACLE.

We assume that our list USUAL-SUSPECTS is sorted so that $Q, Q' \in$ USUAL-SUSPECTS and $\delta_Q < \delta_{Q'} \Rightarrow Q$ precedes Q' in the list USUAL-SUSPECTS. Thus, all the smallest cubes are located at the beginning of the list. We may also assume that the list USUAL-SUSPECTS contains no duplicates.

This can be achieved by doing extra work at most $C(A)N \log N$ in space $C(A)N$.

In preparation for the next two algorithms, we prove a small lemma.

Lemma 44. *Fix constants $A^\# \ge 1$ and $0 < a^\# < 1$. Suppose we are given finite sequences*

$$\mathcal{S}^{[1]} : \delta_1^{[1]}, \delta_2^{[1]}, \dots, \delta_{L^{[1]}}^{[1]}$$
$$\vdots$$
$$\mathcal{S}^{[M]} : \delta_1^{[M]}, \delta_2^{[M]}, \dots, \delta_{L^{[M]}}^{[M]}$$

of positive real numbers. Assume

- $\delta_\ell^{[k]} \le A^\# \cdot (1-a^\#)^{\ell-\ell'} \delta_{\ell'}^{[k]}$ *for* $1 \le k \le M$ *and* $1 \le \ell' \le \ell \le L^{[k]}$.
- $\delta_{L^{[k]}}^{[k]} \le (1-a^\#)^{L^{[k]}-1} \delta_1^{[k]}$ *for* $1 \le k \le M$.
- $\delta_1^{[k+1]} = \delta_{L^{[k]}}^{[k]}$ *for* $1 \le k \le M-1$.

Then the sequence

$$\left(\delta_1^{[1]},\dots,\delta_{L^{[1]}}^{[1]},\delta_2^{[2]},\dots,\delta_{L^{[2]}}^{[2]},\delta_2^{[3]},\dots,\delta_{L^{[3]}}^{[3]},\dots,\delta_{L^{[M-1]}}^{[M-1]},\delta_2^{[M]},\dots,\delta_{L^{[M]}}^{[M]}\right)$$
$$\equiv(\delta_1,\delta_2,\dots,\delta_J)$$

satisfies

$$\delta_j \le (A^{\#})^2\cdot(1-a^{\#})^{j-j'}\delta_{j'} \quad \textit{for } 1\le j'\le j\le J.$$

Proof. Let $1\le j'\le j\le J$.

First, suppose that $\delta_j=\delta_\ell^{[k]}$ and $\delta_{j'}=\delta_{\ell'}^{[k]}$ with $1\le k\le M$ and $1\le\ell'\le\ell\le L^{[k]}$. Then we have

$$\delta_j=\delta_\ell^{[k]}\le A^{\#}\cdot(1-a^{\#})^{\ell-\ell'}\delta_{\ell'}^{[k]}=A^{\#}\cdot(1-a^{\#})^{j-j'}\delta_{j'}.$$

We consider the remaining case. Namely, suppose that $\delta_j=\delta_\ell^{[k]}$ and $\delta_{j'}=\delta_{\ell'}^{[k']}$ with $1\le k'<k\le M$, $1\le\ell\le L^{[k]}$, and $1\le\ell'\le L^{[k']}$. Then we have

$$\begin{aligned}\delta_j=\delta_\ell^{[k]}&\le A^{\#}\cdot(1-a^{\#})^{\ell-1}\delta_1^{[k]}=A^{\#}\cdot(1-a^{\#})^{\ell-1}\delta_{L^{[k-1]}}^{[k-1]}\\&\le A^{\#}\cdot(1-a^{\#})^{\ell-1}(1-a^{\#})^{L^{[k-1]}-1}\delta_1^{[k-1]}.\end{aligned}$$

Now, iterating the above reasoning we obtain

$$\begin{aligned}\delta_j=\delta_\ell^{[k]}&\le A^{\#}\cdot(1-a^{\#})^{\ell-1}(1-a^{\#})^{L^{[k-1]}-1}\cdots(1-a^{\#})^{L^{[k'+1]}-1}\delta_1^{[k'+1]}\\&=A^{\#}\cdot(1-a^{\#})^{\ell-1}(1-a^{\#})^{L^{[k-1]}-1}\cdots(1-a^{\#})^{L^{[k'+1]}-1}\delta_{L^{[k']}}^{[k']}\\&\le(A^{\#})^2(1-a^{\#})^{\ell-1}(1-a^{\#})^{L^{[k-1]}-1}\cdots(1-a^{\#})^{L^{[k'+1]}-1}(1-a^{\#})^{L^{[k']}-\ell'}\delta_{\ell'}^{[k']}\\&=(A^{\#})^2\cdot(1-a^{\#})^{j-j'}\delta_{j'}.\end{aligned}$$

This completes the proof of the lemma. □

<u>Algorithm: Mark usual suspects</u>

We mark each cube Q appearing in the list usual-suspects with a keystone cube $\mathcal{K}(Q)$ such that Q is joined to $\mathcal{K}(Q)$ by an exponentially decreasing path. That is, there exists a finite sequence of CZ cubes

$$Q=Q_1\leftrightarrow Q_2\leftrightarrow\cdots\leftrightarrow Q_{L(Q)}=\mathcal{K}(Q)$$

such that

$$\delta_{Q_{\ell'}}\le C(A)\cdot(1-c(A))^{\ell'-\ell}\delta_{Q_\ell}\quad\text{for } 1\le\ell\le\ell'\le L(Q).$$

We do not compute $Q_1,\dots,Q_{L(Q)-1}$, but we guarantee that they exist.

If Q is keystone, then we guarantee that $\mathcal{K}(Q)=Q$.

The algorithm does work at most $C(A)N\log N$ in space $C(A)N$, and it makes at most $C(A)N$ calls to the CZ-Oracle.

Explanation. For each Q in the list USUAL-SUSPECTS we will compute a keystone cube $\mathcal{K}(Q) \in$ CZ such that we guarantee that there exists a list of sequences of CZ cubes

$$\begin{aligned} \mathcal{S}_1 &= (Q_1^{[1]}, \dots, Q_{L^{[1]}}^{[1]}), \\ \mathcal{S}_2 &= (Q_1^{[2]}, \dots, Q_{L^{[2]}}^{[2]}), \\ &\vdots \\ \mathcal{S}_M &= (Q_1^{[M]}, \dots, Q_{L^{[M]}}^{[M]}), \end{aligned}$$

with the following properties.

- The initial cube of $\mathcal{S}_1$ is Q and the terminal cube of $\mathcal{S}_M$ is $\mathcal{K}(Q)$, i.e.,
$$Q_1^{[1]} = Q \quad \text{and} \quad Q_{L^{[M]}}^{[M]} = \mathcal{K}(Q).$$
- The terminal cube of a sequence matches the initial cube of its successor:
$$Q_{L^{[k]}}^{[k]} = Q_1^{[k+1]} \quad \text{for } 1 \le k \le M-1.$$
- Each sequence is connected and exponentially decreasing:
$$Q_1^{[k]} \leftrightarrow Q_2^{[k]} \leftrightarrow \cdots \leftrightarrow Q_{L^{[k]}}^{[k]}$$
and
$$\delta_{Q_\ell^{[k]}} \le C_\# \cdot (1-c_\#)^{\ell-\ell'} \delta_{Q_{\ell'}^{[k]}} \quad \text{for } 1 \le \ell' \le \ell \le L^{[k]}.$$
Moreover, we guarantee that
$$\delta_{Q_{L^{[k]}}^{[k]}} \le (1-c_\#)^{L^{[k]}-1} \cdot \delta_{Q_1^{[k]}}.$$
Here, $c_\# \in (0,1)$ and $C_\# \ge 1$ are controlled constants. In this discussion, a "controlled constant" is a constant that depends only on A, K, and n.

If these conditions hold, then we say that Q and $\mathcal{K}(Q)$ are *connected by a chain* $\mathcal{S}_1, \dots, \mathcal{S}_M$ with constants $c_\#$ and $C_\#$.

By concatenating the sequences $\mathcal{S}_1, \dots, \mathcal{S}_M$ we obtain a sequence of CZ cubes as in the algorithm MARK USUAL SUSPECTS, with $C(A) = (C_\#)^2$ and $c(A) = c_\#$. See Lemma 44. Thus it suffices to compute a keystone cube $\mathcal{K}(Q)$ and verify the existence of a suitable chain for each Q in the list USUAL-SUSPECTS.

Recall that the cubes in USUAL-SUSPECTS are sorted according to their size. We loop through all the Q in USUAL-SUSPECTS, starting with the smallest cubes at the beginning of the list. We will compute $\mathcal{K}(Q)$ in the body of the loop, which is presented below.

We fix Q in USUAL-SUSPECTS. By induction, we may assume that for each Q' in USUAL-SUSPECTS with $\delta_{Q'} < \delta_Q$ we have computed a keystone cube $\mathcal{K}(Q')$ to which Q' is connected by a chain with constants $c_\#$ and $C_\#$.

We assume that $c_\#$ is less than a small enough controlled constant, and that $C_\#$ is greater than a large enough controlled constant. We will later pick $c_\#$ and $C_\#$ to be controlled constants, but not yet.

We perform the following procedure.

