Numerical Algorithms
https://doi.org/10.1007/s11075-023-01716-5

ORIGINAL PAPER

On the enumeration of subcells within hypercubes and its application to the Borsuk-Ulam theorem

Moody T. Chu[1] · **Matthew M. Lin**[2]

Received: 27 July 2023 / Accepted: 27 November 2023

Abstract
The conventional triangulation of 2-spheres and subdivision of tetrahedrons in $\mathbb{R}^3$ are difficult to generalize to higher dimensions. The challenge lies in finding a systematic way to characterize each of subcells after the division. This work discusses the dissection of high-dimensional hypercubes and presents a way where all subsequent subcells and their symmetries can be systematically enumerated. Of particular interest is a generic coordinate system that is employed to construct all cells through suitable homeomorphisms. By repeatedly applying this generic coordinate to all cells, multitasking in parallel is possible. On the other hand, the Borsuk-Ulam theorem asserts that every continuous function from an n-sphere into the Euclidian n-space maps at least one pair of antipodal points on the sphere with the same function value. The exquisiteness lies in that only the continuity is assumed in the theorem with yet such profound applications. As an application, this enumeration scheme can be employed to find the Borsuk-Ulam antipodal pair guaranteed without evoking any derivative information for the task. Numerical experiments manifest the effectiveness and potential of this enumeration scheme.

Keywords Borsuk-Ulam theorem · Hypercubes · Generic coordinates · Homeomorphism · Spherical mean

1 Introduction

There are many questions in nature that seem basic to formulate but can be challenging to answer. This work takes on two such problems and proposes to develop one possible

✉ Matthew M. Lin
mhlin@mail.ncku.edu.tw

Moody T. Chu
chu@math.ncsu.edu

[1] Department of Mathematics, North Carolina State University, Raleigh NC 27695-8205, USA

[2] Department of Mathematics, National Cheng Kung University, Tainan 701, Taiwan

Published online: 06 January 2024

approach to answer the first problem, which will then be employed to tackle the second problem.

The first problem evolves from the seemingly straightforward dissection of high-dimensional hypercubes. The challenge is at enumerating the resulting subcells systematically in order to make use of them. Hypercubes are the multi-dimensional extensions of squares and cubes, whose vertices and adjacent facets can easily be identified through sequences of binaries. In contrast to the mechanism that generalizes the conventional bisection method to higher dimensional spaces in [1], which proceeds via a sequence of brackets with infinite intersection is the set of points desired, we offer to carry out the subdivision via symbolic tensor products which are realizable through systematic "foldings" of matrices. The most important contribution is that we offer an effective way to manage the enormous amount of data during the calculation. In particular, we explore how the binary representation gives rise to a natural combinatorial description of the dissection process. By exploring this representation, we uncover further its potential to harness high degrees of parallelism, enhancing both the inherent interest and value of our research endeavor.

The second problem delves into the elegantly simple yet profoundly important Borsuk-Ulam theorem [2, 3]. The challenge is at finding the theoretically guaranteed Borsuk-Ulam antipodal points without relying on any derivative information of the underlying function. To fix the idea, we state the Borsuk-Ulam theorem in its most basic form as follows.

Theorem 1 *Let S^n stand for the unit sphere in $\mathbb{R}^{n+1}$. For every continuous function $f : S^n \to \mathbb{R}^n$, there exists a point $\mathbf{x} \in S^n$ such that $f(\mathbf{x}) = f(-\mathbf{x})$.*

A popular interpretation for the case $n = 2$ of the Borsuk-Ulam theorem is that, if we assume that the distributions of barometric pressure and the temperatures over the surface are continuous respectively, then at any given instant there exists one pair of antipodal places on Earth with the same pressure and the same temperature. The Borsuk-Ulam theorem has found applications in a wide range of disciplines, including, for example, combinatorics [4–6], differential equations [7, 8], geometry [3, 9], social science [10, 11], and so on. See also the 457 references collected in the survey article [12]. It is also remarkable that the Borsuk-Ulam theorem implies other theoretical results such as the Brouwer fixed-point theorem [13, 14], the ham sandwich theorem [15–17], the Lusternik-Schnirelmann theorem [18–20], and the Spencer's lemma [21], each of which is of mathematical interest in its own right with applications to other areas. With such a wide range of consequences in both theory and applications, finding an antipodal pair of points numerically for a given continuous function on the unit sphere is of paramount importance. Since the sphere S^n in the theorem can be replaced by any homeomorphic image of S^n, it suffices to work specifically on the cubical sphere Q^{n+1}, i.e., the hypercube in $\mathbb{R}^{n+1}$ that circumscribes S^n.

If the underlying f is piecewise differentiable, then it is plausible to utilize, say, the conventional projected Newton's method to find the zeros of the odd function $g(\mathbf{x}) := f(\mathbf{x}) - f(-\mathbf{x})$. On the other hand, we are aware of at least two constructive proofs of the Borsuk-Ulam theorems in the literature [7, 22]. Both approaches are similar on the principle of path-following [23, 24]. The difference is that the technique in [22] is applicable only to piecewise linear functions on Q^{n+1} and its path is char-

acterized by a polygon arc generated from a sequence of subdivided triangulations of Q^{n+1}, whereas the path in [7] is defined as the pre-image of 0 of a homotopy map connecting the odd function $g(\mathbf{x})$ to the function of projecting an arbitrary rotation of S^n to $\mathbb{R}^n$. For the former, a scheme similar to the simplex method in linear programming is needed to trace the polygon arc. For the latter, if $f(\mathbf{x})$ is piecewise continuously differentiable, then the one-dimensional homotopy curve can be characterized by a differential equation. These constructive proofs are claimed to be implementable for numerical calculation, but so far as we know, the processes are only axiomatized with no actual experimentation. Different from the methods provided in [7, 22], we purposely challenge ourselves in this paper by insisting on not employing any derivative information at all to find the antipodal points.

One possible way for finding the antipodal pair without utilizing any derivative information of f is the brute force approach by systematically taking discrete samples over the underlying sphere. Our initial idea is motivated by the fact that there are many well-established methods for triangulating the unit sphere S^2 effectively. See, for example, the S2-sampling toolbox [25] written in the Matlab syntax. However, it quickly becomes obvious that the "triangulation" for S^n, $n \geq 3$, is a much more complicated and daunting task for the reason that the notion of triangles imposed on S^2 has to be generalized to polytopes on S^n which have many more edges. When subdivisions are needed repeatedly, tracking newly introduced vertices and facets methodically becomes challenging. Later on, we also realize that, even though the well-known Nelder-Mead simplex search algorithm [26] for multidimensional problems does not require derivative information, the algorithm is designed for unconstrained optimization. Some discussions on extending direct search methods to constrained optimization can be found in the literature [27, 28], but the construction of feasible search directions remains to be a major hurdle.

We therefore embark on the task of searching for the antipodal points on the cubical sphere Q^{n+1}. We follow a systematic two-phase approach. In the initial phase, we locate a possible "cell" where the antipodal points might reside. In the second phase, we conduct a refined search within the identified cell, aiming to attain the desired precision in locating the antipodal points. The notion of spherical means is employed to assess the relevance of a given cell. The main feature in our algorithm is that the same mechanism of using generic coordinates is employed universally to carry out all homeomorphisms. Our methodology encompasses both theoretical and algorithmic aspects, providing a strong and reliable solution to both problems mentioned above. Although our approach might appear somewhat aggressive, it has proven to be a sure-fire technique, producing promising results in solving the Borsuk-Ulam theorem.

This paper is organized as follows. We begin in Sect. 2 with a brief justification on considering the Borsuk-Ulam theorem over the cubical spheres. Then, the idea of using symbolic tensor products to characterize the midpoint subdivision of a hypercube leads to the notion of generic coordinates which will serve as a reference point for all subdivisions. The homeomorphism mapping from the generic coordinates to any given cell is discussed in Sect. 3, which facilitates all future subdivisions while maintaining the orientation. We further propose to represent each cell conveniently by a 1-D array of positive integers, by which the tasks of forming new subcells, exploiting symmetry, and identifying antipodality can be handled effectively. Some numerical examples are

given in Sect. 4, including a mimicker of the continuous everywhere but differentiable nowhere Weierstrass function.

2 Cubical sphere

The notion of hypercubes is a generalization of squares in $\mathbb{R}^2$ and cubes in $\mathbb{R}^3$ to higher dimensional spaces. By a standard hypercube Q^{n+1} we mean the convex hull of points in the form $(\pm 1, \ldots, \pm 1) \in \mathbb{R}^{n+1}$ with all possible sign permutations.

One advantage of using hypercubes is that its vertices can be enumerated systematically. If we replace any -1 in the Euclidean coordinates of a given vertex of Q^{n+1} by 0, then the resulting sequence $(x_n x_{n-1} \ldots x_0)$ can be cast as a binary representation of the vertex which can be enumerated as an integer $k \in [0, 2^{n+1})$ via the formula

$$k = (x_n^{(k)} x_{n-1}^{(k)} \ldots x_0^{(k)})_2 = \sum_{j=0}^{n} x_j^{(k)} 2^j. \tag{1}$$

In this way, we have introduced a specific ordering for the vertices of Q^{n+1}. Keeping this ordering in all subsequent subdivisions through homeomorphisms is critically important.

