2-Polytopes Arthur Mills
Feb. 19, 2023

---

1 Introduction

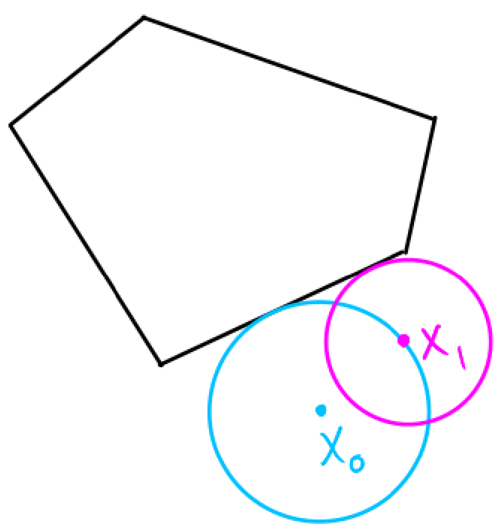
Let M be a two-dimensional convex and compact polytope in $\mathbb{R}^2$. We will also assume that M is *non-degenerate*, in the sense that the internal angle at each vertex is less than π.

In this problem, the polytope M is unknown. We are given an *oracle* that returns the signed (minimum) distance from a point $x \in \mathbb{R}^2$ to the boundary of M. That is, our oracle is the function

$$\begin{aligned} \mathcal{F} : \mathbb{R}^2 &\to \mathbb{R} \\ x &\mapsto d(x, \partial M) \end{aligned}$$

where $d : \mathbb{R}^2 \to \mathbb{R}$ is the signed Euclidean distance function, where distance is positive for points outside M and negative for points inside M. Our goal is to use the oracle to identify the unknown polytope M.

The overall strategy will be to pick sample points in $\mathbb{R}^2$ and evaluate their distance from the polytope using the oracle. Given a point $x_i \in \mathbb{R}^2$, we will write $R_i = \mathcal{F}(x_i)$ as its distance from M. We then have a circle S_i centered at x_i with radius R_i that contains exactly one point of M.



Proposition 1. For any $x_i \in \mathbb{R}^2$, the circle S_i centered at x_i with radius R_i intersects M in exactly one point.

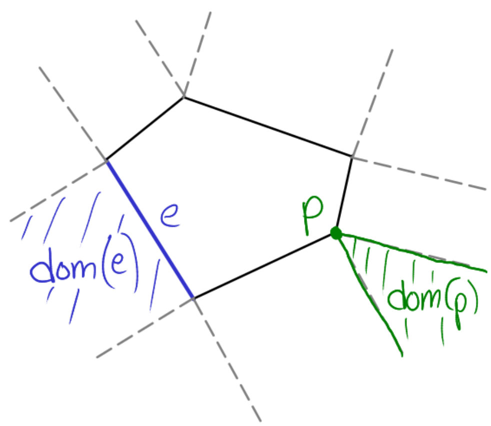
Proof. Recall that the radius R_i of S_i is constructed to be the minimum distance from x_i to the polytope M. Suppose S_i contains two points p and q of M. Since M is convex, M contains the straight line segment ℓ containing p and q. But ℓ is a chord in S_i and so contains points of M closer to x_i than p and q, which is a contradiction. □

2 Identifying Points and Edges

Definition 2.1. Let e be an edge of the polytope M. The **domain** $\operatorname{dom}(e)$ of e is the set of points in $\mathbb{R}^2$ whose minimum distance to M is positive and is equal to the distance to e. That is,

$$\operatorname{dom}(e) = \{x \in \mathbb{R}^2 \mid \mathcal{F}(x) = d(x, e) \text{ and } \mathcal{F}(x) > 0\}.$$

The **domain** $\operatorname{dom}(p)$ of a point p is defined similarly.

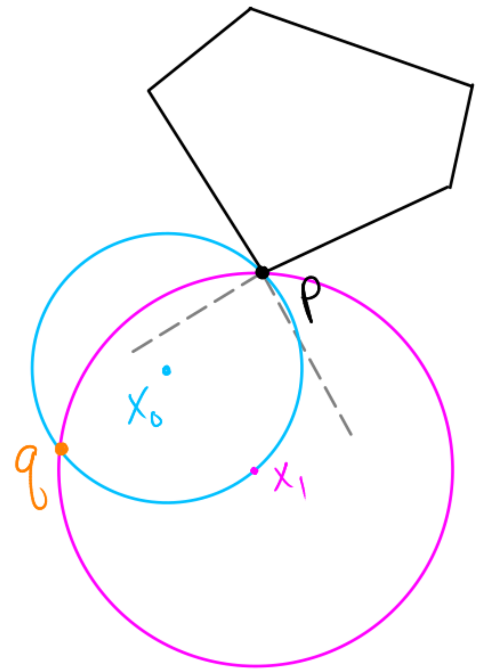


We will view each edge e of M as an open line segment. The domain of e is then an open subset of $\mathbb{R}^2$ bounded by e and the two lines normal to e starting at the endpoints of e. The domain of a point p is the closed cone bounded between the lines normal to the two edges containing p.

In what follows, we will adopt the following strategy for selecting sample points in $\mathbb{R}^2$. First we will select an arbitrary point x_0 that lies outside M, so $R_0 > 0$. Using the oracle, we consider the circle S_0 that touches the unknown polytope M at exactly one point. We will randomly choose a point x_1 that lies on S_0.

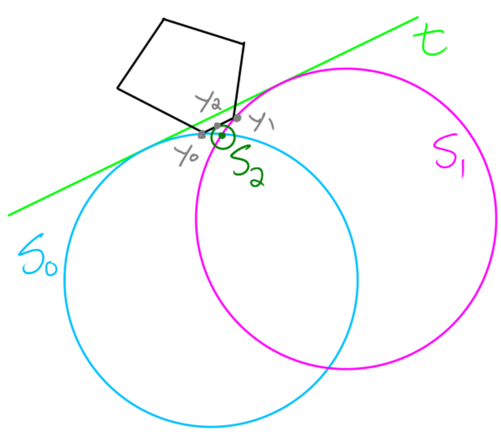
The geometry of the two circles S_0 and S_1 will give us clues as to which domains the points are in. Note that since $x_1 \in S_0$, the circles S_0 and S_1 will always intersect in at least one point.