<u>Main procedure</u>

- We initialize $Q^{[1]} = Q$.
- Let $M_{\max}$ be a large enough integer determined by A, K, and n, to be picked later.
- We perform the following loop: for $(k = 1, \dots, M_{\max} - 1)$
  {
  - We execute the algorithm Keystone-or-not to produce one of two outcomes.
    (Outcome A) We guarantee that $Q^{[k]}$ is a keystone cube. We then return the cube $Q_{\text{out}} = Q^{[k]}$, indicating that it is a keystone cube, and terminate the loop.
    (Outcome B) We witness that $Q^{[k]}$ fails to be a keystone cube: We compute a cube $Q^{[k+1]} \in \text{CZ}$ with $\delta_{Q^{[k+1]}} \le \frac{1}{2}\delta_{Q^{[k]}}$ and $Q^{[k+1]} \cap KQ^{[k]} \neq \emptyset$, such that there exists a sequence $\mathcal{S}_k = (Q_1^{[k]}, \dots, Q_{L^{[k]}}^{[k]})$ of CZ cubes, with $L^{[k]} \le C(K)$ and
    $$Q^{[k]} = Q_1^{[k]} \leftrightarrow Q_2^{[k]} \leftrightarrow \cdots \leftrightarrow Q_{L^{[k]}}^{[k]} = Q^{[k+1]}$$
    such that
    $$\delta_{Q_\ell^{[k]}} \le C(A) \cdot (1 - c(A))^{\ell - \ell'} \delta_{Q_{\ell'}^{[k]}} \quad \text{for } 1 \le \ell' \le \ell \le L^{[k]}.$$
    Combining the estimate $\delta_{Q_{L^{[k]}}^{[k]}} \le \frac{1}{2}\delta_{Q_1^{[k]}}$ with our bound on $L^{[k]}$, we see that
    $$\delta_{Q_{L^{[k]}}^{[k]}} \le (1 - c(A))^{L^{[k]} - 1} \cdot \delta_{Q_1^{[k]}}.$$
    Here, $c(A)$ and $C(A)$ are controlled constants.
  - If $k = M_{\max} - 1$ then we return the cube $Q_{\text{out}} = Q^{[M_{\max}]}$. Using the algorithm Keystone-or-not, we determine whether Q_{out} is a keystone cube, and after indicating the result to the user we terminate the loop.

  }

We will now analyze the output of the Main procedure.

Suppose that the Main procedure returns $Q_{\text{out}} = Q^{[M_0]}$ with $1 \le M_0 \le M_{\max}$. Recall that the Main procedure indicates whether $Q^{[M_0]}$ is keystone.

According to the construction in the Main procedure, the following Main condition holds: there exists a chain connecting $Q = Q^{[1]}$ to $Q^{[M_0]}$ with constants $c(A)$ and $C(A)$.

(If $M_0 = 1$ then a trivial chain connects Q to $Q^{[1]} = Q$.)

The construction proceeds in three cases below.

Case 1. Suppose that $Q^{[M_0]}$ is keystone. According to the Main condition, Q is connected to the keystone cube $Q^{[M_0]}$ by a chain with constants $c_\#$ and $C_\#$.

Here, we assume that $c_{\#} \le c(A)$ and $C_{\#} \ge C(A)$. Therefore, we can define $\mathcal{K}(Q) := Q^{[M_0]}$ and the requisite properties listed in the bullet points at the beginning of the explanation will be satisfied. This concludes the analysis in Case 1.

Note that, if Q is keystone, then $M_0 = 1$ and $Q^{[1]} = Q$. To see this, just examine the MAIN PROCEDURE. Hence, $\mathcal{K}(Q) = Q$ when Q is keystone. This proves one of the conditions in the algorithm.

In the remaining cases, $Q^{[M_0]}$ is not a keystone cube. We then have $M_0 = M_{\max}$ because the loop on k cannot terminate early. Hence, by construction, $Q^{[M_{\max}]}$ is *not* a keystone cube, and

$$\delta_{Q^{[M_{\max}]}} \le 2^{-1} \cdot \delta_{Q^{[M_{\max}-1]}} \le \cdots \le 2^{-M_{\max}+1} \cdot \delta_{Q^{[1]}} = 2^{-M_{\max}+1} \cdot \delta_Q. \tag{4.46}$$

We can determine whether $Q^{[M_{\max}]}$ appears in the list USUAL-SUSPECTS using a binary search. This takes work at most $C(A) \log N$.

Case 2. Suppose that $Q^{[M_{\max}]}$ is in the list USUAL-SUSPECTS. From (4.46) and since $M_{\max} \ge 2$, we have $\delta_{Q^{[M_{\max}]}} \le \frac{1}{2}\delta_Q$, hence $Q^{[M_{\max}]}$ precedes Q in the list USUAL-SUSPECTS. By induction hypothesis, we have computed a keystone cube $\mathcal{K}(Q^{[M_{\max}]})$ to which $Q^{[M_{\max}]}$ is connected by a chain with constants $c_{\#}$ and $C_{\#}$. Moreover, another chain connects Q to $Q^{[M_{\max}]}$ (by the Main Condition). By concatenating these chains, we see that Q is connected to $\mathcal{K}(Q^{[M_{\max}]})$ by a chain with constants $c_{\#}$ and $C_{\#}$. Here, we require that $c_{\#} \le c(A)$ and $C_{\#} \ge C(A)$. We may thus define $\mathcal{K}(Q) := \mathcal{K}(Q^{[M_{\max}]})$ and the requisite properties are satisfied. This concludes the analysis in Case 2.

Case 3. Suppose that $Q^{[M_{\max}]}$ is not in the list USUAL-SUSPECTS. Then $Q^{[M_{\max}]}$ is interstellar, since all non-interstellar CZ cubes appear in the list USUAL-SUSPECTS. Using the algorithm TEST AN INTERSTELLAR CUBE, we determine a value of l such that

$$cA^{10} \cdot \operatorname{diam}(S_l) < |x - x(S_l)| < CA^{-10} \cdot \operatorname{dist}(S_l, E \setminus S_l) \text{ for all } x \in (1 + c_G)Q^{[M_{\max}]}.$$

By definition, the cube $Q^{\text{fin}} := Q_l^{\text{extra}}$ appears in the list USUAL-SUSPECTS.

By Lemma 43 there exists a sequence of CZ cubes $Q_1 \leftrightarrow \cdots \leftrightarrow Q_L$ such that $Q_1 = Q^{[M_{\max}]}$, $Q_L = Q^{\text{fin}}$, and

$$\delta_{Q_\ell} \le C(A) \cdot (1 - c(A))^{\ell - \ell'} \delta_{Q_{\ell'}} \text{ for } 1 \le \ell' \le \ell \le L,$$

for controlled constants $c(A)$ and $C(A)$.

Hence, $\delta_{Q^{\text{fin}}} \le C(A)\delta_{Q^{[M_{\max}]}} \le C(A)2^{-M_{\max}}\delta_Q$. We pick

$$M_{\max} \ge \log_2(C(A)) + 1,$$

and thus we obtain the estimate $\delta_{Q^{\text{fin}}} \le \frac{1}{2}\delta_Q$.

Now, there exists a sequence of CZ cubes $\widetilde{Q}_1 \leftrightarrow \cdots \leftrightarrow \widetilde{Q}_{\widetilde{L}}$ such that $\widetilde{Q}_1 = Q$, $\widetilde{Q}_{\widetilde{L}} = Q^{[M_{\max}]}$, and $\widetilde{L} \le C(A)$ for a controlled constant $C(A)$. This is a consequence of the construction in the MAIN PROCEDURE. We concatenate the sequences $(\widetilde{Q}_\ell)_{1 \le \ell \le \widetilde{L}}$ and $(Q_\ell)_{1 \le \ell \le L}$. The resulting sequence $(\widehat{Q}_1, \ldots, \widehat{Q}_{\widehat{L}})$ satisfies

- $\widehat{Q}_1 = Q$, and $\widehat{Q}_{\widehat{L}} = Q^{\text{fin}}$.
- $\widehat{Q}_\ell \leftrightarrow \widehat{Q}_{\ell+1}$ for $1 \le \ell \le \widehat{L}$.

- $\delta_{\widehat{Q}_\ell} \leq C'(A) \cdot (1 - c'(A))^{\ell-\ell'} \cdot \delta_{Q_{\ell'}}$, for $1 \leq \ell' \leq \ell \leq \widehat{L}$.
- $\delta_{\widehat{Q}_{\widehat{L}}} \leq \frac{1}{2}\delta_{\widehat{Q}_1}$.

Here, $c'(A)$ and $C'(A)$ are controlled constants. The last two bullet points imply that

$$\delta_{\widehat{Q}_{\widehat{L}}} \leq (1 - c''(A))^{\widehat{L}-1}\delta_{Q_1}$$

for a controlled constant $c''(A) \leq c'(A)$. Hence, Q is connected to Q^{fin} by a chain (in fact, the chain consists of a single sequence) with constants $c''(A)$ and $C'(A)$.

Moreover, since $\delta_{Q^{\text{fin}}} \leq \frac{1}{2}\delta_Q$, we know that Q^{fin} precedes Q in the list USUAL-SUSPECTS.

By induction hypothesis, we have computed a keystone cube $\mathcal{K}(Q^{\text{fin}})$ to which Q^{fin} is connected by a chain with constants $c_\#$ and $C_\#$. Moreover, as shown above, Q connects to Q^{fin} by a chain with constants $c''(A)$ and $C'(A)$. Hence, Q connects to $\mathcal{K}(Q^{\text{fin}})$ by a chain with constants $c_\#$ and $C_\#$. Here, we assume that $c_\# \leq c''(A)$ and $C_\# \geq C'(A)$. We may thus define $\mathcal{K}(Q) := \mathcal{K}(Q^{\text{fin}})$ and the requisite properties are satisfied. This concludes the analysis in Case 3.

We review what we have achieved. By looping over all the cubes Q in the list USUAL-SUSPECTS (sorted by size), we have computed for each Q a keystone cube $\mathcal{K}(Q)$, and we have verified that Q is connected to $\mathcal{K}(Q)$ by a chain with constants $c_\#$ and $C_\#$. We may choose $c_\#$ and $C_\#$ to be controlled constants. As mentioned before, by Lemma 44, there thus exists an exponentially decreasing path connecting Q to $\mathcal{K}(Q)$.