The convex hull of vertices that have one common coordinate forms a side of the hypercube. The hypercube Q^{n+1} admits $2(n+1)$ hypercubes in its boundary ∂Q^{n+1}, each is homeomorphic to Q^n. Depicted in Fig. 1 are two of the ten sides of Q^5.

Note that all vertices of the left hypercube have a common bit 0 at the first coordinate, while those of the right hypercube have a common bit 1 at the first coordinate. After removing this common bit, both sides can be identified as Q^4. There are eight other "frusta in-between" with one common bit at other coordinates. The 5-dimensional hypercube Q^5 is made of drawing an additional 16 edges connecting each vertex on the left Q^4 to the corresponding vertex on the right Q^4 that differs by the first bit.

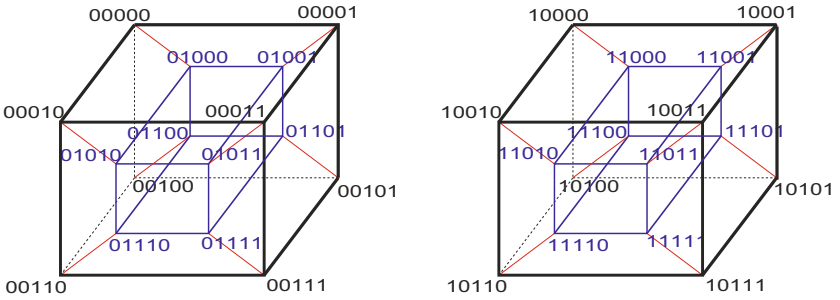

Fig. 1 Vertices and adjacent edges of two of the ten sides of Q^5

2.1 Equivalence

Obviously, the boundary of Q^{n+1} circumscribes S^n. Given $f : S^n \to \mathbb{R}^n$, define $g : \partial Q^{n+1} \to \mathbb{R}^n$ by

$$g(\mathbf{z}) := f(\frac{\mathbf{z}}{\|\mathbf{z}\|_2}). \tag{2}$$

If $g(\mathbf{z}) = g(-\mathbf{z})$ for some $\mathbf{z} \in \partial Q^{n+1}$, then $f(\mathbf{w}) = f(-\mathbf{w})$ with $\mathbf{w} := \frac{\mathbf{z}}{\|\mathbf{z}\|_2} \in S^n$. Conversely, if $f(\mathbf{w}) = f(-\mathbf{w})$ for some $\mathbf{w} \in S^n$, then $g(\mathbf{z}) = g(-\mathbf{z})$ with $\mathbf{z} := \frac{\mathbf{w}}{\|\mathbf{w}\|_\infty} \in \partial Q^{n+1}$. It is in this sense that, without loss of generality, our goal is to find the antipodal points on the boundary of Q^{n+1} to satisfy the Borsuk-Ulam theorem.

2.2 Subdivision mechanism

Each side of Q^{n+1} is a hypercube with $n2^{n-1}$ edges. Two vertices on the hypercube Q^{n+1} are adjacent to each other if their binary sequences differ by one and only one bit. Taking the middle point of each edge and making appropriate connections should divide the side into 2^n sub-hypercubes that are topologically equivalent to Q^n. We shall refer to each of these sub-hypercubes obtained from the dissection of a given hypercube as a cell. The challenge is at maintaining the consistency of ordering as that of the original Q^n and systematically identifying the vertices of each of the cells in all subsequent divisions.

We first outline the general procedure in words. Then, we describe the mathematical tools to carry out this procedure. The progression of complexity sketched in Fig. 2 should help assimilate the ideas.

1. Without loss of generality, assume that the vertices of the side being considered are indexed in the binary form according to those of the standard Q^n.
2. Any two adjacent vertices differ by one bit. Represent the middle point of the connecting edge by replacing that distinct bit with the symbol "a."
3. Connect any two midpoints differing by one binary bit with a new edge. Represent the middle point of the new edge by replacing that distinct bit with the symbol "a." By the Varignon Theorem, the midpoint of the new edge is independent of which edge is used. See the middle graph in Fig. 2 for Q^2.
4. Continue this procedure until all midpoints are labeled. See the right graph in Fig. 2 for Q^3.
5. The vertices of each of the 2^n cells can be identified in exactly the same way as those of the standard hypercube Q^n, except that the binary representations are replaced by the alphanumeric representations.

To carry out the subdivisions in general, we now introduce the idea of symbolic tensor product. First, bisect Q^1 into two parts with the middle point a. Denote the column vectors

$$V_1 := \begin{bmatrix} 0 \\ a \\ 1 \end{bmatrix}, \quad \mathbf{u} := \begin{bmatrix} 0 \\ a \end{bmatrix}, \quad \mathbf{v} := \begin{bmatrix} a \\ 1 \end{bmatrix},$$

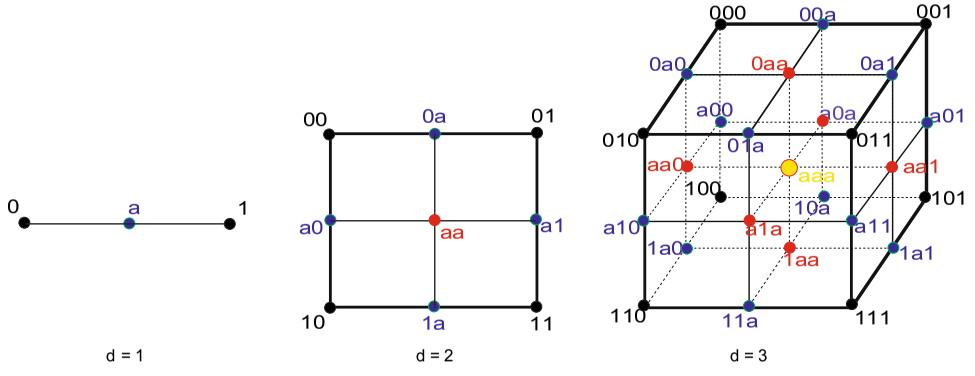

Fig. 2 Enumeration of vertices after subdividing Q^d, $d = 1, 2, 3$, at midpoints along edges. Single "a" indicates midpoints (blue dots) along exiting edges, double "aa" indicates midpoints (red dots) of the edge connecting midpoints, and so on

where entries of V_1 are the collection of vertices, including the midpoint, and $\mathbf{u}$ and $\mathbf{v}$ represent the two subdivisions of Q^1. The subdivisions of Q^2 shown in the middle graph of Fig. 2 with a total of 3^2 vertices can be identified via the symbolic tensor product[1]

$$V_2 := V_1 \otimes V_1 = V_1^{\otimes 2} \equiv \begin{bmatrix} 00 & 0a & 01 \\ a0 & aa & a1 \\ 10 & 1a & 11 \end{bmatrix}.$$

There are 2^2 cells, which can be represented through the four 2×2 matrices

$$\mathbf{u} \otimes \mathbf{u}, \quad \mathbf{u} \otimes \mathbf{v}, \quad \mathbf{v} \otimes \mathbf{u}, \quad \mathbf{v} \otimes \mathbf{v},$$

respectively. If we introduce an artificial bilinear operator $\boxplus$ that satisfies the distributive property with respect to $\otimes$ in the sense that

$$(\mathbf{u} \boxplus \mathbf{v})^{\otimes 2} = (\mathbf{u} \boxplus \mathbf{v}) \otimes (\mathbf{u} \boxplus \mathbf{v}) := (\mathbf{u} \otimes \mathbf{u}) \boxplus (\mathbf{u} \otimes \mathbf{v}) \boxplus (\mathbf{v} \otimes \mathbf{u}) \boxplus (\mathbf{v} \otimes \mathbf{v}),$$

then the cells of Q^2 are the four terms in the expansion of $(\mathbf{u} \boxplus \mathbf{v})^{\otimes 2}$. In a similar vein, the 3^3 vertices for the subdivision of Q^3 in the right graph of Fig. 2 can be obtained via the order-3 tensor product

$$V_3 := V_1 \otimes V_2 = V_1^{\otimes 3},$$

whereas the 2^3 cells can be identified via the binomial expansion $(\mathbf{u} \boxplus \mathbf{v})^{\otimes 3}$. Since the midpoints on each edge will separate the original vertices from each other, each cell contains one and only one original vertex. For example, the cell $\mathbf{v}^{\otimes 3}$ has vertices in

$$\mathbf{v}^{\otimes 3} = \begin{bmatrix} a & a & a & a & 1 & 1 & 1 & 1 \\ a & a & 1 & 1 & a & a & 1 & 1 \\ a & 1 & a & 1 & a & 1 & a & 1 \end{bmatrix},$$

which contains only 111 as the original vertex. Note that the symbol "a" is only a generic indicator of a midpoint whose value depends on the true coordinates of the corresponding endpoints. We shall describe a homeomorphism in Sect. 3.1 to calculate the true values.