Proposition 2. Let $x_0, x_1 \in \mathbb{R}^2$ with $x_1 \in S_0$. Suppose that $S_0 \cap S_1 = \{p, q\}$ where $p \neq q$. If $\mathcal{F}(p) = 0$ then p must be a vertex of M.



Proof. Suppose that $\mathcal{F}(p) = 0$ but p is not a vertex of M, and so lies on an edge e. By *Proposition 1* p is the only point of M that lies on both S_0 and S_1. The edge e is then tangent to both S_0 and S_1 at p. But two intersecting circles that share a tangent line at a point must either be equal or share one intersection point. In either case we have a contradiction. $\square$

Proposition 3. Let $x_0, x_1 \in \mathbb{R}^2$ with $x_1 \in S_0$ and suppose $S_0 \cap S_1 = \{p, q\}$ where $p \neq q$ and $\mathcal{F}(p) < \mathcal{F}(q)$. Let ℓ be the common tangent line to both S_0 and S_1 closest to p. Then there exists at least one sub-segment of an edge of M contained within the region R bounded by ℓ, S_0, and S_1 (including possibly the boundary of this region).



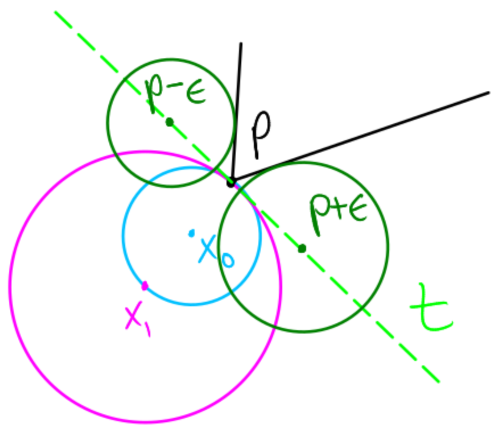
Proof. By *Proposition 1*, M intersects each of S_0, S_1, and S_2 in exactly one point. Say that S_i intersects M at y_i and suppose that y_2 lies outside the region R. Since M is a convex polygon, there is a sequence of edges from y_0 to y_2 and from y_2 to y_1. But if y_2 lies outside of R, this sequence of edges must cross ℓ more than twice. Since any line that crosses through the interior of a convex polygon intersects the edges of the polygon exactly twice, this is a contradiction. $\square$

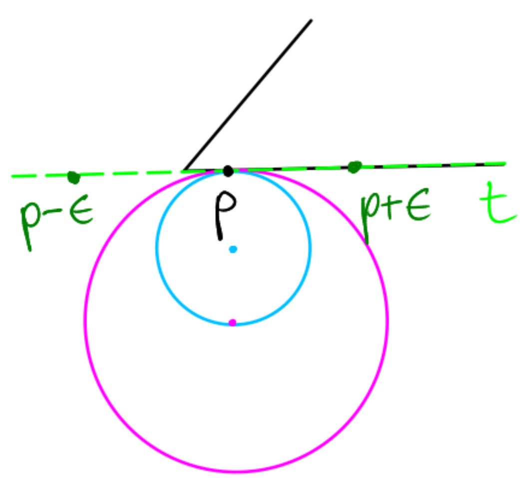
Let $\epsilon > 0$ be a "small" user-defined parameter that will represent the accuracy of the accuracy tolerance of the algorithm. We will utilize ϵ in such a way that edges of M that have length smaller than ϵ will not be found.

Our first step in identifying portions of the boundary of M is to first find any single point on the boundary. This initial stage of the algorithm reduces to one of several cases.

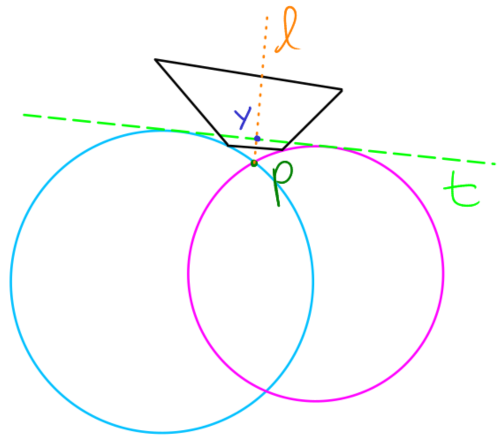
Case 1. Suppose that $S_0 \cap S_1 = \{p\}$. Then if p is interior to an edge e, that edge is determined by the line t containing p that is orthogonal to the line containing x_0 and x_1. If p is instead a vertex, we may still use the line t to determine the adjacent edges.

We may take the line t as an axis of a coordinate system with p at the origin and pick two points $p + \epsilon$ and $p - \epsilon$ that are a distance of ϵ from p along t. If $\mathcal{F}(p \pm \epsilon) = 0$, then that point lies exactly on an edge determined by the line containing p and $p \pm \epsilon$. Otherwise, under the assumption that no edges of M have length smaller than ϵ, the points $p \pm \epsilon$ are in the domains of edges containing p. Therefore we may find an edge to be the line containing p that is tangent to $S_{p\pm\epsilon}$.