The reader may easily check that our algorithm performs as promised in terms of the work, storage, and number of calls to the CZ-Oracle

This concludes the explanation of the algorithm Mark usual suspects. □

Main keystone cube algorithm

We perform one-time work, after which we can answer queries.

A query consists of a cube $\underline{Q} \in \text{CZ}$. The response to a query is a keystone cube $\mathcal{K}(\underline{Q})$.

We guarantee the following:

- For each $\underline{Q} \in \text{CZ}$ there is a finite sequence of CZ cubes

  $$\underline{Q} = Q_1 \leftrightarrow Q_2 \leftrightarrow \cdots \leftrightarrow Q_{\underline{L}} = \mathcal{K}(\underline{Q})$$

  such that

  $$\delta_{Q_\ell} \leq C(A) \cdot (1 - c(A))^{\ell-\ell'}\delta_{Q_{\ell'}} \quad \text{for } 1 \leq \ell' \leq \ell \leq \underline{L}.$$

- If $\underline{Q}$ is keystone, then $\mathcal{K}(\underline{Q}) = \underline{Q}$.
- As part of the one-time work we compute a list called BORDER-DISPUTES, consisting of pairs (Q, Q') with $Q, Q' \in \text{CZ}$. A pair of CZ cubes (Q, Q') belongs to BORDER-DISPUTES if and only if $\mathcal{K}(Q) \neq \mathcal{K}(Q')$ and $Q \leftrightarrow Q'$. We guarantee that the list BORDER-DISPUTES consists of at most $C(A) \cdot N$ pairs of CZ cubes.

- The query work is at most $C(A)\log N$. The query work makes at most $C(A)$ additional calls to the CZ-ORACLE.
- The one-time work is at most $C(A)N\log N$ in space $C(A)N$. The one-time work makes at most $C(A)N$ additional calls to the CZ-ORACLE.

Explanation. As part of the one-time work, we execute the algorithm MARK USUAL SUSPECTS. Hence, each cube Q from the list USUAL-SUSPECTS is marked with a keystone cube $\mathcal{K}(Q)$, and we guarantee that there exists an exponentially decreasing path connecting Q and $\mathcal{K}(Q)$. Furthermore, if Q is keystone, then we guarantee that $\mathcal{K}(Q) = Q$.

We now explain the query algorithm.

Let $\underline{Q}$ be a CZ cube. By a binary search, we can check whether $\underline{Q}$ belongs to the list USUAL-SUSPECTS. This requires work at most $C(A)\log N$.

If $\underline{Q} \in$ USUAL-SUSPECTS, then we have precomputed $\mathcal{K}(\underline{Q})$ satisfying the first bullet point.

Note that all the keystone cubes are among this list of USUAL-SUSPECTS. Hence, the second bullet point will always hold.

If $\underline{Q} \notin$ USUAL-SUSPECTS, then $\underline{Q}$ is interstellar, since all non-interstellar CZ cubes are among the USUAL-SUSPECTS.

Applying the algorithm TEST AN INTERSTELLAR CUBE, we compute an index l for which

$$cA^{10} \cdot \operatorname{diam}(S_l) < |x - x(S_l)| < CA^{-10} \cdot \operatorname{dist}(S_l, E \setminus S_l) \quad \text{for all } x \in (1 + c_G)\underline{Q},$$

where $S_l = 3Q_l^{CD} \cap E$. Hence, for this index l we have

$$(1 + c_G)\underline{Q} \subset H(S_l). \tag{4.47}$$

(See the definition of the halo $H(S_l)$ in (4.36).) This computation requires work at most $C(A)\log N$ and uses at most $C(A)$ calls to the CZ-ORACLE.

By Lemma 43, there exists an exponentially decreasing path of CZ cubes joining $\underline{Q}$ to Q_l^{extra}; moreover, Q_l^{extra} is among the USUAL-SUSPECTS. Therefore, we have precomputed a keystone cube $\mathcal{K}(Q_l^{\text{extra}})$, to which Q_l^{extra} may be joined by an exponentially decreasing path.

We set $\mathcal{K}(\underline{Q}) := \mathcal{K}(Q_l^{\text{extra}})$. Note that $\underline{Q}$ is joined by an exponentially decreasing path to Q_l^{extra} and that Q_l^{extra} is joined by an exponentially decreasing path to $\mathcal{K}(Q_l^{\text{extra}})$. Hence, there exists an exponentially decreasing path as in the first bullet point. The second bullet point holds vacuously. Indeed, all the keystone cubes are among the USUAL-SUSPECTS, and $\underline{Q}$ is not among the USUAL-SUSPECTS, hence $\underline{Q}$ is not keystone.

Thus, we have succeeded in responding to the query $\underline{Q}$.

We see that the work and the number of calls to the CZ-ORACLE in the query work are controlled as required.

This concludes our explanation of the query algorithm.

Next, we explain how to generate the list BORDER-DISPUTES.

Suppose that $\underline{Q}, \underline{\tilde{Q}} \in \mathrm{CZ}$, with $\underline{Q} \leftrightarrow \underline{\tilde{Q}}$. Assume that neither $\underline{Q}$ nor $\underline{\tilde{Q}}$ appears on the list of USUAL-SUSPECTS. Then our query algorithm sets $\mathcal{K}(\underline{Q}) := \mathcal{K}(Q_l^{\text{extra}})$ and $\mathcal{K}(\underline{\tilde{Q}}) := \mathcal{K}(Q_{\tilde{l}}^{\text{extra}})$, where $(1+c_G)\underline{Q} \subset H(S_l)$ and $(1+c_G)\underline{\tilde{Q}} \subset H(S_{\tilde{l}})$, with the usual definitions $S_l = 3Q_l^{\mathrm{CD}} \cap E$ and $S_{\tilde{l}} = 3Q_{\tilde{l}}^{\mathrm{CD}} \cap E$. See (4.47).

We recall Lemma 6.5 in Section 6 of [18], which states that the halos $H(S)$ are pairwise disjoint as S varies over all the clusters.

Since $(1+c_G)\underline{Q} \cap (1+c_G)\underline{\tilde{Q}} \neq \emptyset$, it follows that $S_l = S_{\tilde{l}}$, hence $Q_l = \mathrm{DC}(S_l) = \mathrm{DC}(S_{\tilde{l}}) = Q_{\tilde{l}}$, hence $l = \tilde{l}$.

Therefore, $\mathcal{K}(\underline{Q}) = \mathcal{K}(Q_l^{\text{extra}}) = \mathcal{K}(Q_{\tilde{l}}^{\text{extra}}) = \mathcal{K}(\underline{\tilde{Q}})$.

Consequently, whenever $Q, Q' \in \mathrm{CZ}$ with $Q \leftrightarrow Q'$ and $\mathcal{K}(Q) \neq \mathcal{K}(Q')$, either Q or Q' is among the USUAL-SUSPECTS.

Using our list USUAL-SUSPECTS and the CZ-ORACLE, we can easily generate a list of all pairs of CZ cubes

$$\left[\ (Q, Q') \text{ such that } Q \leftrightarrow Q' \text{ and} Q \text{ or } Q' \in \text{USUAL-SUSPECTS.}\ \right] \tag{4.48}$$

There are at most $C(A)N$ such pairs, and we can generate them, sort them and remove duplicates with work $\leq C(A)N \log N$ in space $C(A)N$, making at most $C(A)N$ calls to the CZ-ORACLE.

Using our query algorithm, we can simply test each pair (Q, Q') satisfying (4.48) to determine whether $\mathcal{K}(Q) = \mathcal{K}(Q')$.

This produces the list BORDER-DISPUTES, satisfying the third bullet point of our algorithm. Since the are at most $C(A)N$ pairs satisfying (4.48), the fourth bullet point holds as well.

Note that we perform work $\leq C(A)N \log N$ in space $C(A)N$, and make at most $C(A)N$ calls to the CZ-ORACLE, in pruning the list (4.48) to make the list BORDER-DISPUTES.

This concludes our explanation of the MAIN KEYSTONE CUBE ALGORITHM. □

The one-dimensional case. We now assume that $n=1$. As noted before, Lemma 43 does not apply in this case. The cause of this failure is the fact that in one dimension the halo $H(S)$ has multiple connected components. Indeed,

$$H(S) = \{y \in \mathbb{R}^n : A \cdot \mathrm{diam}(S) < |y - x(S)| < A^{-1} \cdot \mathrm{dist}(E \setminus S, S)\}$$

is the union of two disjoint intervals.

We start by modifying the construction of x_l^{extra} and Q_l^{extra} from the algorithm MAKE AUXILIARY CUBES. Instead of the points x_l^{extra}, we define

$$x_l^{\text{extra},\pm} = x(S_l) \pm 4A \cdot \mathrm{diam}(S_l).$$

This definition yields the following result, just as before.

- <u>ALGORITHM: MAKE AUXILIARY CUBES (VERSION II)</u>. For each Q_l^{CD}, $S_l = 3Q_l^{\mathrm{CD}} \cap E$, produced by the algorithm MAKE CLUSTER DESCRIPTORS, we compute two points $x_l^{\text{extra},-}, x_l^{\text{extra},+} \in H(S_l)$ such that

$$2A \cdot \mathrm{diam}(S_l) \leq \left|x_l^{\text{extra},j} - x(S_l)\right| \leq 8A \cdot \mathrm{diam}(S_l) \quad \text{for } j \in \{+,-\}.$$

We guarantee that each of the connected components of $H(S_l)$ contains one of the points $x_l^{\text{extra},-}$, $x_l^{\text{extra},+}$. We also compute $Q_l^{\text{extra},-}$ and $Q_l^{\text{extra},+}$, the CZ cubes containing $x_l^{\text{extra},-}$ and $x_l^{\text{extra},+}$, respectively. The algorithm uses work at most $C(A)N \log N$ in space $C(A)N$, and makes at most $C(A)N$ calls to the CZ-ORACLE.