For high-dimensional problems, it becomes increasingly more difficult to conjure up the image of all midpoints and newly created edges. For example, one side of Q^5 is equivalent to Q^4 with 32 edges. A midpoint subdivision of Q^4 will produce 2^4 cells. Each cell has 2^4 vertices. A sketch of further subdivisions would be more confusing, if possible at all. We really should rely on a mathematical way to characterize the new vertices and the new subcells of the subdivisions. The mechanism outlined above can serve exactly that purpose.

[1] The tensor product is not the Kronecker product, though in the literature often the same notation is used for both. Suppose we use $\circ$ for the tensor product and $\otimes$ for the Kronecker product, and assumes the commutativity of multiplications among scalars or symbols, then $\mathbf{vec}(\mathbf{u} \circ \mathbf{v}) = \mathbf{v} \otimes \mathbf{u} = \mathbf{vec}(\mathbf{u}\mathbf{v}^\top)$. Relevant to this paper, however, is that we do not assume the commutativity of multiplications among the alphanumeric bits. The tensor product $\mathbf{u} \otimes \mathbf{v}$ (more indicatively, $\mathbf{u} \circ \mathbf{v}$ or $\mathbf{u}\mathbf{v}^\top$), therefore, yields the set of ordered pairs $\{0a, aa, 01, a1\}$, which represents vertices for the cell at the upper-right corner in the middle graph of Fig. 2.

2.3 Generic coordinates

The enumeration of the 2^n vertices of Q^n based on their binary digits sets a standard orientation of the hypercube. From there, one subdivision of Q^n contains 3^n vertices representable by $V^{\otimes n}$ and 2^n cells representable by $(\mathbf{u} \boxplus \mathbf{v})^{\otimes n}$, all of which can also be enumerated systematically. A typical vertex is made of n alphanumeric values 0, a, and 1, which will be called the generic coordinates. Being able to systematically and automatically identify vertices and group them into relevant cells while keeping their orientation is the main contribution of this paper.

Furthermore, the boundary ∂Q^{n+1} is made of $2(n+1)$ sides each of which is topologically equivalent to Q^n. Every subcell obtained from a subdivision of any cell at any level remains equivalent to Q^n. The only difference is that the vertices of different cells are located at places with different coordinates. Since the calculation for each cell as well as its successive subdivisions is similar, it suffices to build one mechanism of subdivision for the standard hypercube Q^n, and use it repeatedly whenever a subdivision is needed. Once we know how the generic coordinates of Q^n in $\mathbb{R}^n$ are mapped to the true coordinates of each cell of ∂Q^{n+1} in $\mathbb{R}^{n+1}$, the calculation is embarrassingly parallelizable. It will be seen in the subsequent discussion that the homeomorphism is in fact easy to obtain.

The procedure can be applied repeatedly to all sides of Q^{n+1} to generate a finer mesh on ∂Q^{n+1}. Actually, because of the symmetry, it suffices to work with $n+1$ sides. The goal is to locate one patch or several patches of interest where the antipodal points might reside. This is the initial phase. The same procedure can then be applied repeatedly to the patch of interest, which is also equivalent to Q^n, to zoom in the search for the antipodal points. This is the second phase. Both phases employ the same procedure.

3 Implementation

We have argued in the above that the symbolic tensor products can facilitate the representations of the vertices and the subdivided hypercubes. In practice, we have to know what the symbol a stands for in real values. Also, individual variables in most computing devices, including symbolic variables, are typically assumed to be scalars whose multiplications are commutative. We cannot use the Kronecker product directly to simulate the tensor product. In our application, the point with coordinates $0a$ is different from that with $a0$ and the tensor product certainly cannot be simplified to 0. In this section, we propose a numerical algorithm to work around this hurdle, to distinguish the values of a, and to effectively identify the needed homeomorphism.

3.1 Midpoint evaluation

We first explain the underlying homeomorphism between a given cell and the generic coordinates. We then use the generic coordinates to calculate the true midpoints of the given cell and complete one subdivision.

Let the columns of $C = [\mathbf{c}_0, \mathbf{c}_1, \ldots \mathbf{c}_{2^n-1}] \in \mathbb{R}^{(n+1)\times 2^n}$ denote the true coordinates of 2^n points in $\mathbb{R}^{n+1}$. These points can be the vertices of the top cell, i.e., one side of the original Q^{n+1}, or those of any of the smaller subcells after several levels of subdivisions. Since C originates from one particular side of Q^{n+1}, one row of C must have the same value of either 1 or -1, regardless of the levels of subdivisions. This unique marker can be regarded as a group identification. Inherited in C is that its columns are specially ordered, which is passed down from the very top cell with the same group ID and is maintained throughout all subdivisions. That is, the generic coordinates corresponding to the kth column of C, $k = 0, \ldots, 2^n - 1$, are precisely the binary representation of the integer k over the vertices of Q^n. This defines a homeomorphism between Q^n and the convex hull of C.

Recall that $V_1^{\otimes n}$ represents one subdivision of Q^n. Suppose now that C is to be subdivided by taking midpoints at its edges. Our goal is to evaluate the true coordinates of points corresponding to $V_1^{\otimes n}$. The task can be accomplished as follows. For each given generic point $\mathbf{g} = [g_i] \in \{0, a, 1\}^n$ in $V_1^{\otimes n}$, partition the indices $1, \ldots, n$ into three disjoint sets

$$\mathcal{Z}(\mathbf{g}) := \{i_\alpha | g_{i_\alpha} = 0\}, \quad \mathcal{A}(\mathbf{g}) := \{i_\beta | g_{i_\beta} = a\}, \quad \mathcal{O}(\mathbf{g}) := \{i_\gamma | g_{i_\gamma} = 1\},$$

where some sets might be empty. While keeping $g_{i_\alpha} = 0$ for $i_\alpha \in \mathcal{Z}$ and $g_{i_\gamma} = 1$ for $i_\gamma \in \mathcal{O}$, replace a by 0 and 1 for each $i_\beta \in \mathcal{A}$ to create a total $2^{|\mathcal{A}|}$ new binary arrays, where $|\mathcal{A}|$ denotes the cardinality of $\mathcal{A}$. Convert the resulting binary arrays to integers $k_1, \ldots, k_{2^{|\mathcal{A}|}}$. Then, corresponding to the generic coordinates $\mathbf{g}$, the true coordinates of the point in the convex hull of C are given by

$$\mathbf{g} \longmapsto \frac{1}{2^{|\mathcal{A}|}} \sum_{j=1}^{2^{|\mathcal{A}|}} \mathbf{c}_{k_j}. \tag{3}$$

In this way, the cell C of 2^n vertices is dissected into 2^n subcells with a total of 3^n vertices. Once these subcells are identified, they can be further subdived.

3.2 Cell formation

We have already discussed that the subcells after one division of Q^n can be obtained via the expansion of $(\mathbf{u} \boxplus \mathbf{v})^{\otimes n}$, and that $V_1^{\otimes n}$ specifies all vertices in an ordered manner. Each cell, therefore, can be conveniently represented by a 1-D array of 2^n positive integers which are ordinal numbers of their vertices in the list of $V_1^{\otimes n}$.

We begin with the example of ∂Q^4 which has eight sides. Each row of the matrix

$$\partial Q^4 = \begin{bmatrix} 0 & 2 & 4 & 6 & 8 & 10 & 12 & 14 \\ 1 & 3 & 5 & 7 & 9 & 11 & 13 & 15 \\ 0 & 1 & 4 & 5 & 8 & 9 & 12 & 13 \\ 2 & 3 & 6 & 7 & 10 & 11 & 14 & 15 \\ 0 & 1 & 2 & 3 & 8 & 9 & 10 & 11 \\ 4 & 5 & 6 & 7 & 12 & 13 & 14 & 15 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \end{bmatrix},$$

denotes one side of Q^4. The eight vertices of one side are identified by the integer converted from the binary coordinates of vertices in the original Q^4. See Fig. 1. In general, the boundary ∂Q^{n+1} can be represented by a $2(n+1) \times 2^n$ integer matrix. These rows of ∂Q^{n+1} are easy to form because each side has only one common coordinate of its vertices.

The symbolic tensor product $V_1^{\otimes n}$ can be achieved by repeatedly and systematically copying the vector V_1 to interleaved blocks. This undertaking can be effectively implemented. See, for example, the command combvec in Matlab. However, to avoid evoking the symbolic math engine, and also to allow a convenient arithmetic conversion which will be explained later, we make an artificial substitution

$$0 \to 1 \quad a \to 2 \quad 1 \to 3,$$

where the choice of the digits $\{1, 2, 3\}$ is immaterial. Instead of writing $V_n := V_1^{\otimes n}$ as an order-n tensor whose entries are made of arrays of n alphanumeric bits in $\{0, a, 1\}$, we represent V_n as a matrix of size $n \times 3^n$ whose entries are from the set $\{1, 2, 3\}$. Instead of using the generic coordinates, the ordinal numbers of columns of V_n form a natural way to enumerate the mesh points in the subdivision of Q^n. Instead of keeping the full coordinates, we may use the ordinal numbers of V_n to identify vertices of the subcells.