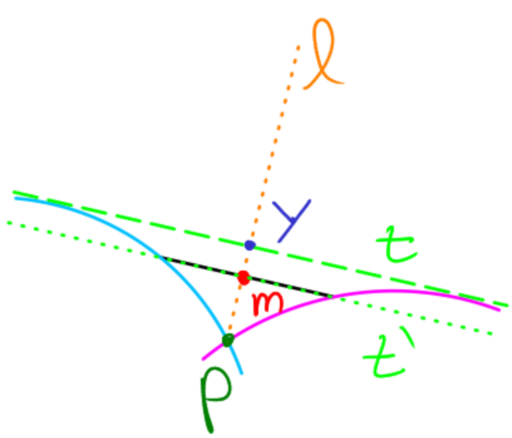




Case 2. Suppose now that $S_0 \cap S_1 = \{p, q\}$ where $p \neq q$, and say that $\mathcal{F}(p) = 0$. Then we know that p is a vertex of M by *Proposition 2*, and we seek to find the edges containing p. Let ℓ_0 be the line tangent to S_0 containing p and choose a point $p + \epsilon_0$ on ℓ_0 a distance ϵ away from p in the direction away from S_1. Then we may find an edge to be the line containing p that is tangent to $S_{p+\epsilon_0}$. Similarly, we may find the other edge containing p.

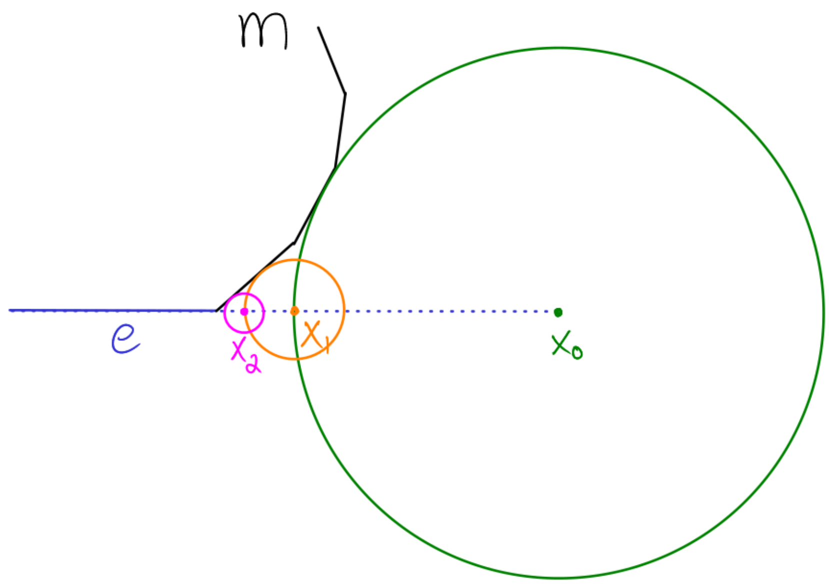


Case 3. Assume that $S_0 \cap S_1 = \{p, q\}$ where $p \neq q$ and $\mathcal{F}(p) < \mathcal{F}(q)$. By *Proposition 3* there exists a sub-segment of an edge of M contained within the region R bounded by S_0, S_1, and the common tangent line t to both S_0 and S_1. Consider the line ℓ through p that is orthogonal to t and let y be the intersection of t and ℓ. Note that by convexity of M, ℓ intersects M in either one or two points. We aim to find the closest intersection point to p. This can be done by a linear or binary search using the oracle along the line ℓ, starting at y. Note that $\mathcal{F}(y) = 0$ when the tangent line t contains an edge of M. Once the closest intersection point z is found, we choose the axis to be aligned with a line t' parallel to t that passes through z. We may then find the edges containing/adjacent to z by stepping along the axis by a distance of $\pm\epsilon$, as in the previous cases.



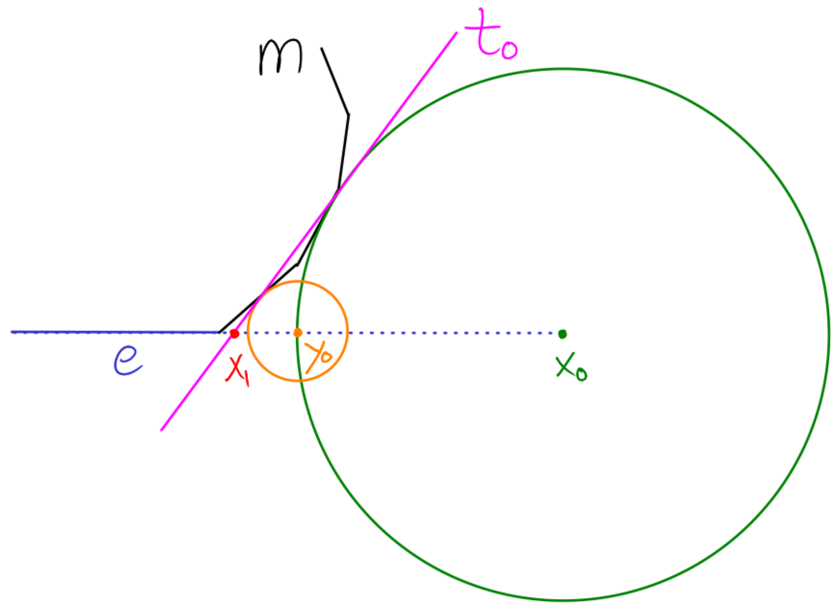
The above algorithm determines an edge of the polygon M given any starting position x_0 outside of M. The next step is to discover the endpoints of a given edge. We currently have four algorithms for determining an endpoint. For the algorithms below, suppose that we have identified that an edge e lies on the line ℓ. We will first find a point x_0 on ℓ that does not lie on e by sampling points along ℓ with a given step size, until the oracle returns a value greater than zero. Our goal is to find the endpoint p of e closest to x_0.

Linear Search. Since $\mathcal{F}(x_0) > 0$ we have a circle centered at x_0 that intersects the polytope M at some point. Note that this intersection point may not be on e, or even on an edge adjacent to e. We will inductively construct a sequence of points converging to p. Given a point x_n, we choose x_{n+1} to be the intersection of the circle S_n and ℓ closest to M (as reported by the oracle). Since each circle intersects ℓ exactly twice and we choose the point closest to M, the sequence approaches the endpoint p.