We require the following result, which is a modified version of Lemma 43.

Lemma 45. *Assume that* $n = 1$. *Let* $Q \in CZ$, *and suppose that, for some* $x \in (1 + c_G)Q$,

$$cA^{10} \cdot \operatorname{diam}(S_l) < |x - x(S_l)| < CA^{-10} \cdot \operatorname{dist}(S_l, E \setminus S_l). \tag{4.49}$$

Assume that $j \in \{+, -\}$ *is chosen so that* $x_l^{\text{extra},j}$ *and* x *belong to the same connected component of* $H(S_l)$.

Then there exists an exponentially decreasing path $\mathcal{S} = (Q_1, \dots, Q_{\bar{J}})$ *joining* Q *to* $Q_l^{\text{extra},j}$.

To prove this lemma we mimic the proof of Lemma 43. Let x and $x_l^{\text{extra},j}$ be as above. We apply Lemma 38 to the points x and $x_l^{\text{extra},j}$, which belong to the same connected component of $H(S_l)$ according to hypothesis. Thus there exists a sequence $x_1, \dots, x_J \in H(S_l)$ such that $x_1 = x$ and $x_1 = x_l^{\text{extra},j}$, which satisfies the remaining conditions described in the proof of Lemma 43. The remainder of the argument follows the proof of Lemma 43 in an obvious way.

The remaining modifications necessary for the case $n = 1$ are described below.

- (Following algorithm LIST ALL NON-INTERSTELLAR CUBES, in the definition of USUAL-SUSPECTS.)

  The list USUAL-SUSPECTS consists of all keystone cubes and all non-interstellar cubes, and all the cubes $Q_l^{\text{extra},+}$, $Q_l^{\text{extra},-}$ produced by the algorithm MAKE AUXILIARY CUBES (VERSION II).

- (The explanation of the algorithm MARK USUAL SUSPECTS, in the analysis of Case 2.)

  We know that $(1+c_G)Q' \subset H(S_l)$. We determine $j \in \{+, -\}$ such that $x_l^{\text{extra},j}$ belongs to the connected component of $H(S_l)$ that contains $(1 + c_G)Q'$. For that l and that j, the cube $Q'' = Q_l^{\text{extra},j}$ appears in the list USUAL-SUSPECTS, and by Lemma 45 there exists a sequence of CZ cubes $Q' = Q_1 \leftrightarrow Q_2 \leftrightarrow \cdots \leftrightarrow Q_L = Q''$ such that...

  We set $\mathcal{K}(Q) := \mathcal{K}(Q'')$.

- (The explanation of the MAIN KEYSTONE CUBE ALGORITHM, in the analysis of the case in which $\underline{Q}$ is *not* interstellar.)

  We know that $(1+c_G)\underline{Q} \subset H(S_l)$. We determine $j \in \{+, -\}$ such that $x_l^{\text{extra},j}$ belongs to the same connected component of $H(S_l)$ which contains $(1+c_G)\underline{Q}$. We know that $\underline{Q}$ can be joined by an exponentially decreasing path of CZ cubes to $Q_l^{\text{extra},j}$, and that $Q_l^{\text{extra},j}$ is among the USUAL-SUSPECTS. Therefore, we have precomputed a keystone cube $\mathcal{K}(Q_l^{\text{extra},j})$, to which $Q_l^{\text{extra},j}$ may be joined by an exponentially decreasing path.

We set $\mathcal{K}(\underline{Q}) := \mathcal{K}(Q_l^{\text{extra},j})$.

- (The explanation of the MAIN KEYSTONE CUBE ALGORITHM, in the definition of BORDER-DISPUTES.)

  Then our query algorithm sets $\mathcal{K}(\underline{Q}) = \mathcal{K}(Q_l^{\text{extra},j})$ and $\mathcal{K}(\underline{\tilde{Q}}) = \mathcal{K}(Q_{\tilde{l}}^{\text{extra},\tilde{j}})$, where $(1 + c_G)\underline{Q}$ and $x_l^{\text{extra},j}$ are contained in the same connected component of $H(S_l)$, and where $(1 + c_G)\underline{\tilde{Q}}$ and $x_{\tilde{l}}^{\text{extra},\tilde{j}}$ are contained in the same connected component of $H(S_{\tilde{l}})$.

  Since $(1 + c_G)\underline{Q} \cap (1 + c_G)\underline{\tilde{Q}} \neq \emptyset$, while the halos $H(S)$ are pairwise disjoint as S varies over all clusters, it follows that $S_l = S_{\tilde{l}}$. Moreover, $(1+c_G)\underline{Q}$ and $(1+c_G)\underline{\tilde{Q}}$ are contained in the same connected component of $H(S_l) = H(S_{\tilde{l}})$, hence $x_l^{\text{extra},j} = x_{\tilde{l}}^{\text{extra},\tilde{j}}$. Thus we have $l = \tilde{l}$ and $j = \tilde{j}$.

  Therefore, $\mathcal{K}(\underline{Q}) = \mathcal{K}(Q_l^{\text{extra},j}) = \mathcal{K}(Q_{\tilde{l}}^{\text{extra},\tilde{j}}) = \mathcal{K}(\underline{\tilde{Q}})$.

This concludes the list of modifications required to treat the case $n = 1$.

4.6. CZ decompositions

4.6.1. Preliminaries.

Lemma 46. *Let $0 < \gamma < 1$ with γ an integer power of two. Let CZ be a collection of pairwise disjoint dyadic cubes. We assume either that CZ is a dyadic decomposition of a unit cube Q° or that CZ is a dyadic decomposition of $\mathbb{R}^n$.*

Assume that for all $Q, Q' \in \text{CZ}$ with $Q \leftrightarrow Q'$, we have $\gamma\delta_{Q'} \leq \delta_Q \leq \gamma^{-1}\delta_{Q'}$.

Then, for any $Q, Q' \in \text{CZ}$ with $(1+\gamma/2)Q \cap (1+\gamma/2)Q' \neq \emptyset$, we have $Q \leftrightarrow Q'$.

Proof. We assume that CZ is a dyadic decomposition of Q°, where Q° is a unit cube. The case in which CZ is a dyadic decomposition of $\mathbb{R}^n$ is treated similarly.

Let $Q, Q' \in \text{CZ}$ satisfy $(1 + \gamma/2)Q \cap (1 + \gamma/2)Q' \neq \emptyset$ and $\delta_Q \geq \delta_{Q'}$. For the sake of contradiction suppose that Q and Q' do not meet. That is, we assume that the closure of Q is disjoint from the closure of Q'.

Fix a point $z \in (1 + \gamma/2)Q \cap (1 + \gamma/2)Q'$. Now,

$$d(Q, Q') = \inf_{\substack{x \in Q \\ x' \in Q'}} |x - x'| \leq \inf_{\substack{x \in Q \\ x' \in Q'}} |x - z| + |x' - z| \leq (\gamma/4)\delta_Q + (\gamma/4)\delta_{Q'},$$

where in the last inequality we use that $z \in (1 + \gamma/2)Q$ and $z \in (1 + \gamma/2)Q'$. (Recall that we use the ℓ^∞ norm on $\mathbb{R}^n$.) Since $\delta_{Q'} \leq \delta_Q$, we conclude that $d(Q, Q') \leq (\gamma/2)\delta_Q$.

Consider the subset

$$\mathcal{D}_Q = \bigcup \{\overline{Q} : \overline{Q} \in \text{CZ},\ \overline{Q} \leftrightarrow Q\} \subset Q^\circ.$$

According to good geometry, each of the above $\overline{Q}$ satisfies $\delta_{\overline{Q}} \geq \gamma\delta_Q$. Thus, because the cubes in CZ are a partition of Q°, we have

$$\{y \in Q^\circ : d(y, Q) \leq (3\gamma/4) \cdot \delta_Q\} \subset \mathcal{D}_Q.$$

Hence, since $d(Q, Q') \leq (\gamma/2)\delta_Q$, we know that $\mathcal{D}_Q$ intersects Q'. Therefore, there exists $\overline{Q} \in$ CZ with $\overline{Q} \leftrightarrow Q$ and $\overline{Q} \cap Q' \neq \emptyset$. Hence, because the cubes in CZ are pairwise disjoint, we must have $\overline{Q} = Q'$. Thus, $Q' \leftrightarrow Q$, which contradicts our assumption that Q and Q' do not meet. This completes the proof of the lemma by contradiction. □

4.6.2. Review of known results. We review several results from Sections 20-26 in [17]. In those sections, we are given the following data (see Section 20 in [17]):

- A finite subset $E \subset \mathbb{R}^n$, with $\#(E) = N$, $N \geq 2$.
- A real number $A_2 \geq 1$, assumed to be an integer power of 2.
- For each $x \in E$ and $\mathcal{A} \subset \mathcal{M}$, a positive real number $\delta(x, \mathcal{A})$. [3]

These data define a family of Calderón–Zygmund decompositions of $\mathbb{R}^n$, called CZ$(\mathcal{A})$, indexed by subsets $\mathcal{A} \subset \mathcal{M}$.

Here, CZ$(\mathcal{A})$ consists of the maximal dyadic cubes $Q \subset \mathbb{R}^n$ of sidelength $\delta_Q \leq A_2^{-1}$ such that either

(a) $\#(5Q \cap E) \leq 1$, or

(b) for some $\mathcal{A}' \leq \mathcal{A}$ we have $\delta(x, \mathcal{A}') \geq A_2\delta_Q$ for all $x \in E \cap 5Q$.

The following algorithm is presented in Section 26 of [17].

Given A_2, E, $(\delta(x, \mathcal{A}))_{\mathcal{A} \subset \mathcal{M}, x \in E}$, we perform one-time work at most $CN \log N$ in space CN, after which we can answer queries as follows.