With the above substitution, the first step is to replace $\mathbf{u} \boxplus \mathbf{v}$ by the 1×4 row vector

$$U_1 := [1, 2, 2, 3]\,,$$

so the information contained in $(\mathbf{u} \boxplus \mathbf{v})^{\otimes n}$ is precisely the same as that in $U_n := U_1^{\otimes n}$ which, similar to V_n, is obtained by repeatedly copying U_1 to interleaved blocks, i.e., the tensor product $U_1^{\otimes n}$ can be represented by a matrix U_n of size $n \times 4^n$. For example, $U_1^{\otimes 3}$ leads to the 3×4^3 matrix

$$U_3 = \begin{bmatrix} 1223122312231223122312231223122312231223122312231223122312231223 \\ 1111222222222333311112222222233331111222222223333111122222222333 \\ 1111111111111111222222222222222222222222222222223333333333333333 \end{bmatrix}.$$

Note the specific structure manifested in U_3 by the action of the interleaved block copying. Exploiting this regular pattern enables us to identify the vertices of individual subcells by simply "folding" this matrix properly. Each of the 4^n columns of U_n corresponds to one mesh point in V_n. Identifying the mesh points by the column numbers in V_n, we can represent U_n by a 1-D array Υ_n of integers in the range $[1, 3^n]$. Checking the membership of U_n in V_n can be done as follows. Let $\mathbf{p}$ denote an n-dimensional vector of large random prime numbers. We encode each vertex $\mathbf{v}$ in V_n in a tag number

$$t(\mathbf{v}) := \mathbf{p}^\top \mathbf{v}.$$

If the prime numbers are sufficiently far apart, then $t(\mathbf{v})$ is unique. In this way, we can effectively generate Υ_n by checking the membership of $t(U_n)$ against $t(V_n)$. For instance, the above U_3 can be identified as the integer array

$$\begin{array}{r} [\ 1\ \ 2\ \ 2\ \ 3\ \ 4\ \ 5\ \ 5\ \ 6|\ \ 4\ \ 5\ \ 5\ \ 6\ \ 7\ \ 8\ \ 8\ \ 9|\ 10\ 11\ 11\ 12\ 13\ 4\ 14\ 15| \\ \Upsilon_3 = 13\ 14\ 14\ 15\ 16\ 17\ 17\ 18|\ 10\ 11\ 11\ 12\ 13\ 14\ 14 15|13\ 14\ 14\ 15\ 16\ 17\ 17\ 18| \\ 19\ 20\ 20\ 21\ 22\ 23\ 23\ 24|\ 22\ 23\ 23\ 24\ 25\ 26\ 26\ 27\ ] \end{array}$$

We have used the delimiter "|" to separate the entries into blocks of eight to prepare for forming cells that will be explained below.

To completely characterize one subdivision of Q^n, the 4^n integers in Υ_n must be regrouped into 2^n subcells. The task can be accomplished by systematically merging and folding Υ_n, which we outline below. The rationale behind is nothing but a mechanical way to track the correspondence between combvec and the expansion of $(\mathbf{u} \boxplus \mathbf{v})^{\otimes n}$. The procedure involves no floating-point arithmetic.

1. Partition Υ_n from left to right into blocks with block size 1×8. See the example Υ_3. Swap entries $\{3, 4\}$ with $\{5, 6\}$ in each block. Such a swap is to mechanically fix an aftereffect due to the way the interleaving works. Denote the resulting blocks as $B_1, \ldots, B_{2^{2n-3}}$.
2. Arrange the blocks row-wise as a cell array

$$W_n := \begin{bmatrix} B_1 & B_2 & B_3 & B_4 \\ B_5 & B_6 & B_7 & B_8 \\ \vdots & & & \vdots \\ B_{2^{2n-3}-3} & & & B_{2^{2n-3}} \end{bmatrix}.$$

   That is, W_n has $2^{2n-5} \times 4$ containers, each with block size 1×8. This matrix will be futher folded in the next step.
3. Assuming the relevant sizes are compatible, introduce the three commands for convenience:

$$\begin{aligned} \text{reshape(X,M,N)} &= \text{rearrange elements from } X \text{ columnwise to an } M \times N \text{ matrix,} \\ \text{cell2mat(C)} &= \text{convert a multidimensional cell array } C \text{ to a single matrix,} \\ \text{mat2tiles(X,M,N)} &= \text{break } X \text{ with adjacent chunks of size } M \times N \text{ into a cell array.} \end{aligned}$$

The rearrangement of W_n can be described algorithmically as follows:

```
if n = 3
   W_3 = mat2tiles(cell2mat(W_3),2,8);
   W_3 = mat2tiles(cell2mat(reshape(W_3,1,[])),2,4);
elseif n > 3
   for iwrap = 1:n-3
      fold = 2^(iwrap+1);
      W_n = reshape(W_n',fold,[])';
      W_n = mat2tiles(cell2mat(W_n),fold,8);
```

```
    end
    W_n = mat2tiles(cell2mat(reshape(W_n,1,[])),
fold,4);
end
```

At the end, W_n consists of 2^n blocks with block size $2^{n-2} \times 4$. Each block contains 2^n vertices of one subcell. Each vertex corresponds to one mesh point in V_n.

4. Reshaping each block of the final W_n into a row, we obtain a $2^n \times 2^n$ matrix F_n of integers. The entries in each row of F_n point to which vertices in V_n are to be gathered to form one cell after one subdivision at the midpoints of Q^n, and maintain consistently the orientation of the original Q^n. We shall call F_n the cell ID of Q^n after one subdivision.

As an example, each side of ∂Q^6 is equivalent to Q^5 whose subdivision contains 32 cells. Each row of the 32×32 matrix F_5 below represents the vertices of one cell. The integers in a row correspond to the positions of columns in V_5.