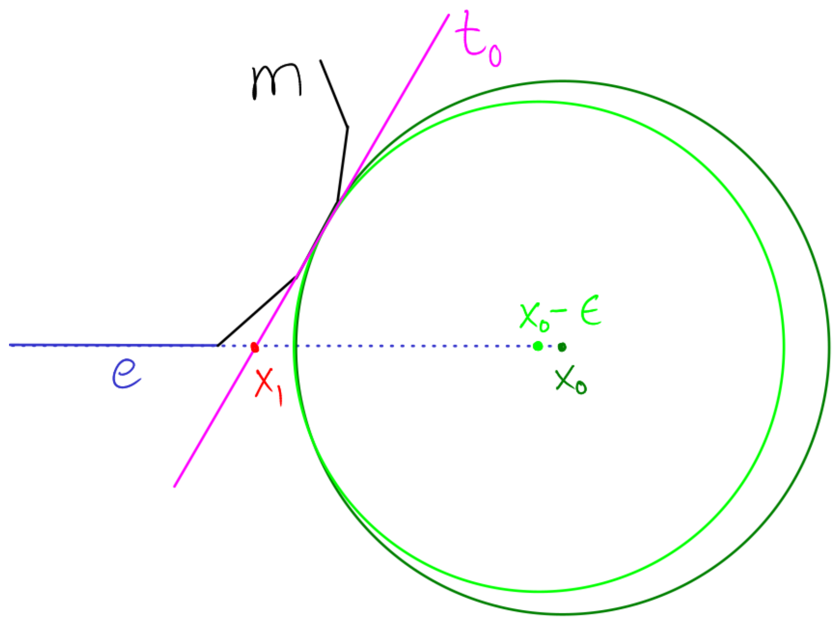


Tangent Line I. Given x_0, we first choose the point y_0 to be the intersection of S_{x_0} and ℓ. Let t_0 be any of the common tangent lines to both S_{x_0} and S_{y_0}. Since both x_0 and y_0 lie on ℓ, the tangent t_0 intersects ℓ at a point x_1 closer to M than x_0 and y_0. We continue inductively to create a sequence converging to p.

Note that if the two points x_n and y_n used to generate the tangent t_n lie in the domain of a common edge, then t_n contains that edge. In particular, if this edge is adjacent to e then the next point in the sequence x_{n+1} is exactly equal to p.



Tangent Line II. We employ the same technique as in the previous algorithm, but instead of choosing y_n to be the intersection of ℓ and S_{x_n}, we use the point $x_0 - \epsilon$ that is a distance of ϵ closer to M than x_n. The tangent line t_n is constructed in the same fashion, as is the next point x_{n+1} in the sequence.



Point Domain. Suppose the point p is attached to the edges e and e'. The domain of p is an open cone bounded by rays orthogonal to e and e' emanating from p facing away from M. Say that these rays L and L', respectively.

Consider a ray parallel to L that starts at x_0. We wish to find a point on this ray that is in the domain of p. Since M does not have any degenerate edges, this ray and L' must intersect at some point q. Points below q with respect to the ray must the lie in the open cone representing dom(p).

If we assume that ϵ is the smallest unit of distance for our algorithm, then $1/\epsilon$ is the largest unit of distance. For small enough ϵ, we may assume that the point $x_0 - \frac{1}{\epsilon}$ found a distance of $1/\epsilon$ away from x_0 in the direction of the ray L emanating from x_0 lies in the domain of p.

Let $d = \mathcal{F}(x_0 - \frac{1}{\epsilon})$. Since L is orthogonal to ℓ we may then solve for the distance a from x_0 to p to be

$$a = \sqrt{d^2 - \frac{1}{\epsilon^2}}.$$

The point p is then the point that is a distance of a away from x_0 closer to M, as measured along ℓ.

Note that if ϵ is not small enough, then the point $x_0 - \frac{1}{\epsilon}$ may not be in $\mathrm{dom}(p)$. However, we can identify the error by checking whether $\mathcal{F}(x_0 - a) = 0$. If not, then we can either choose a point along the ray further than $x_0 - \frac{1}{\epsilon}$, or take $x_1 = x_0 - a$ and repeat the process from x_1.

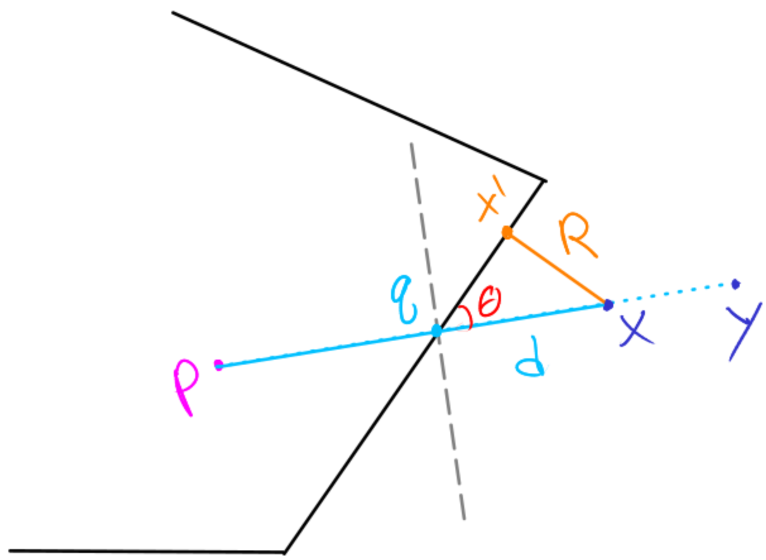
3 Identifying the Polygon

Once an edge and a vertex is found, we may easily find identify all edges and vertices of the polygon. Let p be a vertex on an edge e contained in the line ℓ. Let $p + \epsilon$ be the point on ℓ a distance of ϵ away from p that has a positive distance from M. Then the next edge e' containing p is determined by the line tangent to $S_{p+\epsilon}$ that contains p.

4 New Strategy: Normal Lines

In this section we will explore a new strategy that is more robust in higher dimensions. This new procedure is iterative, seeking to produce finer and finer approximations of the polytope at each stage.