A query consists of a subset $\mathcal{A} \subset \mathcal{M}$ and a point $\underline{x} \in \mathbb{R}^n$. The response to the query $(\mathcal{A}, \underline{x})$ is the one and only one cube $Q \in$ CZ$(\mathcal{A})$ containing $\underline{x}$. The work to answer a query is at most $C \log N$. Here, C depends only on m and n.

We will make a slight change here by replacing $5Q$ by $3Q$ in the definition of CZ$(\mathcal{A})$ (see (a) and (b) above). This change affects nothing significant in the relevant discussion in [17].

The only point worth mentioning is the proof of good geometry. Lemma 2 in Section 21 of [17] asserts that if $(1+2c_G)Q \cap (1+2c_G)Q' \neq \emptyset$ with $Q, Q' \in$ CZ$(\mathcal{A})$, then $\frac{1}{2}\delta_Q \leq \delta_{Q'} \leq 2\delta_Q$. Here, $c_G > 0$ is a small constant depending only on the dimension n.

The proof of that lemma requires slight changes; the argument given in [17] shows that $\frac{1}{2}\delta_Q \leq \delta_{Q'} \leq 2\delta_Q$ for any $Q, Q' \in$ CZ$(\mathcal{A})$ with $Q \leftrightarrow Q'$. Thus, applying Lemma 46 (with $\gamma = 1/2$), we see that

$$\left[Q, Q' \in \mathrm{CZ}(\mathcal{A}), (1+2c_G)Q \cap (1+2c_G)Q' \neq \emptyset\right] \implies \frac{1}{2}\delta_Q \leq \delta_{Q'} \leq 2\delta_Q.$$

This proves the "good geometry" of the cubes in CZ$(\mathcal{A})$.

[3] We recall from [17] that Lemma 5 in Section 20 there makes use of a particular choice of the $\delta(x, \mathcal{A})$, but that lemma has no effect on anything else in Sections 20-26 of [17]. Again, see the remarks in the first few paragraphs of Section 20 in [17].

4.6.3. A Calderón–Zygmund oracle. Assume we are given the following data.

- We are given a finite set $E \subset Q^\circ$, with $Q^\circ \subset \mathbb{R}^n$ a dyadic cube of unit sidelength; we assume that $\#(E) = N$, $N \geq 2$.
- We are given a number $\Delta(x) \in (0, 1]$ for each $x \in E$. We denote $\vec{\Delta} = (\Delta(x))_{x \in E}$.

Given the data above, we define a Calderón–Zygmund decomposition $\mathrm{CZ}(\vec{\Delta})$ of Q° as follows: $\mathrm{CZ}(\vec{\Delta})$ consists of the maximal dyadic cubes $Q \subset Q^\circ$ such that either $\#(E \cap 3Q) \leq 1$ or $\Delta(x) \geq \delta_Q$ for all $x \in E \cap 3Q$.

Algorithm: Plain vanilla CZ-oracle

Given E, $\vec{\Delta}$ as above, we perform one-time work at most $CN \log N$ in space CN, after which we can answer queries.

A query consists of a point $\underline{x} \in Q^\circ$. The response to the query $\underline{x}$ is the one and only one cube $Q \in \mathrm{CZ}(\vec{\Delta})$ containing $\underline{x}$.

The work to answer a query is at most $C \log N$. Here, C depends only on the dimension n.

Explanation. We take $A_2 = 1$ and $\delta(x, \mathcal{A}) = \Delta(x)$ for each $x \in E$, $\mathcal{A} \subset \mathcal{M}$, and we apply the query algorithm given in the previous section. □

Remark 47. As a special case, we can apply the Plain vanilla CZ-Oracle to the "classic Whitney decomposition" of Q°, which consists of the maximal dyadic subcubes $Q \subset Q^\circ$ such that $\#(E \cap 3Q) \leq 1$. In fact, we need only pick δ_{small} with $0 < \delta_{\text{small}} < \frac{1}{100} \min\{|x - y| : x, y \in E, x \neq y\}$, and then take $\Delta(x) = \delta_{\text{small}}$ for all $x \in E$.

Such a number δ_{small} may be computed with one-time work $\leq CN$ once we have the well-separated pairs decomposition available. The classic Whitney decomposition coincides with $\mathrm{CZ}(\vec{\Delta})$. We will use the Oracle for this decomposition in a later section.

We close this section with an easy generalization of the Plain vanilla CZ-Oracle. We assume we have already defined a decomposition CZ_{old} of Q° consisting of pairwise disjoint dyadic subcubes. We make the following assumptions:

- If $Q \subset Q^\circ$ is a dyadic subcube and $\#(E \cap 3Q) \leq 1$, then Q is contained in a cube of CZ_{old}.
- Good geometry: If $Q, Q' \in \mathrm{CZ}_{\text{old}}$ and $Q \leftrightarrow Q'$ then $\frac{1}{2}\delta_Q \leq \delta_{Q'} \leq 2\delta_Q$.
- We have available a CZ_{old}-Oracle: Given a query point $\underline{x} \in Q^\circ$, the CZ_{old}-Oracle returns the one and only one cube $Q \in \mathrm{CZ}_{\text{old}}$ containing $\underline{x}$.

We define a decomposition CZ_{new} of Q° to consist of the maximal dyadic cubes $Q \subset Q^\circ$ such that either $Q \in \mathrm{CZ}_{\text{old}}$ or $\Delta(x) \geq \delta_Q$ for all $x \in E \cap 3Q$.

We clearly see that the decomposition CZ_{new} has good geometry, namely

$$\text{If } Q, Q' \in \mathrm{CZ}_{\text{new}} \text{ and } Q \leftrightarrow Q' \text{ then } \frac{1}{2}\delta_Q \leq \delta_{Q'} \leq 2\delta_Q.$$

Applying Lemma 46 with $\gamma = 1/2$, we obtain
(4.50)

$$\text{If } Q, Q' \in CZ_{\text{new}} \text{ and } \frac{65}{64}Q \cap \frac{65}{64}Q' \neq \emptyset, \text{ then } Q \leftrightarrow Q' \text{ and } \frac{1}{2}\delta_Q \leq \delta_{Q'} \leq 2\delta_Q.$$

We finish this section with the following algorithm.

ALGORITHM: GLORIFIED CZ-ORACLE

Given E and $\vec{\Delta}$ as above, we perform one-time work at most $CN \log N$ in space CN, after which we can answer queries.

A query consists of a point $\underline{x} \in Q^\circ$. The response to the query $\underline{x}$ is a list containing all the cubes $Q \in CZ_{\text{new}}$ such that $\underline{x} \in \frac{65}{64}Q$.

We answer the query using at most $C \log N$ computer operations as well as at most C calls to the CZ_{old}-ORACLE.

Explanation. First, given $\underline{x} \in Q^\circ$, we show how to compute the unique cube $Q_{\underline{x}} \in CZ_{\text{new}}$ containing $\underline{x}$. In fact, $Q_{\underline{x}}$ is simply the larger of the following two cubes:

- The cube returned by the CZ_{old}-ORACLE in response to the query $\underline{x}$.
- The cube returned by the PLAIN VANILLA CZ-ORACLE applied to $\vec{\Delta} = (\Delta(x))_{x \in E}$ in response to the query $\underline{x}$.

The above computation requires work at most $C \log N$ and one call to the CZ_{old}-ORACLE.

Next, given $\underline{x} \in Q^\circ$, we show how to compute a list of all $Q \in CZ_{\text{new}}$ such that $\underline{x} \in \frac{65}{64}Q$. To do so, we first compute the cube $Q_{\underline{x}} \in CZ_{\text{new}}$ containing $\underline{x}$. By (4.50), our desired list of cubes consists only of dyadic cubes $Q \subset Q^\circ$ such that

$$\left[\underline{x} \in \frac{65}{64}Q \text{ and } \frac{1}{2}\delta_{Q_{\underline{x}}} \leq \delta_Q \leq 2\delta_{Q_{\underline{x}}}\right].$$

There are at most C such cubes, and we can easily list them all.

Now, we test each such Q to see whether $Q \in CZ_{\text{new}}$. To do that, we just compute the one and only one cube $\widehat{Q} \in CZ_{\text{new}}$ containing the center of Q, and we check whether $\widehat{Q} = Q$.

This completes our description of the GLORIFIED CZ-ORACLE. It is trivial to check that the algorithm works, and that the one-time work, query work, the storage, and the number of calls to the CZ_{old}-ORACLE are as promised. □

4.6.4. Basic algorithms. In the present section and in the next section (Section 4.6.5), we assume that we are given the following.

- A finite set $E \subset \frac{1}{32}Q^\circ$, with Q° a dyadic cube of unit sidelength in $\mathbb{R}^n$, such that $N := \#(E) \geq 2$.
- A collection CZ consisting of dyadic cubes $Q \subset \mathbb{R}^n$. We assume that CZ is locally finite, i.e., any given compact set $S \subset \mathbb{R}^n$ intersects a finite number of cubes $Q \in CZ$. Furthermore, we assume

$$\text{(Good geometry):} \quad \text{if } Q \leftrightarrow Q' \text{ and } Q, Q' \in CZ, \text{ then } \frac{1}{8}\delta_Q \leq \delta_{Q'} \leq 8\delta_Q.$$

- We assume we are either in
  Setting 1: The cubes in CZ partition Q°, or
  Setting 2: The cubes in CZ partition $\mathbb{R}^n$.
- We assume that a CZ-ORACLE is available. The CZ-ORACLE accepts queries. In **Setting 1**, a query consists of a point $\underline{x} \in Q^\circ$; in **Setting 2**, a query consists of a point $\underline{x} \in \mathbb{R}^n$. Given a query $\underline{x}$, the CZ-ORACLE produces a list of the cubes $Q \in CZ$ such that $\underline{x} \in \frac{65}{64}Q$. This requires work at most $C \cdot \log N$.