$$
\begin{bmatrix}
1 & 10 & 28 & 37 & 82 & 91 & 109 & 118 & 2 & 11 & 29 & 38 & 83 & 92 & 110 & 119 & 4 & 13 & 31 & 40 & 85 & 94 & 112 & 121 & 5 & 14 & 32 & 41 & 86 & 95 & 113 & 122 \\
2 & 11 & 29 & 38 & 83 & 92 & 110 & 119 & 3 & 12 & 30 & 39 & 84 & 93 & 111 & 120 & 5 & 14 & 32 & 41 & 86 & 95 & 113 & 122 & 6 & 15 & 33 & 42 & 87 & 96 & 114 & 123 \\
82 & 91 & 109 & 118 & 163 & 172 & 190 & 199 & 83 & 92 & 110 & 119 & 164 & 173 & 191 & 200 & 85 & 94 & 112 & 121 & 166 & 175 & 193 & 202 & 86 & 95 & 113 & 122 & 167 & 176 & 194 & 203 \\
83 & 92 & 110 & 119 & 164 & 173 & 191 & 200 & 84 & 93 & 111 & 120 & 165 & 174 & 192 & 201 & 86 & 95 & 113 & 122 & 167 & 176 & 194 & 203 & 87 & 96 & 114 & 123 & 168 & 177 & 195 & 204 \\
4 & 13 & 31 & 40 & 85 & 94 & 112 & 121 & 5 & 14 & 32 & 41 & 86 & 95 & 113 & 122 & 7 & 16 & 34 & 43 & 88 & 97 & 115 & 124 & 8 & 17 & 35 & 44 & 89 & 98 & 116 & 125 \\
5 & 14 & 32 & 41 & 86 & 95 & 113 & 122 & 6 & 15 & 33 & 42 & 87 & 96 & 114 & 123 & 8 & 17 & 35 & 44 & 89 & 98 & 116 & 125 & 9 & 18 & 36 & 45 & 90 & 99 & 117 & 126 \\
85 & 94 & 112 & 121 & 166 & 175 & 193 & 202 & 86 & 95 & 113 & 122 & 167 & 176 & 194 & 203 & 88 & 97 & 115 & 124 & 169 & 178 & 196 & 205 & 89 & 98 & 116 & 125 & 170 & 179 & 197 & 206 \\
86 & 95 & 113 & 122 & 167 & 176 & 194 & 203 & 87 & 96 & 114 & 123 & 168 & 177 & 195 & 204 & 89 & 98 & 116 & 125 & 170 & 179 & 197 & 206 & 90 & 99 & 117 & 126 & 171 & 180 & 198 & 207 \\
10 & 19 & 37 & 46 & 91 & 100 & 118 & 127 & 11 & 20 & 38 & 47 & 92 & 101 & 119 & 128 & 13 & 22 & 40 & 49 & 94 & 103 & 121 & 130 & 14 & 23 & 41 & 50 & 95 & 104 & 122 & 131 \\
11 & 20 & 38 & 47 & 92 & 101 & 119 & 128 & 12 & 21 & 39 & 48 & 93 & 102 & 120 & 129 & 14 & 23 & 41 & 50 & 95 & 104 & 122 & 131 & 15 & 24 & 42 & 51 & 96 & 105 & 123 & 132 \\
91 & 100 & 118 & 127 & 172 & 181 & 199 & 208 & 92 & 101 & 119 & 128 & 173 & 182 & 200 & 209 & 94 & 103 & 121 & 130 & 175 & 184 & 202 & 211 & 95 & 104 & 122 & 131 & 176 & 185 & 203 & 212 \\
92 & 101 & 119 & 128 & 173 & 182 & 200 & 209 & 93 & 102 & 120 & 129 & 174 & 183 & 201 & 210 & 95 & 104 & 122 & 131 & 176 & 185 & 203 & 212 & 96 & 105 & 123 & 132 & 177 & 186 & 204 & 213 \\
13 & 22 & 40 & 49 & 94 & 103 & 121 & 130 & 14 & 23 & 41 & 50 & 95 & 104 & 122 & 131 & 16 & 25 & 43 & 52 & 97 & 106 & 124 & 133 & 17 & 26 & 44 & 53 & 98 & 107 & 125 & 134 \\
14 & 23 & 41 & 50 & 95 & 104 & 122 & 131 & 15 & 24 & 42 & 51 & 96 & 105 & 123 & 132 & 17 & 26 & 44 & 53 & 98 & 107 & 125 & 134 & 18 & 27 & 45 & 54 & 99 & 108 & 126 & 135 \\
94 & 103 & 121 & 130 & 175 & 184 & 202 & 211 & 95 & 104 & 122 & 131 & 176 & 185 & 203 & 212 & 97 & 106 & 124 & 133 & 178 & 187 & 205 & 214 & 98 & 107 & 125 & 134 & 179 & 188 & 206 & 215 \\
95 & 104 & 122 & 131 & 176 & 185 & 203 & 212 & 96 & 105 & 123 & 132 & 177 & 186 & 204 & 213 & 98 & 107 & 125 & 134 & 179 & 188 & 206 & 215 & 99 & 108 & 126 & 135 & 180 & 189 & 207 & 216 \\
28 & 37 & 55 & 64 & 109 & 118 & 136 & 145 & 29 & 38 & 56 & 65 & 110 & 119 & 137 & 146 & 31 & 40 & 58 & 67 & 112 & 121 & 139 & 148 & 32 & 41 & 59 & 68 & 113 & 122 & 140 & 149 \\
29 & 38 & 56 & 65 & 110 & 119 & 137 & 146 & 30 & 39 & 57 & 66 & 111 & 120 & 138 & 147 & 32 & 41 & 59 & 68 & 113 & 122 & 140 & 149 & 33 & 42 & 60 & 69 & 114 & 123 & 141 & 150 \\
109 & 118 & 136 & 145 & 190 & 199 & 217 & 226 & 110 & 119 & 137 & 146 & 191 & 200 & 218 & 227 & 112 & 121 & 139 & 148 & 193 & 202 & 220 & 229 & 113 & 122 & 140 & 149 & 194 & 203 & 221 & 230 \\
110 & 119 & 137 & 146 & 191 & 200 & 218 & 227 & 111 & 120 & 138 & 147 & 192 & 201 & 219 & 228 & 113 & 122 & 140 & 149 & 194 & 203 & 221 & 230 & 114 & 123 & 141 & 150 & 195 & 204 & 222 & 231 \\
31 & 40 & 58 & 67 & 112 & 121 & 139 & 148 & 32 & 41 & 59 & 68 & 113 & 122 & 140 & 149 & 34 & 43 & 61 & 70 & 115 & 124 & 142 & 151 & 35 & 44 & 62 & 71 & 116 & 125 & 143 & 152 \\
32 & 41 & 59 & 68 & 113 & 122 & 140 & 149 & 33 & 42 & 60 & 69 & 114 & 123 & 141 & 150 & 35 & 44 & 62 & 71 & 116 & 125 & 143 & 152 & 36 & 45 & 63 & 72 & 117 & 126 & 144 & 153 \\
112 & 121 & 139 & 148 & 193 & 202 & 220 & 229 & 113 & 122 & 140 & 149 & 194 & 203 & 221 & 230 & 115 & 124 & 142 & 151 & 196 & 205 & 223 & 232 & 116 & 125 & 143 & 152 & 197 & 206 & 224 & 233 \\
113 & 122 & 140 & 149 & 194 & 203 & 221 & 230 & 114 & 123 & 141 & 150 & 195 & 204 & 222 & 231 & 116 & 125 & 143 & 152 & 197 & 206 & 224 & 233 & 117 & 126 & 144 & 153 & 198 & 207 & 225 & 234 \\
37 & 46 & 64 & 73 & 118 & 127 & 145 & 154 & 38 & 47 & 65 & 74 & 119 & 128 & 146 & 155 & 40 & 49 & 67 & 76 & 121 & 130 & 148 & 157 & 41 & 50 & 68 & 77 & 122 & 131 & 149 & 158 \\
38 & 47 & 65 & 74 & 119 & 128 & 146 & 155 & 39 & 48 & 66 & 75 & 120 & 129 & 147 & 156 & 41 & 50 & 68 & 77 & 122 & 131 & 149 & 158 & 42 & 51 & 69 & 78 & 123 & 132 & 150 & 159 \\
118 & 127 & 145 & 154 & 199 & 208 & 226 & 235 & 119 & 128 & 146 & 155 & 200 & 209 & 227 & 236 & 121 & 130 & 148 & 157 & 202 & 211 & 229 & 238 & 122 & 131 & 149 & 158 & 203 & 212 & 230 & 239 \\
119 & 128 & 146 & 155 & 200 & 209 & 227 & 236 & 120 & 129 & 147 & 156 & 201 & 210 & 228 & 237 & 122 & 131 & 149 & 158 & 203 & 212 & 230 & 239 & 123 & 132 & 150 & 159 & 204 & 213 & 231 & 240 \\
40 & 49 & 67 & 76 & 121 & 130 & 148 & 157 & 41 & 50 & 68 & 77 & 122 & 131 & 149 & 158 & 43 & 52 & 70 & 79 & 124 & 133 & 151 & 160 & 44 & 53 & 71 & 80 & 125 & 134 & 152 & 161 \\
41 & 50 & 68 & 77 & 122 & 131 & 149 & 158 & 42 & 51 & 69 & 78 & 123 & 132 & 150 & 159 & 44 & 53 & 71 & 80 & 125 & 134 & 152 & 161 & 45 & 54 & 72 & 81 & 126 & 135 & 153 & 162 \\
121 & 130 & 148 & 157 & 202 & 211 & 229 & 238 & 122 & 131 & 149 & 158 & 203 & 212 & 230 & 239 & 124 & 133 & 151 & 160 & 205 & 214 & 232 & 241 & 125 & 134 & 152 & 161 & 206 & 215 & 233 & 242 \\
122 & 131 & 149 & 158 & 203 & 212 & 230 & 239 & 123 & 132 & 150 & 159 & 204 & 213 & 231 & 240 & 125 & 134 & 152 & 161 & 206 & 215 & 233 & 242 & 126 & 135 & 153 & 162 & 207 & 216 & 234 & 243
\end{bmatrix}
$$

Since each cell of Q^n is homeomorphic to Q^n itself, and since we know their exact locations in the space, the same algorithm can be applied repeatedly to subdivide the cells for further refinement, i.e., each cell can be subdivided into another 2^n subcells. Let $V_n^{(k)}$ and $F_n^{(k)}$, $k = 1, 2, \ldots$, denote the set of vertices and cell IDs after Q^n has been subdivided k times. In the above example, therefore, $F_5^{(1)} = F_5$. Because we use the same machinery to produce subcells, we know how the subcells in $F_n^{(k)}$ are stratified one layer after another.

3.3 Symmetry

Since ∂Q^{n+1} is centrally symmetric, we should exploit the symmetry inherent in both the vertices and the cells throughout the subdivision. By doing so, we can effectively cut the calculation by half. The symmetry inherited in $V_n^{(1)}$ and $F_n^{(1)}$ is easy to discern, but after Q^n has been subdivided k times, tracking the symmetry is not that obvious. Our way of enumerating $V_n^{(k)}$ and $F_n^{(k)}$ identifies symmetry easily.

For convenience, we store the data after k subdivisions in a single structure array Qd with the field containers to distinguish the data, that is, $Qd.X = V_n^{(k)}$ and $Qd.F = F_n^{(k)}$. We exploit the stratification in $Qd.F$ by folding and flipping as follows to obtain $F1$ and $F2$ as antipodal cells, and columns in the matrices $X1$ and $X2$ as antipodal vertices.

```
FF = reshape(Qd.F',(2^n)^(k+1)),[]);
FF(:,2:2:end) = flipud(FF(:,2:2:end));
F1 = reshape(FF(:,1:2:end),2^n,[])';
F2 = reshape(FF(:,2:2:end),2^n,[])';

BB = sort(reshape(FF',2,[])',2);
Anti = unique(BB,'rows','stable');
X1 = Qd.X(:,Anti(:,1));
X2 = Qd.X(:,Anti(:,2));
```

3.4 Spherical mean

Assign the weight $\omega(\mathbf{x}) := \|f(\mathbf{x}) - f(-\mathbf{x})\|$ to each point $\mathbf{x} \in \partial Q^{n+1}$. We use $\omega(\mathbf{x})$ to help search for an approximate location of the Borsuk-Ulam points. Ideally, we should look for the place where $\omega(\mathbf{x}) = 0$. Since we only have a discrete approximation of ∂Q^n, we shall settle for $\omega(\mathbf{x}) \le \epsilon$ with a preselected tolerance ϵ.