We will first demonstrate the idea of the algorithm in the two-dimensional case. Let $p \in \mathbb{R}^2$ be a point in the interior of M and let $y \in \mathbb{R}^2$ be in the exterior of M. That is, $\mathcal{F}(p) < 0$ and $\mathcal{F}(y) > 0$. We may obtain a point $q \in \partial M$ by performing a search (using the oracle) along the line segment between p and y. For now, let us assume that q lies on an edge e.



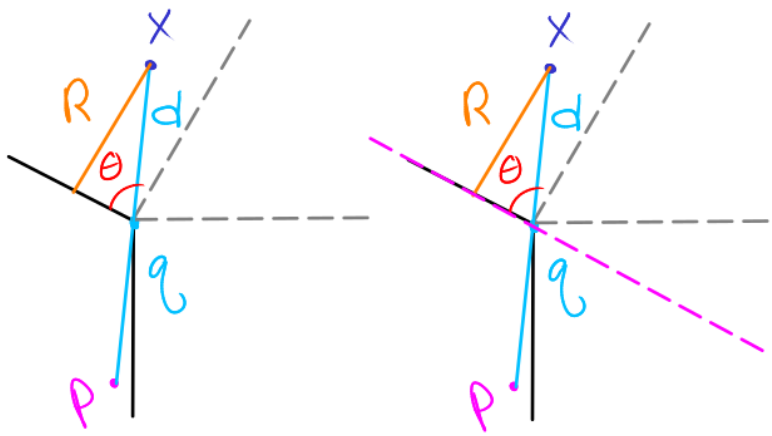
Now choose a point $x \in \mathbb{R}^2$ along the same line between q and y such that the distance between q and x is less than ϵ. We will then assume that x being within ϵ of q ensures that x is in the domain of the edge that contains q. Let d be the distance between q and x and let $R = \mathcal{F}(x)$. Suppose that the orthogonal projection of x onto e is the point $x' \in e$, so R is the distance between x and x'.

Since x is in the domain of e, the points q, x, x' determine a right triangle. Let θ be the angle at q with opposite side with length R and with hypotenuse d. We then have that $\theta = \cos^{-1}(R/d)$. However, note that there are two possible unknown edge positions, corresponding to an angle of either θ or $\pi + \theta$. The proper edge can be chosen by checking the oracle evaluation at a point on one of the possible edges; consider a coordinate system with origin at q and with one coordinate vector e_1 pointing from q to x and the other e_2 orthogonal to e_1. Then the point

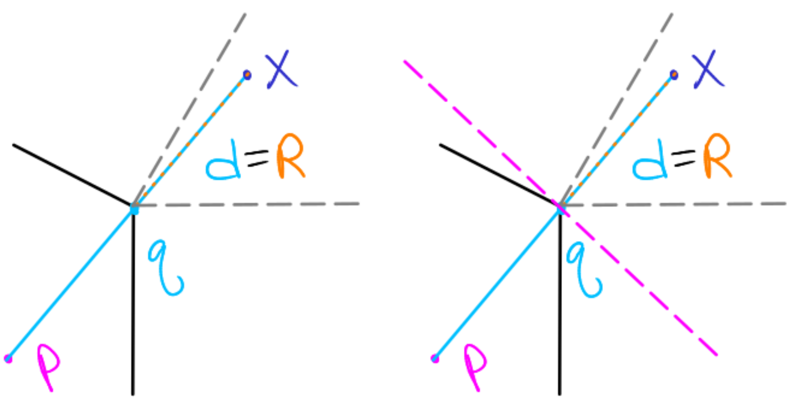
$$(\epsilon \cos\theta, \epsilon \sin\theta)$$

lies on one of the two possible edge configurations.

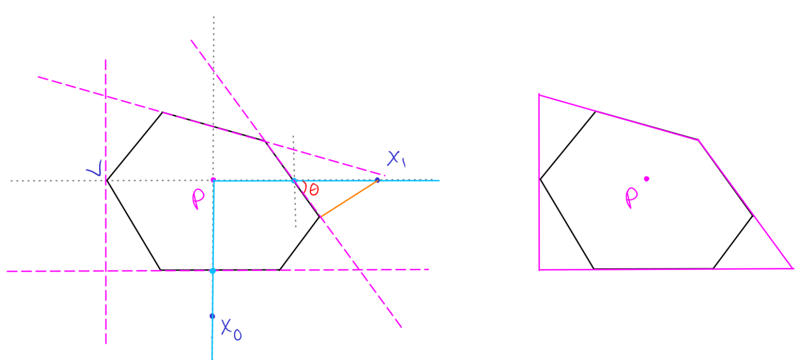
Now suppose that q is a vertex of M. We consider two cases: whether x is in the domain of the vertex q or in the domain of an edge e'. Since x is within ϵ of q, we will assume that e' is an edge that contains q. If $x \in \text{dom}(e')$ then the algorithm proceeds as normal: since e' contains q, we obtain a right triangle whose angle θ can be solved. The resulting bounding line is the line that contains e'.



So assume that $x \in \mathrm{dom}(q)$, so $R = d$. Since M is convex, the interior angle at q is less than π. We may then take the line orthogonal to the line from q to x as our approximation of M.



An iteration of the algorithm determines a bounding line ℓ for M. By convexity, all of M lies on one side of ℓ. Therefore, by running this algorithm over the four coordinate directions $\pm e_1, \pm e_2$, we obtain four bounding lines for M.



Questions:

- Are the only problem points the vertices in the approximation? What about if all coordinate direction points were vertices?

- What happens in the case that the ϵ approximation fails? That is, when there are edges of length less than ϵ in each stage of the algorithm?

4.1 Higher Dimensions

The basic idea of the above algorithm generalizes to n dimensions. We may still choose points along each coordinate direction and search for points on the polytope. We must find a more robust way of comparing distances with the oracle distance than using angles, as this does not generalize well to $\mathbb{R}^n$.

Let $M \subset \mathbb{R}^n$ be an n-dimensional convex polytope in $\mathbb{R}^n$. Once again, let $p \in \mathbb{R}^n$ be a point in the interior of M and $x \in \mathbb{R}^n$ be exterior to M. Search for a point $q \in \partial M$ along the line segment containing p and x.