We see that CZ satisfies the hypotheses of Lemma 46 with $\gamma = 1/8$. Thus,

$$\text{If } \frac{65}{64}Q \cap \frac{65}{64}Q' \neq \emptyset \text{ and } Q, Q' \in CZ, \text{ then } Q \leftrightarrow Q' \text{ and } \frac{1}{8}\delta_Q \leq \delta_{Q'} \leq 8\delta_Q. \tag{4.51}$$

Let $Q \in CZ$ and $\overline{Q} \in CZ \setminus \{Q\}$ be given. Let x_Q be the center of Q. Suppose that $\frac{65}{64}\overline{Q} \cap B(x_Q, \delta) \neq \emptyset$ for some $0 < \delta < \frac{1}{64}\min\{\delta_{\overline{Q}}, \delta_Q\}$. Then $\frac{65}{64}\overline{Q} \cap \frac{65}{64}Q \neq \emptyset$. From (4.51), we see that δ_Q and $\delta_{\overline{Q}}$ differ by at most a factor of 16. Thus, because Q and $\overline{Q}$ are disjoint dyadic cubes, we have

$$d(x_Q, \overline{Q}) \geq \frac{1}{2}\delta_Q \geq \frac{1}{32}\delta_{\overline{Q}}.$$

(Recall, distances are measured using the ℓ^∞ metric.)

However, our assumption that $\frac{65}{64}\overline{Q} \cap B(x_Q, \delta) \neq \emptyset$ implies that $d(x_Q, \overline{Q}) \leq \delta + \frac{1}{64}\delta_{\overline{Q}} < \frac{1}{32}\delta_{\overline{Q}}$. This contradiction establishes

$$\text{if } Q, \overline{Q} \in CZ \text{ and } Q \neq \overline{Q}, \text{ then } B(x_Q, \delta) \cap \frac{65}{64}\overline{Q} = \emptyset \text{ for } \delta < \frac{1}{64}\min\{\delta_{\overline{Q}}, \delta_Q\}. \tag{4.52}$$

Under the above assumptions, we give the following algorithms.

<u>ALGORITHM: FIND NEIGHBORS</u>

We can answer queries as follows. A query consists of a cube $Q \in CZ$. The response to the query Q is the list of all cubes $Q' \in CZ$ such that $Q' \leftrightarrow Q$. To answer the query requires work at most $C \log N$.

Explanation. We first explain how to test whether a given dyadic cube $Q' \subset \mathbb{R}^n$ belongs to the collection CZ. In **Setting 1**, it is necessary that $Q' \subset Q^\circ$. Assuming that this is the case, we examine the center $x_{Q'}$ of Q'. We query the CZ-ORACLE on $x_{Q'}$ to produce the list of all cubes $Q \in CZ$ with $x_{Q'} \in \frac{65}{64}Q$. This list contains at most C cubes, thanks to Good Geometry. Note that Q' belongs to this list if and only if Q' belongs to CZ. We can check the former condition using work at most C.

Let $Q \in CZ$ be given. We wish to list all the $Q' \in CZ$ such that $Q' \leftrightarrow Q$. According to good geometry, each such Q' also satisfies $\frac{1}{8}\delta_Q \leq \delta_{Q'} \leq 8\delta_Q$. We can list all the dyadic cubes Q' with $Q' \leftrightarrow Q$ and $\frac{1}{8}\delta_Q \leq \delta_{Q'} \leq 8\delta_Q$. We remove from this list those cubes that do not belong to CZ. We return a list of the remaining cubes.

This completes our description of the algorithm FIND NEIGHBORS. It is easy to check that the algorithm operates as promised, and that the amount of work is as promised. □

ALGORITHM: FIND MAIN-CUBES

After one-time work at most $CN \log N$ in space CN, we produce the collection of cubes $CZ_{\text{main}} := \{Q \in CZ : \frac{65}{64}Q \cap E \neq \emptyset\}$. We mark each cube $Q \in CZ_{\text{main}}$ with a point $x(Q) \in \frac{65}{64}Q \cap E$.

Explanation. We loop over $x \in E$. For each point $x \in E$, we list all the cubes $Q \in CZ$ such that $x \in \frac{65}{64}Q$. This requires N calls to the CZ-ORACLE. For each Q obtained above, we set $x(Q) := x$ for the relevant x. Thus, we produce a list of all the cubes in CZ_{main}, possibly containing duplicates. After sorting this list, we can find and remove duplicates, and obtain our desired list of the cubes $Q \in CZ_{\text{main}}$ marked by points $x(Q)$. □

4.6.5. Partitions of unity. Aside from a decomposition CZ satisfying the conditions laid out in Section 4.6.4, we assume that we are given a cube $\widehat{Q} \subset \mathbb{R}^n$, and real numbers $0 < \bar{r} \leq 1/64$ and $A \geq 1$. We are also given a finite subcollection $\mathcal{Q} \subset CZ$ with the following properties:

(4.53) For each $x \in \widehat{Q}$ we have $x \in (1 + \bar{r}/2)Q$ for some $Q \in \mathcal{Q}$.

(4.54) $\delta_Q \leq A\delta_{\widehat{Q}}$ for each $Q \in \mathcal{Q}$.

(We do not assume here that $\widehat{Q}$ is dyadic.)

By (4.51), we see that the collection $\{\frac{65}{64}Q : Q \in \mathcal{Q}\}$ has *bounded overlap*, meaning that for each $Q \in \mathcal{Q}$ there are at most C cubes $Q' \in \mathcal{Q}$ such that $\frac{65}{64}Q \cap \frac{65}{64}Q' \neq \emptyset$. Here, C depends only on the dimension n.

For each $Q \in CZ$ we choose a cutoff function $\widetilde{\theta}_Q \in C^m(\mathbb{R}^n)$ such that

- $\operatorname{supp}(\widetilde{\theta}_Q) \subset (1 + \frac{3\bar{r}}{4})Q$.
- $\widetilde{\theta}_Q \geq 0$ on $\mathbb{R}^n$.
- $\widetilde{\theta}_Q \geq 1/2$ on $\left(1 + \frac{\bar{r}}{2}\right) Q$.
- $|\partial^\alpha \widetilde{\theta}_Q(x)| \leq C(\bar{r}) \cdot \delta_Q^{-|\alpha|}$ for $x \in \mathbb{R}^n$, $|\alpha| \leq m$.

We choose $\widetilde{\theta}_Q$ to depend only on Q and $\bar{r}$.

We assume the existence of a query algorithm for $\widetilde{\theta}_Q$. For instance, we can take $\widetilde{\theta}_Q$ to be a tensor product of univariate splines, in which case the next algorithm is trivial.

ALGORITHM: COMPUTE CUTOFF FUNCTION

Given a cube $Q \in CZ$, a point $\underline{x} \in Q^\circ$, and $0 < \bar{r} \leq 1/64$, we compute the jet $J_{\underline{x}}(\widetilde{\theta}_Q)$ using work and storage at most C.

In the next lemma we use the cutoff functions $\widetilde{\theta}_Q$ to construct a *partition of unity*. Recall that x_Q denotes the center of a cube Q.

Lemma 48. *There exists* $\theta_Q^{\widehat{Q}} \in C^m(\mathbb{R}^n)$ *for each* $Q \in \mathcal{Q}$, *such that*

$$\operatorname{supp} \theta_Q^{\widehat{Q}} \subset \left(1 + \frac{3\overline{r}}{4}\right) Q, \tag{4.55}$$

$$|\partial^\alpha \theta_Q^{\widehat{Q}}(x)| \leq C(\overline{r}) \cdot \delta_Q^{-|\alpha|} \quad \textit{for } x \in \mathbb{R}^n,\ |\alpha| \leq m, \tag{4.56}$$

$$1 = \sum_{Q \in \mathcal{Q}} \theta_Q^{\widehat{Q}} \quad \textit{on } \widehat{Q}. \tag{4.57}$$

Moreover, $\theta_Q^{\widehat{Q}} = 1$ *near* x_Q *and* $\theta_Q^{\widehat{Q}} = 0$ *near* $x_{Q'}$ *for all* $Q' \in \mathcal{Q} \setminus \{Q\}$.

Here, the constant $C(\overline{r})$ *depends only on* $\overline{r}$, m *and* n.

Proof. We set

$$\Psi(x) = \sum_{\overline{Q} \in \mathcal{Q}} \widetilde{\theta}_{\overline{Q}}(x) \quad \text{for } x \in \mathbb{R}^n.$$

Because $\widetilde{\theta}_{\overline{Q}} \geq 1/2$ on $(1 + \overline{r}/2)\overline{Q}$, the condition (4.53) implies that $\Psi \geq 1/2$ on $\widehat{Q}$. We can easily see that

$$|\partial^\alpha \Psi(x)| \leq C(\overline{r}) \delta_Q^{-|\alpha|} \quad \text{for } x \in \frac{65}{64}Q, Q \in \mathcal{Q},\ |\alpha| \leq m. \tag{4.58}$$

More precisely, since $\operatorname{supp}(\widetilde{\theta}_{\overline{Q}}) \subset \frac{65}{64}\overline{Q}$, any cube $\overline{Q} \in \mathcal{Q}$ that contributes to the sum defining $\Psi(x)$ must satisfy $\frac{65}{64}Q \cap \frac{65}{64}\overline{Q} \neq \emptyset$. (Recall that $x \in \frac{65}{64}Q$.) Moreover, $\delta_{\overline{Q}}$ and δ_Q differ by at most a factor of 16 for any such $\overline{Q}$; see (4.51). Hence, (4.58) follows from (4.56) and from the fact that the sum defining $\Psi(x)$ has at most C nonzero terms for each fixed x, a consequence of the bounded overlap of the cubes in $\mathcal{Q}$.