Suppose that $C = [\mathbf{c}_0, \mathbf{c}_1, \ldots \mathbf{c}_{2^n-1}] \in \mathbb{R}^{(n+1)\times 2^n}$ denote a typical cell on ∂Q^{n+1} after several levels of subdivisions. The average weight of the cell C is given by

$$\frac{1}{\text{Volume of } C} \int_C \omega(\mathbf{x})\, \mathrm{d}V(\mathbf{x}) \tag{4}$$

which is analogous to the notion of spherical mean [29–31]. We look for the few cells that have relatively lighter weights than others and then continue to refine the cells by further subdivisions. The rationale is that, by continuity, the values of $\omega(\mathbf{x})$ at points nearby the Borsuk-Ulam points should also be nearly zero. The proper size of the neighboring cell, of course, is problem-dependent. If the mesh is not fine enough or if the weights at the vertices of one specific cell vary drastically, it is possible that we might miss the correct cell. To safeguard against this possible failure, we choose in the initial phase a few extra antipodal cells, and apply the refinement process to all of them.

Since all cells at the same level of subdivision have the same volume, we only need to compute the integral which can be approximated by the Riemann sum. To better approximate the Riemann sum, we use our subdivision technique to dissection C a few more levels deeper and compare the sums of $\omega(\mathbf{x})$ at all vertices within the cell C with its peers. These extra dissections are not wasted because they can be reused for the next subdivision.

3.5 Overhead count

While the subdivision mechanism described above for generating mesh points involves mostly combinatorial manipulations and very few actual computations, the algorithm does generate an enormous amount of data.

To subdivide Q^3, we find that the total number of distinct vertices generated in $V_3^{(k)}$ is given by exactly $3(2^{2k+1})+2$, whereas for the case Q^4, there will be $4(2^{3k+1}+2^{k+1})$ distinct vertices in $V_4^{(k)}$. For Q^n in general, the cardinality of points in $V_n^{(k)}$ grows like $O(n2^{(n-1)k+1})$. A numerical example is given in Sect. 4.

3.6 Code organization

The above discussion can be put together to become an algorithm which we summarize below.

Algorithm 1 Finding Borsuk-Ulam Points

REQUIRE: Create an homeomorphism, if necessary, define the objective function $f : \partial Q^{n+1} \to \mathbb{R}^n$.
ENSURE: Antipodal points where $f(\mathbf{x}) \approx f(-\mathbf{x})$.

1: Create and store the cell information of ∂Q^{n+1} in Qd.
2: Create the main mechanism "Cube Divider" using generic coordinates.
3: Apply the "Cube Divider" to divide ∂Q^{n+1} k_1 times and store the cell information in $Qd1$. ▷ [can be done in parallel]
4: Multitasking: Use the procedure described in Section 3.3 to identify antipodal cells F_1 and F_2, denoted by the cell IDs b_1 and b_2, respectively, and copy $Qd1$ to $Qd3$ to prepare for the local search.
5: Multitasking: Apply the "Cube Divider" to subdivide $Qd1$ k_2 times and store the cell information in $Qd2$ to prepare for spherical means calculation.
6: Evaluate $f(Q2.F)$ and patch groups into S_d. ▷ [can be done in parallel]
7: Identify the patches with minimal spherical means and integrate the corresponding patches that make up the cells in $Qd3$.
8: Apply the "Cube Divider" to subdivide the potential cell $Qd3$ k_3 times.
9: Test whether the "meshsize" is less than ϵ. Either stop the iteration or redefine $Qd3$ as $Qd1$ and return to Step 3. ▷ [sizes are much smaller]

It might be more informative to also outline the algorithm in the flowchart depicted in Fig. 3. A recurring theme throughout the calculation is the call for the subroutine "Cube Divider" which is the apparatus of applying the generic coordinates to the current cells. We also denote in the flowchart the two phases of calculation which essentially embody the same procedure by using different sets of data marked in the structure array $Qd1$. In the global search, $Qd1$ contains all data after ∂Q^{n+1} is

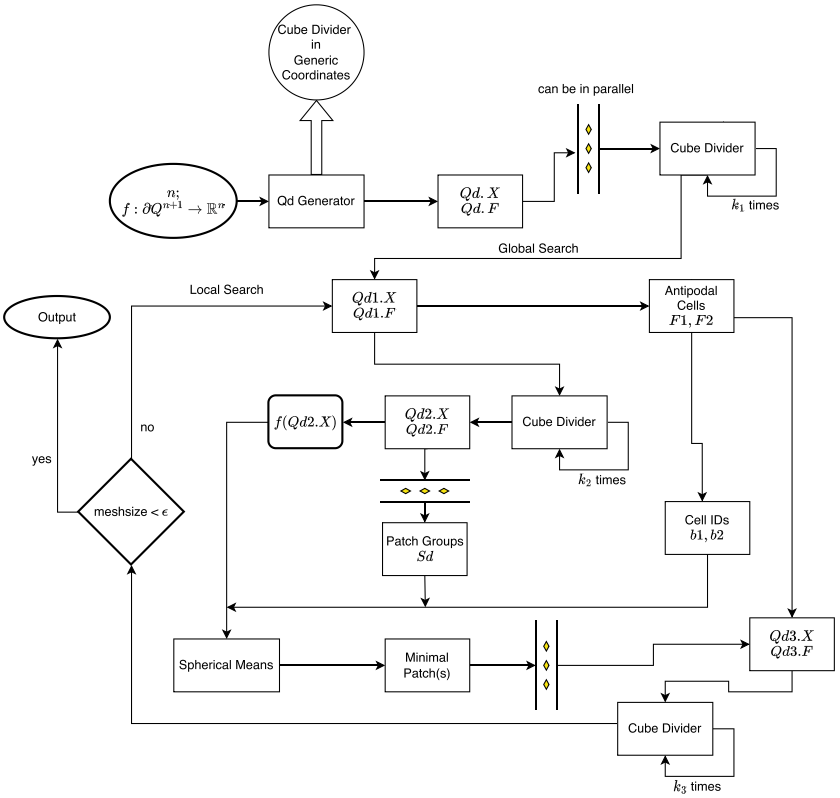

Fig. 3 Algorithm flowchart

subdivided k_1 times. In the local search, $Qd1$ contains data after one patch, identified as $Qd3$, is subdivided k_3 times. The data needed in the local search is much smaller than that in the global search, as will be demonstrated in Table 1 in our numerical experiments. The subdivision of $Qd1$ to $Qd2$ is for the purpose of approximating the spherical means for cells in $Qd3$.

At three places in the flowchart, we mark that multitasking is possible because the data involved can be divided into independent work. For example, when generating data from Qd to $Qd1$ by dividing ∂Q^{n+1}, the task can be accomplished by working with separate sides. How the parallelism is carried out depends on what comprises the parallel device and how the compiler handles the multitasking. In our numerical experiments, we use merely the available Matlab Parallel Computing Toolbox on an 8-core machine.

The few snippets in Sect. 3.2 demonstrate how the generic coordinates are formed by matrix folding, which is the essence of the routine "Cube Divider," whereas the cell IDs, $b1$ and $b2$, of the antipodal cells $F1$ and $F2$, together with the patch group IDs Sd are integer arrays for helping the calculation of the spherical means. The actual program we have developed for the following experiments is much more sophisticated. For interested readers, we will make our code available upon request.

4 Numerical experiments

In this section, we apply our dissection method to some selected functions and report the empirical results.

Example 1 Though our emphasis in this work is on not utilizing any derivative information, a test on smooth functions has the advantage of using other techniques to produce a more accurate approximation to the true solution, which can be used as a basis of comparison with our brute force algorithm. Consider the case of a high-order polynomial system $f : \partial Q^3 \to \mathbb{R}^2$ where

$$f(\mathbf{x})=\begin{bmatrix} x_1^2+x_2^4+x_1x_2x_3+x_1x_2+x_2x_3-x_1-1-x_2-x_3 \\ x_2^{12}+4x_2^{11}+6x_2^{10}+4x_2^9+x_2^8+((x_3^3+x_3^2+1)(x_1^3+x_1^2)+x_3^3+x_3^2-1)(x_2^3+x_2^2)+x_1^6+2x_1^5+x_1^4-x_1^3-x_3^3-x_1^2-x_3^2 \end{bmatrix}.$$

It can be checked that the zero set of the odd function $g(\mathbf{x}) = f(\mathbf{x}) - f(-\mathbf{x})$ is a 1-dimensional algebraic variety in $\mathbb{R}^3$. When restricted to ∂Q^3, there is one unique antipodal pair. We find by the Newton method that the antipodal pair is given by

$$\widetilde{\mathbf{x}} := [\pm 1, \mp 0.8570772441361144, \mp 0.07696112604641343]^\top$$

which almost satisfy the Borsuk-Ulam theorem with $\omega(\widetilde{\mathbf{x}}) \approx 1.8310 \times 10^{-15}$.