Assume that q lies in the interior of a (top-dimension) facet F of M, and that all points within an ϵ-ball of x lie in the domain of F. If $N = (N_1, \ldots, N_n)$ is a normal vector to F pointing to the exterior of M, then the distance from x to F is given by

$$\mathcal{F}(x) = \frac{\langle x - q, N \rangle}{|N|} = \left\langle x - q, \frac{N}{|N|} \right\rangle.$$

Note that our choice of x on the exterior of M ensures that this distance is positive.

Since all points in an ϵ-neighborhood of x lie within the domain of F, we can choose n samples $\{x_i\}_{i=1}^n$ in this neighborhood whose distance to F is given by

$$\mathcal{F}(x_i) = \frac{\langle x_i - q, N \rangle}{|N|} = \left\langle x_i - q, \frac{N}{|N|} \right\rangle.$$

We then have a linear system of n equations in the n unknown components of the unit normal vector $\frac{N}{|N|}$ to the facet F. Under the assumption that all points lie in the domain of F, the solution exists and is unique. (A simple numerical experiment indicates that a solution for the *unit* normal vector is readily found using samples chosen from a uniform distribution on an ϵ-sphere surrounding x. See *experiment.py*.)

In the case that some of the samples x_i are contained in the domains of different faces of M, the linear system will not have a unique solution. Our original proposal to fix this issue is to simply choose the hyperplane containing q that is orthogonal to the line from p to x. Unfortunately this technique does not always work, as shown by the figure below.

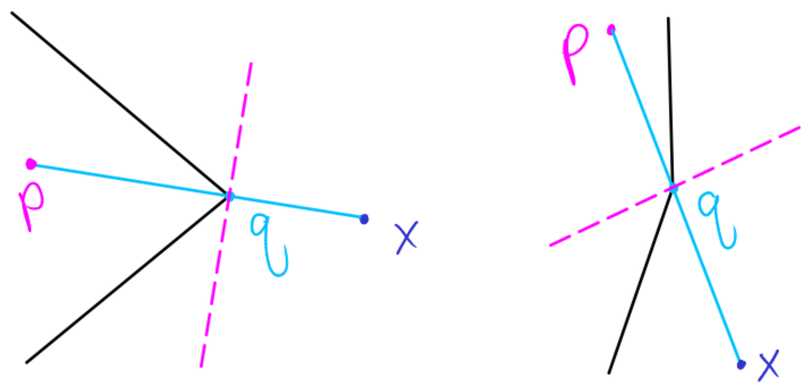


Figure 1: The dotted purple line is the chosen hyperplane at a vertex. Left: The orthogonal hyperplane is valid. Right: The orthogonal hyperplane is invalid.

We may solve this issue by changing the sampling strategy. Rather than selecting random samples from an ϵ-ball surrounding x, we pick samples along the line segment from q to x that are within ϵ of q.

Consider a ray ℓ emanating from a point q of a face F (of any dimension) of M. There exists a neighborhood U of q on ℓ such that all points of U are contained in the domain of the same face F' of M. The common face F' is either equal to F or is an adjacent face to F, as shown below.

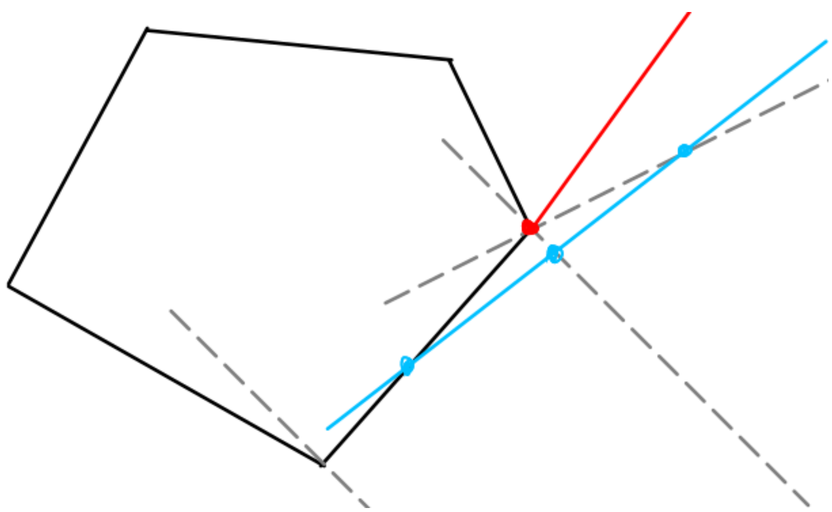


Figure 2: Light blue: a ray that has a neighborhood contained in the original face domain. Red: a ray that has a neighborhood contained in an adjacent face domain.

We may use this fact to obtain a linear system of equations with a unique solution for any face F. We still assume that we have a line segment from an interior point p to an exterior point x, passing through a point $q \in \partial M$. Let U be the neighborhood of q along the segment from q to x for which U is in the domain of some face F containing q. Sample n points $\{x_i\}_{i=1}^n$ in U, which all project to F under the oracle.

Note that F may not be a facet and so may not have a notion of normal vector, such as in the case of a vertex of a 2-polytope. However (needs proof for higher dimensions!) the distance function

$\mathcal{F}$ of U onto F is equivalent to the orthogonal projection of U onto some hyperplane containing F. The unit normal of the hyperplane is directed into the domain of F and so M lies on the negative side of the hyperplane. Therefore this hyperplane is a valid choice for generating a face of Q_i. The figures below illustrate the choice of the hyperplane in $\mathbb{R}^2$.