Let $\eta \in C^m([0,\infty))$ be a function with $\eta(t) \geq 1/4$ for $t \in [0, 1/2)$, and $\eta(t) = t$ for $t \geq 1/2$. Let $Q \in \mathcal{Q}$. We define

$$\theta_Q^{\widehat{Q}}(x) := \frac{\widetilde{\theta}_Q(x)}{\eta \circ \Psi(x)}, \quad \text{a function in } C^m(\mathbb{R}^n).$$

Clearly, $\operatorname{supp} \theta_Q^{\widehat{Q}} \subset \operatorname{supp} \widetilde{\theta}_Q \subset (1 + 3\overline{r}/4)Q$. Moreover, (4.58) implies that

$$|\partial^\alpha [\eta \circ \Psi](x)| \leq C(\overline{r}) \delta_Q^{-|\alpha|} \quad \text{for } x \in \frac{65}{64}Q,\ |\alpha| \leq m.$$

Using that $\eta \circ \Psi(x) \geq 1/4$, we obtain

$$|\partial^\alpha \theta_Q^{\widehat{Q}}(x)| = \left| \partial^\alpha \left[\frac{\widetilde{\theta}_Q}{\eta \circ \Psi} \right](x) \right| \leq C(\overline{r}) \delta_Q^{-|\alpha|} \quad \text{for } x \in \frac{65}{64}Q,\ |\alpha| \leq m.$$

Finally, note that

$$\sum_{Q \in \mathcal{Q}} \theta_Q^{\widehat{Q}}(x) = \sum_{Q \in \mathcal{Q}} \frac{\widetilde{\theta}_Q(x)}{\eta \circ \Psi(x)} = \sum_{Q \in \mathcal{Q}} \frac{\widetilde{\theta}_Q(x)}{\Psi(x)} = 1 \quad \text{for } x \in \widehat{Q}. \tag{4.59}$$

(Here, we use the fact that $\eta \circ \Psi(x) = \Psi(x)$, since $\Psi \geq 1/2$ on $\widehat{Q}$.)

Recall that $\theta_{\overline{Q}}^{\widehat{Q}} \geq 0$ and that $\operatorname{supp}(\theta_{\overline{Q}}^{\widehat{Q}}) \subset \frac{65}{64}\overline{Q}$ for each $\overline{Q} \in \mathcal{Q}$. Thus, from (4.52) and (4.59) we deduce that there exists $\delta > 0$ such that $\theta_Q^{\widehat{Q}} = 1$ on $B(x_Q, \delta)$ and $\theta_Q^{\widehat{Q}} = 0$ on $B(x_{Q'}, \delta)$ for every $Q' \in \mathcal{Q} \setminus \{Q\}$.

This completes the proof of the lemma. □

Lemma 49. *Given a function* $F_Q \in \mathbb{X}((1+\overline{r})Q \cap \widehat{Q})$ *for each* $Q \in \mathcal{Q}$*, we define*

$$F := \sum_{Q \in \mathcal{Q}} F_Q \theta_Q^{\widehat{Q}} \quad \textit{on } \widehat{Q}, \quad \textit{with } \theta_Q^{\widehat{Q}} \textit{ as in Lemma 48.}$$

Then, given a polynomial $P_Q \in \mathcal{P}$ *and a point* $y_Q \in Q$ *for each* $Q \in \mathcal{Q}$*, we have*

$$\begin{aligned} \|F\|^p_{\mathbb{X}(\widehat{Q})} \leq C(A, \overline{r}) \cdot \Big[&\sum_{Q \in \mathcal{Q}} \big[\|F_Q\|^p_{\mathbb{X}((1+\overline{r})Q \cap \widehat{Q})} + \delta_Q^{-mp} \|F_Q - P_Q\|^p_{L^p((1+\overline{r})Q \cap \widehat{Q})}\big] \\ &+ \sum_{\substack{Q, Q' \in \mathcal{Q} \\ (1+\overline{r})Q \cap (1+\overline{r})Q' \neq \emptyset}} \sum_{|\beta| \leq m-1} \delta_Q^{(|\beta|-m)p+n} |\partial^\beta (P_Q - P_{Q'})(y_Q)|^p \Big]. \end{aligned} \tag{4.60}$$

Here, the constant $C(A, \overline{r})$ *depends only on* $\overline{r}$, A, m, n, *and* p.

Proof. Let $Q' \in \mathcal{Q}$ be given, with $\widehat{Q} \cap (1 + \frac{\overline{r}}{2})Q' \neq \emptyset$.

Let $x \in \widehat{Q} \cap (1 + \frac{\overline{r}}{2})Q'$. Recall that $\sum_{Q \in \mathcal{Q}} \theta_Q^{\widehat{Q}} = 1$ on $\widehat{Q}$, hence

$$F = F_{Q'} + \sum_{Q \in \mathcal{Q}} \theta_Q^{\widehat{Q}} \cdot (F_Q - F_{Q'}) \quad \text{on } \widehat{Q}.$$

Differentiating the above equation, for $|\alpha| = m$ we have

$$\partial^\alpha F(x) = \partial^\alpha F_{Q'}(x) + \sum_{\substack{Q \in \mathcal{Q} \\ (1+\frac{3\overline{r}}{4})Q \ni x}} \sum_{\beta + \gamma = \alpha} \operatorname{coeff}(\beta, \gamma) \cdot \partial^\beta (F_Q - F_{Q'})(x) \cdot \partial^\gamma \theta_Q^{\widehat{Q}}(x).$$

There are at most C nonzero terms in the above sum, thanks to bounded overlap of $\{(1+\overline{r})Q : Q \in \mathcal{Q}\}$.

Let $Q \in \mathcal{Q}$ be such that $(1 + 3\overline{r}/4)Q \ni x$. Note that $x \in \widehat{Q} \cap (1+\overline{r})Q$ and $x \in \widehat{Q} \cap (1+\overline{r})Q'$.

In the above sum, if $|\beta| = m$ then $\beta = \alpha$ and $\gamma = 0$. These terms are bounded in magnitude by $|\partial^\alpha F_Q(x)| + |\partial^\alpha F_{Q'}(x)|$.

In the above sum, if $|\beta| \leq m - 1$ then we have

$$|\partial^\beta (F_Q - F_{Q'})(x)| \leq |\partial^\beta (F_Q - P_Q)(x)| + |\partial^\beta (P_Q - P_{Q'})(x)| + |\partial^\beta (F_{Q'} - P_{Q'})(x)|.$$

Since $\widehat{Q} \cap (1 + 3\overline{r}/4)Q \neq \emptyset$ and $\delta_Q \leq A\delta_{\widehat{Q}}$ (see (4.54)), the sidelengths of the rectangular box $\widehat{Q} \cap (1+\overline{r})Q$ are comparable to δ_Q (up to a constant factor depending on $\overline{r}$, A and n). Similarly, the sidelengths of the rectangular box $\widehat{Q} \cap$

$(1+\bar{r})Q'$ are comparable to $\delta_{Q'}$. Thus, by an easy rescaling argument, Lemma 10 shows that

$$\begin{aligned}&|\partial^\beta(F_Q - P_Q)(x)|\\ &\quad\le C(A,\bar{r})\cdot\Big(\delta_Q^{-|\beta|-\frac{n}{p}}\|F_Q - P_Q\|_{L^p((1+\bar{r})Q\cap\widehat{Q})} + \delta_Q^{m-|\beta|-\frac{n}{p}}\|F_Q\|_{\mathbb{X}((1+\bar{r})Q\cap\widehat{Q})}\Big).\\ &|\partial^\beta(F_{Q'} - P_{Q'})(x)|\\ &\quad\le C(A,\bar{r})\cdot\Big(\delta_{Q'}^{-|\beta|-\frac{n}{p}}\|F_{Q'} - P_{Q'}\|_{L^p((1+\bar{r})Q'\cap\widehat{Q})} + \delta_{Q'}^{m-|\beta|-\frac{n}{p}}\|F_{Q'}\|_{\mathbb{X}((1+\bar{r})Q'\cap\widehat{Q})}\Big).\end{aligned}$$

If $\beta+\gamma=\alpha$ then $|\gamma| = m-|\beta|$, hence

$$|\partial^\gamma \theta_Q^{\widehat{Q}}| \le C(A,\bar{r})\cdot\delta_Q^{-|\gamma|} = C(A,\bar{r})\cdot\delta_Q^{|\beta|-m}.$$

Hence,

$$\begin{aligned}|\partial^\alpha F(x)| &\le C(A,\bar{r}) \sum_{\substack{Q\in\mathcal{Q}\\ (1+\bar{r})Q\ni x}} \Big[|\partial^\alpha F_Q(x)| + \delta_Q^{-m-\frac{n}{p}}\|F_Q - P_Q\|_{L^p((1+\bar{r})Q\cap\widehat{Q})}\\ &\qquad + \delta_Q^{-\frac{n}{p}}\|F_Q\|_{\mathbb{X}((1+\bar{r})Q\cap\widehat{Q})} + \sum_{|\beta|\le m-1}|\partial^\beta(P_Q - P_{Q'})(x)|\cdot\delta_Q^{|\beta|-m}\Big]\\ &\qquad\qquad \text{(note that the cube } Q' \text{ enters into the above sum)}\\ &\le C(A,\bar{r}) \sum_{\substack{Q\in\mathcal{Q}\\ (1+\bar{r})Q\ni x}} \Big[|\partial^\alpha F_Q(x)| + \delta_Q^{-m-\frac{n}{p}}\|F_Q - P_Q\|_{L^p((1+\bar{r})Q\cap\widehat{Q})}\\ &\qquad + \delta_Q^{-\frac{n}{p}}\|F_Q\|_{\mathbb{X}((1+\bar{r})Q\cap\widehat{Q})} + \sum_{|\beta|\le m-1}|\partial^\beta(P_Q - P_{Q'})(y_Q)|\cdot\delta_Q^{|\beta|-m}\Big]\end{aligned}$$

(note that $y_Q \in Q$ and $x \in (1+\bar{r})Q$, hence $|y_Q - x| \le C\delta_Q$; thus, the above inequality follows from Lemma 7).