For the global search, we subdivide each side four times with mesh size 2^{-3}. Using the spherical mean, we find one subcell that happens to contain $\widetilde{\mathbf{x}}$ in its interior. Upon further refinement, we locate our numerical solution $\mathbf{x}^*$ with mesh size at approximately 2^{-51} and final $\omega(\mathbf{x}^*) \approx 3.2790 \times 10^{-11}$. Subdividing the hypercube 52 times to reach nearly machine precision seems expensive, but the dissection process is applied to one patch only each time and the technique described in this paper makes the bookkeeping effectively.

Example 2 The Weierstrass function defined by the Fourier series

$$h(x; a, b) := \sum_{k=0}^{\infty} a^k \cos(b^k \pi x), \tag{5}$$

where (a, b) are fixed parameters with $0 < a < 1$ and b being a positive odd integer satisfying $ab > 1 + \frac{3}{2}\pi$, is a classical example of a continuous everywhere but differentiable nowhere function. This Fourier series converges uniformly on $\mathbb{R}$. It is not possible to visualize this function precisely even on high-resolution computers.

Take $a = .9$ and $b = 7$. Consider the partial sum

$$\overline{h}(x) := \sum_{k=0}^{7} .9^k \cos(7^k \pi x) + p(x), \quad x \in [-1, 1],$$

where the term $p(x)$ is added to break the evenness of the cosine function. Although $\overline{h}(x)$ is continuously differentiable, we use its high-oscillation behavior to simulate the

non-differentiability of $h(x; a, b)$. For our test, we consider the function $f : \partial Q^3 \to \mathbb{R}$ defined by a mixture of $\overline{h}(x)$ via

$$f(\mathbf{x}) = (\overline{h}(x_1) + sin(x_2))(\overline{h}(x_2) + x_3^3).$$

This function f is scalar-valued, so it is not in the full form of the Borsuk-Ulam theorem. Still, antipodal points satisfying $f(\mathbf{x}) = f(-\mathbf{x})$ should exist. This particular $f(\mathbf{x})$ is a complicated function that cannot be easily resolved. The left graph in Fig. 4 is a low-resolution plot (at 50 evaluation points per direction) of the set $\Xi := \{\mathbf{x} \in Q^3 | f(\mathbf{x}) = f(-\mathbf{x})\}$. This stalactitic and stalagmitic surface is not even close to revealing the delicate details. We are only interested in the solutions on the boundary ∂Q^3. The right graph in Fig. 4 is a high-resolution plot (at 5000 evaluation points per direction) of Ξ at the face $z = 1$. The bottom graph is a high-resolution plot of Ξ at $y = -1$. Needless to say, there are lots of antipodal solutions on ∂Q^3 to satisfy the Borsuk-Ulam theorem.

Using the same initial mesh size 2^{-3} as in Example 1 and the minimal spherical mean criteria, we find an approximate solution

$$\mathbf{x}^* \approx [\pm 0.513602491897643, \pm 0.555251817860182, \pm 1.000000000000000]^\top$$

with final ω value at approximately 6.8593×10^{-9}. Using a smaller initial mesh size 2^{-4} for the global search, we find another approximate solution

$$\mathbf{x}^* \approx [\mp 0.140994477289453, \mp 0.858994588808685, \pm 1.000000000000000]^\top$$

with $\omega(\mathbf{x}^*) \approx 5.7130 \times 10^{-9}$.

Example 3 On a broader scope, we may interpret our algorithm as a means for finding the global minimum of a continuous, nonnegative, and even function $\omega : \partial Q^{n+1} \to \mathbb{R}$. The extension of the well-known Nelder-Mead simplex search algorithm for this purpose is not very satisfactory because determining the search direction and proving the convergence are challenging, as has been discussed in [27], and, even more disappointingly, gives rise to only a local minimum. In contrast, our brute force approach deals specifically with the constraint ∂Q^{n+1} and is a sure-fire method.

Consider the minimization of $\omega(\mathbf{x}) = |p(\mathbf{x})|$ and its minimizer(s) over the compact set ∂Q^5 where $p(\mathbf{x})$ is a randomly generated odd polynomial given by:

$$\begin{aligned} p(\mathbf{x}) = {} & 8{x_3}^9{x_4}^2 + 24x_2{x_3}^6{x_4}^2 + (2{x_5}^2 + 2){x_3}^5 + (2{x_1}^3 + 2x_1){x_3}^4 \\ & + (2{x_1}^3{x_5}^3 + 2x_1{x_5}^3 + 8{x_4}^6 + (24{x_2}^2 + 2){x_4}^2 + 2{x_5}^2 - 2){x_3}^3 \\ & + (2{x_4}^2{x_5}^3 + 4{x_1}^3 + 2x_2{x_5}^2 + 4x_1 + 2x_2)x_3^2 + (2{x_1}^3x_2 \\ & + 2x_1x_2 + 2{x_4}^2 - 2)x_3 + ((2{x_5}^2 + 2){x_4}^2 + 2x_2{x_5}^3 - 2)({x_1}^3 + x_1) \\ & + 8x_2{x_4}^6 + (8{x_2}^3 + 2{x_5}^3){x_4}^2 + 2x_2{x_5}^2 - 2{x_5}^3 - 2x_2. \end{aligned}$$

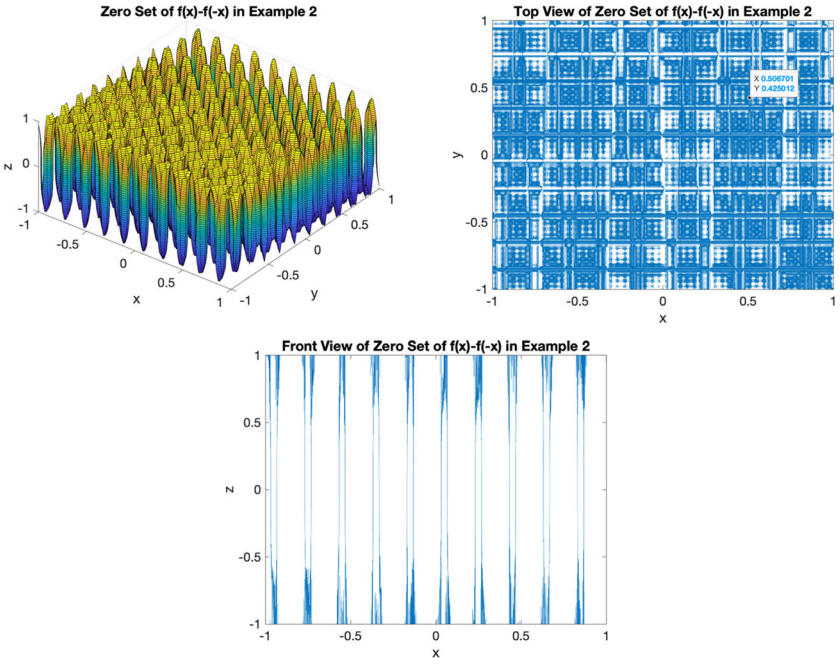

Fig. 4 Solutions to Example 2

Using our algorithm, we do find an approximate minimizer at

$$\begin{aligned}\mathbf{x}^* \approx [&\pm 0.625005779438879, \mp 0.749996406646233,\\ &\pm 0.539866285713742, \mp 0.187325815015356,\\ &\mp 1.000000000000000]^\top\end{aligned}$$

with $\omega(\mathbf{x}^*) \approx 1.2335 \times 10^{-13}$.

Tabulated in Table 1 is a possible configuration of our division mechanism and the corresponding sizes of data generated in trying to solve the problem. In the left table at level $k_1 = 0$, ∂Q^5 has 2^5 vertices and 10 sides to begin with. Each side is equivalent to a hypercube Q^4 whose subdivision at the midpoint will generate 2^4 subcells. Therefore, the number of cells grows exponentially as $10(2^4)^{k_1}$. Suppose that we have subdivided ∂Q^5 three times, i.e., $k_1 = 3$. There will be 40960 candidates of subcells for further refinement. However, recall that all cells are identified by integers and that we can enumerate them systematically for further subdivision if so needed.

A different strategy is to work with one side at a time and choose the potentially best side. In this way, the number of subcells for each side is reduced by a factor of 10. Taking into account the symmetry, we can further reduce the number of cells by a factor of 2. The work on each side is independent of each other and, hence, can be executed in parallel.

Table 1 Overhead at different configurations of subdivisions

Global Search			Spherical Mean		
level k_1	# vertices	# cells	level k_2	# vertices	# subcells
0	32	10			
1	242	160			
2	2882	2560	Subdivide for Spherical Means		
			1	42242	40960
			2	660482	655360
3	42242	40960	Subdivide for Spherical Means		
			1	660482	655360
			2	10506242	10485760
4	660482	655360	Subdivide for Spherical Means		
			1	10506242	10485760

Local Search		
level k_3	# vertices	# cells
0	32	2
1	162	32
2	1250	512
3	13122	8192
4	167042	131072

To choose a potential subcell, we subdivide each subcell k_2 times and compare the approximate spherical means. Once a patch is selected for refinement, the same strategy can be applied repeatedly. In each cycle of the local search, we start with 2^5 vertices and 2 cells in tandem of symmetry, which is much cheaper than the global search.