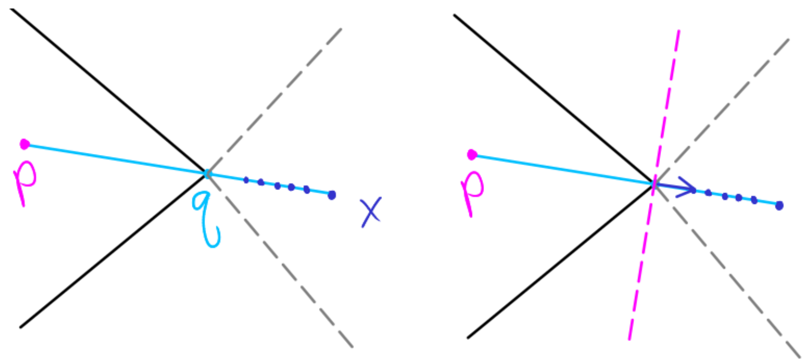


Figure 3: A common normal vector associated to a hyperplane containing a vertex.

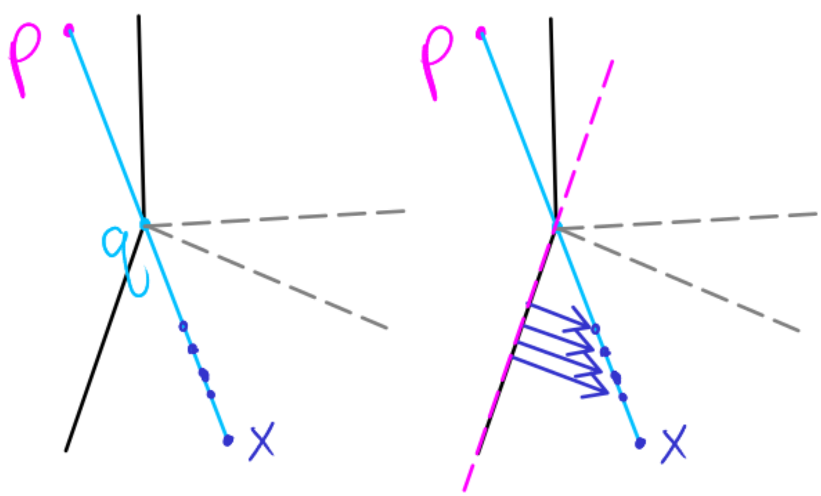
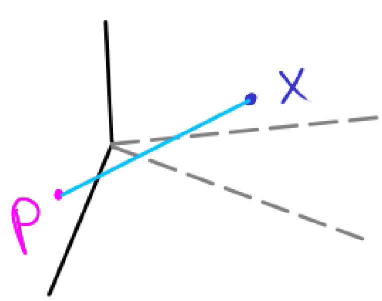


Figure 4: A common normal vector associated to a hyperplane containing a hyperplane: results are consistent with the original algorithm for a facet.

The weakness of this strategy is the unknown neighborhood U. Even with the assumption that ϵ is the minimum tolerance, the size of U depends on the positions of p and x. In the figure below, the neighborhood U may be very small in relation to the size of a face.



It is possible that this issue arises when p is “too close” to a face. One solution might be to more intelligently choose p to be further in the interior of M, but it is unclear how this can be done. Alternatively, we can reselect the point x in some fashion. However, this method fails in further iterations of the algorithm, when x is a vertex of Q_i and so cannot be reselected.

5 Multiple Polytopes: Axis-Aligned Boxes

We will now consider the case of multiple convex polytopes $M_1, \ldots, M_k$ in $\mathbb{R}^n$. We will assume that each polytope is an axis-aligned box. That is, each of the $2n$ facets of M_i is orthogonal to one of the coordinate axes. Our goal is to identify the union

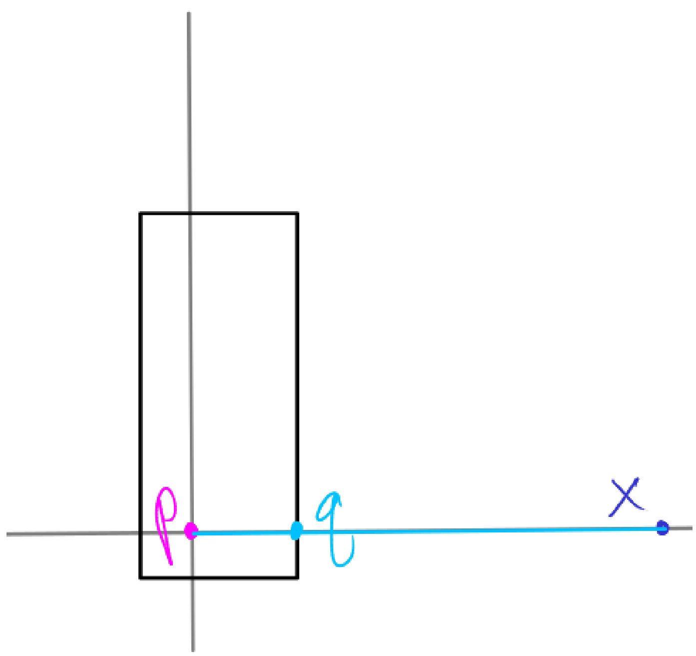
$$M = \bigcup_{i=1}^{k} M_i$$

of axis-aligned boxes. Note that the union may not be disjoint, so a connected component of M may not be an axis-aligned box.

For a single axis-aligned box M, the algorithm from the previous section can be greatly simplified. The goal of the algorithm seeks to identify the hyperplanes that correspond to each facet of a polytope, but the hyperplanes of an axis-aligned box are already known. Instead, we simply need to determine where these hyperplanes sit in space. Suppose that we have a point $p \in \text{int}(M)$ that we set to be the origin. Let $x_i \in \mathbb{R}^n$ be a point along the e_i-axis such that $\mathcal{F}(x) > 0$ and let

$$r_i = d(x_i, p) - \mathcal{F}(x_i).$$

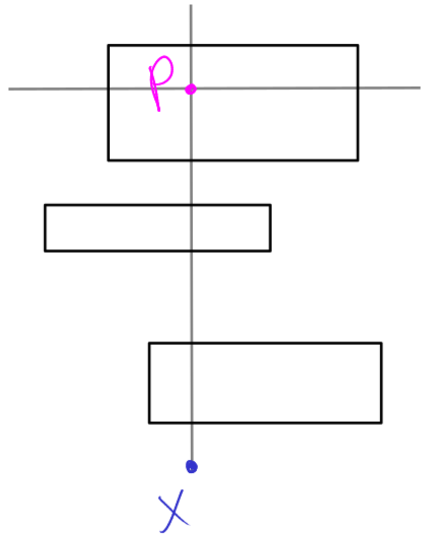
Then a facet of M is contained in the hyperplane whose unit normal vector is e_i and passes through the point q_i whose ith coordinate is r_i and all others are zero.