We now raise each side to the power p, integrate over $\widehat{Q}\cap(1+\frac{\bar{r}}{2})Q'$, and sum over $|\alpha| = m$. Thus we obtain

$$\begin{aligned}\|F\|^p_{\mathbb{X}(\widehat{Q}\cap(1+\frac{\bar{r}}{2})Q')} &\le C(A,\bar{r}) \sum_{\substack{Q\in\mathcal{Q}\\ (1+\bar{r})Q'\cap(1+\bar{r})Q\neq\emptyset}} \Big[\|F_Q\|^p_{\mathbb{X}((1+\bar{r})Q\cap\widehat{Q})}\\ &+ \delta_Q^{-mp}\|F_Q - P_Q\|^p_{L^p((1+\bar{r})Q\cap\widehat{Q})} + \sum_{|\beta|\le m-1}|\partial^\beta(P_Q - P_{Q'})(y_Q)|^p\cdot\delta_Q^{(|\beta|-m)p+n}\Big].\end{aligned}$$

Finally, summing over $Q'\in\mathcal{Q}$, we obtain the conclusion of the lemma, thanks to (4.53) and the bounded overlap and good geometry of $\mathcal{Q}$. □

We resume our discussion in [20].

References

[1] Arya, S., Mount, D., Netanyahu, N., Silverman, R. and Wu, A.: An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *J. ACM* **45** (1998), no. 6, 891–923.

[2] Batson, J., Spielman, D. and Srivastava, N.: Twice-Ramanujan sparsifiers. In *STOC'09 – Proceedings of the 2009 ACM International Symposium on Theory of Computing,* 255–262. ACM, New York, 2009.

[3] Bierstone, E., Milman, P. and Pawłucki, W.: Differentiable functions defined in closed sets. A problem of Whitney. *Invent. Math.* **151** (2003), no. 2, 329–352.

[4] Bierstone, E., Milman, P. and Pawłucki, W.: Higher-order tangents and Fefferman's paper on Whitney's extension problem. *Annals of Math. (2)* **164** (2006), no. 1, 361–370.

[5] Brudnyi, Y. and Shvartsman, P.: A linear extension operator for a space of smooth functions defined on a closed subset of $\mathbb{R}^n$. *Soviet Math. Dokl* **31** (1985), no. 1, 48–51.

[6] Brudnyi, Y. and Shvartsman, P.: Generalizations of Whitney's extension theorem. *Internat. Math. Res. Notices* **1994**, no. 3, 129–139.

[7] Brudnyi, Y. and Shvartsman, P.: The Whitney problem of existence of a linear extension operator. *J. Geom. Anal.* **7** (1997), no. 4, 515–574.

[8] Brudnyi, Y. and Shvartsman, P.: Whitney's extension problem for multivariate $C^{1,\omega}$ functions. *Trans. Amer. Math. Soc.* **353** (2001), no. 6, 2487–2512.

[9] Callahan, P. B. and Kosaraju, S. R.: A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *J. Assoc. Comput. Mach.* **42** (1995), no. 1, 67–90.

[10] Fefferman, C.: A sharp form of Whitney's extension theorem. *Ann. of Math. (2)* **161** (2005), no. 1, 509–577.

[11] Fefferman, C.: Interpolation and extrapolation of smooth functions by linear operators. *Rev. Mat. Iberoamericana* **21** (2009), no. 1, 313–348.

[12] Fefferman, C.: Whitney's extension problem for C^m. *Ann. of Math. (2)* **164** (2006), no. 1, 313–359.

[13] Fefferman, C.: C^m extension by linear operators. *Ann. of Math. (2)* **166** (2007), no. 3, 779–835.

[14] Fefferman, C.: Extension of $C^{m,\omega}$-smooth functions by linear operators. *Rev. Mat. Iberoam.* **25** (2009), no. 1, 1–48.

[15] Fefferman, C.: Fitting a C^m-smooth function to data. III. *Ann. of Math. (2)* **170** (2009), no. 1, 427–441.

[16] Fefferman, C. and Klartag, B.: Fitting a C^m-smooth function to data. I. *Ann. of Math. (2)* **169** (2009), no. 1, 315–346.

[17] Fefferman, C. and Klartag, B.: Fitting a C^m-smooth function to data. II. *Rev. Mat. Iberoam.* **25** (2009), no. 1, 49–273.

[18] Fefferman, C., Israel, A. and Luli, G. K.: Sobolev extension by linear operators. *J. Amer. Math. Soc.* **27** (2014), 69–145.

[19] Fefferman, C., Israel, A. and Luli, G. K.: The structure of Sobolev extension operators. *Rev. Mat. Iberoam.* **30** (2014), no. 2, 419–429.

[20] Fefferman, C., Israel, A. and Luli, G. K.: Fitting a Sobolev function to data II. *Rev. Mat. Iberoam.* (to appear).

[21] Fefferman, C., Israel, A. and Luli, G. K.: Fitting a Sobolev function to data III. *Rev. Mat. Iberoam.* (to appear).

[22] Glaeser, G.: Étude de quelques algèbres tayloriennes. *J. Analyse Math.* **6** (1958), 1–124; erratum, insert to 6 (1958), no. 1.

[23] Hirn, M. and Le Gruyer, E.: A general theorem of existence of quasi absolutely minimal Lipschitz extensions. *Math. Ann.* **359** (2014), no. 3-4, 595–628.

[24] Hartmanis, J. and Simon, J.: On the power of multiplication in random-access machines. In *15th Annual Symposium on Switching and Automata Theory (1974)* 13–23. IEEE Comput. Soc., Long Beach, Calif., 1974.

[25] Israel, A.: A bounded linear extension operator for $L^{2,p}(\mathbb{R}^2)$. *Ann. of Math. (2)* **178** (2013), no. 1, 183–230.

[26] Jones, P. W.: Quasiconformal mappings and extendability of functions in Sobolev spaces. *Acta Math.* **147** (1981), no. 1-2, 71–88.

[27] Luli, G. K.: $C^{m,\omega}$ extension by bounded-depth linear operators. *Adv. Math.* **224** (2010), no. 5, 1927–2021.

[28] Le Gruyer, E.: On absolutely minimizing Lipschitz extensions and PDE $\Delta_\infty(u)=0$. *NoDEA Nonlinear Differential Equations Appl.* **14** (2007), no. 1-2, 29–55.

[29] Le Gruyer, E.: Minimal Lipschitz extensions to differentiable functions defined on a Hilbert space. *Geom. Funct. Anal.* **19** (2009), no. 4, 1101–1118.

[30] Gilbarg, D. and Trudinger, N. S.: *Elliptic partial differential equations of second order.* Springer, 1998.

[31] Schönhage, A.: On the power of random access machines. In *Automata, languages and programming (Sixth Colloq., Graz, 1979),* 520–529. Lecture Notes Comput. Sci. 71, Springer, Berlin, 1979.

[32] Shvartsman, P.: Lipschitz sections of set-valued mappings and traces of functions from the Zygmund class on an arbitrary compactum. *Soviet Math. Dokl.* **29** (1984), no. 3, 565–568.

[33] Shvartsman, P.: Traces of functions of Zygmund class. *Siberian Math. J.* **28** (1987), 853–863.

[34] Shvartsman, P.: Lipschitz selections of set-valued mappings and Helly's theorem. *J. Geom. Anal.* **12** (2002), no. 2, 289–324.

[35] Shvartsman, P.: Sobolev W^1_p-spaces on closed subsets of $\mathbb{R}^n$. *Adv. Math.* **220** (2009), no. 6, 1842–1922.

[36] Shvartsman, P.: On the sum of a Sobolev space and a weighted L_p-space. *Adv. Math.* **248** (2013), 155–228.

[37] Shvartsman, P.: Sobolev L^2_p-functions on closed subsets of $\mathbb{R}^2$. *Adv. Math.* **252** (2014), 22–113.

[38] von Neumann, J.: 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 Ann. Hist. Comput.* **15** (1993), no. 4, 27–75.

[39] Wells, J.: Differentiable functions on Banach spaces with Lipschitz derivatives. *J. Differential Geometry* **8** (1973), 135–152.

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

[41] Whitney, H.: Differentiable functions defined in closed sets. I. *Trans. Amer. Math. Soc.* **36** (1934), no. 2, 369–387.

[42] Whitney, H.: Functions differentiable on the boundaries of regions. *Ann. of Math. (2)* **35** (1934), no. 3, 482–485.

[43] Zobin, N.: Whitney's problem on extendability of functions and an intrinsic metric. *Adv. Math.* **133** (1998), no. 1, 96–132.

[44] Zobin, N.: Extension of smooth functions from finitely connected planar domains. *J. Geom. Anal.* **9** (1999), no. 3, 491–511.

Received November 19, 2014.

Charles Fefferman: Department of Mathematics, Princeton University, Fine Hall, Washington Road, Princeton, NJ 08544, USA.
E-mail: cf@math.princeton.edu

Arie Israel: Department of Mathematics, University of Texas at Austin, RLM Hall, 2515 Speedway, Austin, TX 78712, USA.
E-mail: arie@math.utexas.edu

Garving Luli: Department of Mathematics, University of California at Davis, One Shields Avenue, Davis, CA 95616, USA.
E-mail: kluli@math.ucdavis.edu

Charles Fefferman is supported by NSF grant DMS-1265524 and AFOSR grant FA9550-12-1-0425. Arie Israel is supported by an NSF postdoctoral fellowship, DMS-1103978. Garving Luli is supported by NSF grant DMS-1355968.