In this experiment, we have chosen $k_1 = 4$, $k_2 = 1$, $k_3 = 3$, and repeated the local search 45 times. The final mesh size is approximately 3.5526×10^{-15} and an approximate solution with reasonable precision is found.

5 Conclusion

We propose a brute force algorithm to find the antipodal points on the boundary of the hypercube Q^{n+1} promised by the Borsuk-Ulam theorem. Only the continuity of the underlying function is used. In order to accomplish this task, we must be able to enumerate all cells, their vertices, and their orientation. The main contribution of this paper is to offer a way to overcome this challenge. As with all direct search algorithms, an enormous amount of points will be needed for comparison. However, by taking advantage of the binary representations of vertices, we argue that just one dissection of Q^n is enough to serve as a universal mechanism of generic coordinates, and that this universal frame can be constructed via appropriate foldings of the data matrix. Any further dissections can be obtained with appropriate homeomorphisms. Numerical experiments suggest the effectiveness and potential of this enumeration scheme.

Acknowledgements The second author would like to express gratitude to the National Center for Theoretical Sciences of Taiwan and the Miin Wu School of Computing at the National Cheng Kung University for their valuable assistance in research.

Author contribution Conceptualization, MTC and MML; methodology, MTC; software, MTC and MML; validation, MTC and MML; formal analysis, MTC and MML; investigation, MTC; resources, MTC; data curation, MTC; writing—original draft preparation, MTC; writing—review and editing, MML; visualiza-

tion, MTC; supervision, MTC; project administration, MTC; funding acquisition, MTC and MML. All authors have read and agreed to the published version of the manuscript.

Funding The first author was supported in part by the National Science Foundation under grants DMS-1912816 and DMS-2309376. The second author was supported in part by the National Center for Theoretical Sciences of Taiwan and by the National Science and Technology Council of Taiwan under grants 112-2636-M-006-002, 112-2628-M-006-009-MY4, and 112-2119-M-006-004.

Data availability The Matlab code associated with the algorithm proposed in this paper can be made available upon request.

Declarations

Ethical approval Not applicable

Conflict of interest The authors declare no competing interests.

References

1. Wood, G.R.: The bisection method in higher dimensions. Math. Program. **55**(1), 319–337 (1992). https://doi.org/10.1007/BF01581205
2. Borsuk, K.: Drei sätze über die n-dimensionale euklidische sphäre. Fundam. Math. **20**(1), 177–190 (1933)
3. Yamabe, H., Yujobô, Z.: On the continuous function defined on a sphere. Osaka Math. J. **2**, 19–22 (1950)
4. Matoušek, J.: Using the Borsuk-Ulam Theorem. Universitext, p. 196. Springer, (2003). Lectures on topological methods in combinatorics and geometry
5. Müller, T., Stehlík, M.: Generalised Mycielski graphs and the Borsuk-Ulam theorem. Electron. J. Comb. **26**, 4–8 (2019)
6. Roy, S., Steiger, W.: Some combinatorial and algorithmic applications of the Borsuk-Ulam theorem. Graphs Combin. **23**(suppl. 1), 331–341 (2007). https://doi.org/10.1007/s00373-007-0716-1
7. Alexander, J.C., Yorke, J.A.: The homotopy continuation method: numerically implementable topological procedures. Trans. Amer. Math. Soc. **242**, 271–284 (1978). https://doi.org/10.2307/1997737
8. Fenn, R.: Some generalizations of the Borsuk-Ulam theorem and applications to realizing homotopy classes by embedded spheres. Proc. Cambridge Philos. Soc. **74**, 251–256 (1973). https://doi.org/10.1017/s0305004100048040
9. Kakutani, S.: A proof that there exists a circumscribing cube around any bounded closed convex set in R^3. Ann. of Math. **2**(43), 739–741 (1942). https://doi.org/10.2307/1968964
10. Deligkas, A., Fearnley, J., Melissourgos, T., Spirakis, P.G.: Computing exact solutions of consensus halving and the Borsuk-Ulam theorem. J. Comput. System Sci. **117**, 75–98 (2021). https://doi.org/10.1016/j.jcss.2020.10.006
11. Simmons, F.W., Su, F.E.: Consensus-halving via theorems of Borsuk-Ulam and Tucker. Math. Soc. Sci. **45**(1), 15–25 (2003). https://doi.org/10.1016/S0165-4896(02)00087-2
12. Steinlein, H.: Borsuk's antipodal theorem and its generalizations and applications: a survey. In: Topological Methods in Nonlinear Analysis. Sém. Math. Sup., vol. 95, pp. 166–235. Presses Univ. Montréal, Montreal, QC, (1985)
13. Su, F.E.: Borsuk-Ulam implies Brouwer: a direct construction. Amer. Math. Monthly **104**(9), 855–859 (1997). https://doi.org/10.2307/2975293
14. Volovikov, A.Y.: Borsuk-Ulam implies Brouwer: a direct construction revisited. Amer. Math. Monthly **115**(6), 553–556 (2008). https://doi.org/10.1080/00029890.2008.11920563
15. Felsner, S., Pilz, A.: Ham-sandwich cuts for abstract order types. Algorithmica **80**(1), 234–257 (2018). https://doi.org/10.1007/s00453-016-0246-4
16. Suciu, A., Fries, M.: The Borsuk-Ulam theorem and applications. Tapas seminar, Northeastern University, (2005). https://doi.org/10.13140/RG.2.2.20061.51687

17. Karthik, C.S., Saha, A.: Ham sandwich is equivalent to Borsuk-Ulam. In: 33rd International Symposium on Computational Geometry. LIPIcs. Leibniz Int. Proc. Inform., vol. 77, pp. 24–15. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, (2017)
18. Bollobás, B.: The art of mathematics, p. 359. Cambridge University Press, New York, (2006). https://doi.org/10.1017/CBO9780511816574. https://doi-org.prox.lib.ncsu.edu/10.1017/CBO9780511816574
19. Lusternik, L.A., Schnirelmann, L.G.: Méthodes Topologiques dans les Problèmes Variationnels vol. 188. Hermann & cie, (1934)
20. Oprea, J.: Applications of Lusternik-Schnirelmann category and its generalizations. J. Geom. Symmetry Phys. **36**, 59–97 (2014)
21. Nyman, K.L., Su, F.E.: A Borsuk-Ulam equivalent that directly implies Sperner's lemma. Amer. Math. Monthly **120**(4), 346–354 (2013). https://doi.org/10.4169/amer.math.monthly.120.04.346
22. Meyerson, M.D., Wright, A.H.: A new and constructive proof of the Borsuk-Ulam theorem. Proc. Amer. Math. Soc. **73**(1), 134–136 (1979). https://doi.org/10.2307/2042898
23. Eaves, B.C.: A short course in solving equations with PL homotopies. In: Nonlinear Programming (Proc. SIAM-AMS Sympos., NewYork, 1975), pp. 73–143 (1976)
24. Eaves, B.C., Scarf, H.: The solution of systems of piecewise linear equations. Math. Oper. Res. **1**(1), 1–27 (1976). https://doi.org/10.1287/moor.1.1.1
25. Semechko, A.: Suite of functions to perform uniform sampling of a sphere. MIT, (2021). MIT. Package available at https://github.com/AntonSemechko/S2-Sampling-Toolbox
26. Nelder, J.A., Mead, R.: A simplex method for function minimization. Comput. J. **7**(4), 308–313 (1965). https://doi.org/10.1093/comjnl/7.4.308
27. Kolda, T.G., Lewis, R.M., Torczon, V.: Optimization by direct search: new perspectives on some classical and modern methods. SIAM Rev. **45**(3), 385–482 (2003). https://doi.org/10.1137/S003614450242889
28. Wright, M.H.: Direct search methods: once scorned, now respectable. In: Numerical Analysis 1995 (Dundee, 1995). Pitman Res. Notes Math. Ser., vol. 344, pp. 191–208. Longman, Harlow, (1996)
29. Finch, D.: Rakesh: the spherical mean value operator with centers on a sphere. Inverse Prob. **23**(6), 37–49 (2007). https://doi.org/10.1088/0266-5611/23/6/S04
30. Görner, T., Hielscher, R., Kunis, S.: Efficient and accurate computation of spherical mean values at scattered center points. Inverse Probl. Imaging **6**(4), 645–661 (2012). https://doi.org/10.3934/ipi.2012.6.645
31. Langer, T., Belyaev, A., Seidel, H.-P.: Mean value coordinates for arbitrary spherical polygons and polyhedra in $\mathbb{R}^3$. In: Curve and Surface Design: Avignon 2006. Mod. Methods Math., pp. 193–202. Nashboro Press, Brentwood, TN, (2007)