Repeating this process for the $2n$ signed coordinate directions produces the polytope.

5.1 Disjoint Unions

We will first consider the case of a disjoint union of axis-aligned boxes $\{M_i\}_{i=1}^k$. It is no longer guaranteed that a point $x \in \mathbb{R}^n$ on one of the coordinate axes lies within the domain of a facet of a polytope that contains the origin.



In fact, the domain structure of the exterior of M becomes very complicated. However, each individual axis-aligned box M_i has a simple interior domain structure. Our general approach to the union problem will then be opposite to our exterior-based approach to the arbitrary convex polytope case.

Consider $x \in \text{int}(M_i)$. Note that the circle S_x centered at x whose radius is $|\mathcal{F}(x)|$ cannot contain a vertex of M_i. This fact holds for circles inscribed in general convex polytopes, but here we can make greater use of this fact since we know the hyperplanes of our axis-aligned boxes.

We will first illustrate the technique for a single polytope M_i. Suppose that we have a point $p \in \text{int}(M_i)$ and consider the circle S_p. Since M_i is an axis-aligned box, S_p intersects M_i in at most $2n$ points which lie on rays emanating from p parallel to the signed coordinate axes. So consider the points

$$\{p \pm \mathcal{F}(p)e_j \mid j = 1, \ldots, n\}.$$

By definition of the oracle, one of these points lies on M_i. We may simply test these points using the oracle to find a point on M_i. If we find that

$$\mathcal{F}(p \pm \mathcal{F}(p)e_j) = 0$$

using the oracle, then we know that the $\pm e_j$-hyperplane of M_i passes through the point $p \pm \mathcal{F}(p)e_j$. In **Figure**

Once a facet has been found, the others may be found by walking along the signed coordinate directions from p and applying the same algorithm until a new facet is found. Note that while there were $2n$ directions to check from p, there are $2n - 2$ directions to check after one facet is found: if the next point is obtained from p by moving in the e_ℓ direction, then there is no reason to check the $-e_\ell$ direction, and the $\pm e_j$ direction is ignored since that hyperplane was found on the previous iteration. The figure below shows the first two iterations for a 2-polytope.

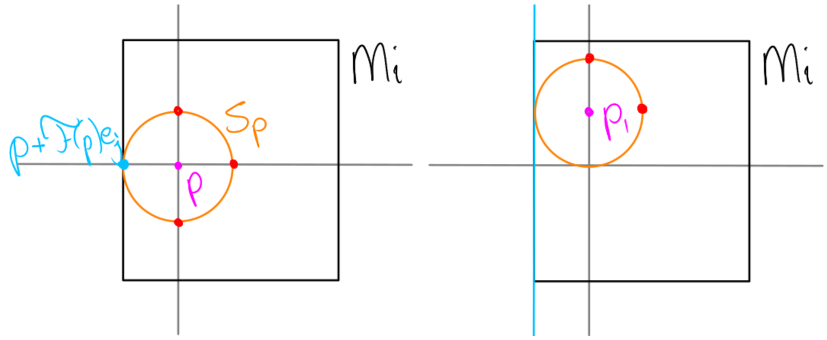
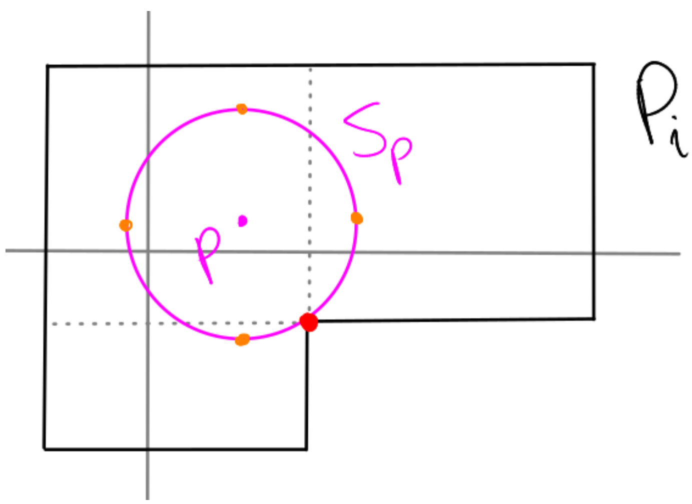


Figure 5: Left: The first iteration using an interior point p. The four "compass directions" are tested, with the leftmost point being found to be on the boundary. Right: The second iteration, choosing to use the point p_1 above p. Note that only two points on S_{p_1} are being checked.

Note that it is important that the sample points lie within a polytope. If a sample is found to be outside, it may lie in the domain of another polytope. A search function may then be used to find another point within the polytope so that the algorithm may be applied correctly.

5.2 Non-Disjoint Unions

The procedure for a non-disjoint union is precisely the same as in the disjoint union case. The only difference is that for a point p in the interior of a connected component P_i of M, p may now lie in the domain of a vertex.



When this occurs, it is possible that none of the sample points along the signed coordinate directions on S_p will lie on P_i. However, the rest of the procedure may continue as normal.

Questions:

- How do we find a point p in the interior of a connected component of M? Is it worth sampling an entire disk of $\mathbb{R}^n$ containing the origin to find these points?
- Once a single connected component is identified, what is the best way to find the others?

- What is the worst-case scenario for a union of boxes? Some sort of ”devil's staircase”-like construction such that sample points lie within the domains of vertices and fail to reach the edges?
- What is the most effective way to “walk around” in the interior of a polytope?