Technische Universiteit Delft
Faculteit Elektrotechniek, Wiskunde en Informatica
Delft Institute of Applied Mathematics

Constructing the medial axis of a discrete level-set curve

(Nederlandse titel: Construeren van de mediale as van een discrete niveau-set kromme)

Verslag ten behoeve van het
Delft Institute of Applied Mathematics
als onderdeel ter verkrijging

van de graad van

BACHELOR OF SCIENCE
in
TECHNISCHE WISKUNDE

door

E. Frima

Delft, Nederland
Juni 2021

BSc verslag TECHNISCHE WISKUNDE

" Constructing the medial axis of a discrete level-set curve (Nederlandse titel: Construeren van de mediale as van een discrete niveau-set kromme)"

E. Frima

Technische Universiteit Delft

Begeleider

Dr. ir. D. den Ouden-van der Horst

Overige commissieleden

Dr. ir. L.J.J. van Iersel

Juni, 2021 Delft

Contents

1 Introduction

In (mathemetical) physics, you could encounter a problem which contains a phase change such as a melting block of ice or the growth of a precipitate crystal. These so called Stefan problems are often solved using the level-set method which focusses on the distance to the border between the two phases. Sometimes numerical problems are encountered in the points that have multiple closest points on that border. These points are the medial axis of the zero level-set curve. Therefore in this paper I will try to approximate this medial axis using Fortune's algorithm.

In Section 2 we will look at the exact definitions and some properties of the Stefan problem and the medial axis. In Section 3 we will look at different approaches to calculating the medial axis and some algorithms that could be used. In detail we will look at Fortune's algorithm which is used to find a Voronoi diagram, which is closely related to the medial axis. In Section 4 we will introduce some test functions and use our method to approximate the medial axis. In the last section we will discuss some ways to improve the results of Section 4 and point to areas of future research.

2 Mathematical background

2.1 Stefan problem

One of the most common problems in the theory of partial differential equations is the heat equation. There exist a lot of methods to solve this equation ranging from analytical solutions for the simpler versions to numerical methods for more complicated versions. An extension to the heat equation can come in the form of a Stefan problem. A Stefan problem considers a phase change in the model, which in case of the heat equation this could be a block of ice melting in a basin of water. The phase change doesn't necessarily need to be one relying on thermodynamics, but could also be a more chemical process such as the precipitate growth in binary alloys [1]. In a Stefan problem the whole domain $\Omega(t)$ is seperated into to main parts $\Omega_D(t)$ and $\Omega_P(t)$, which are seperated by a border $\Gamma(t)$. The main difficulty lies in the fact that the phase boundary $\Gamma(t)$ itself moves in time, indicated by the so-called Stefan condition, which relates the change of the boundary to the condition on both sides. To solve the Stefan problems specific numercial solution methods are needed.

2.2 Level-set method

The level-set method can be used to solve a Stefan problem, because this method captures the motion of boundary using a signed-distance function. The signed distance function $\phi(\mathbf{x}, t)$ will typically be defined by

$$\phi(x,t) = \begin{cases} \min_{\mathbf{y}\in\Gamma(t)} ||\mathbf{y}-\mathbf{x}||, & \text{if } x \in \Omega_P(t), \\ 0, & \text{if } x \in \Gamma(t), \\ -\min_{\mathbf{y}\in\Gamma(t)} ||\mathbf{y}-\mathbf{x}||, & \text{if } x \in \Omega_D(t). \end{cases}$$

Rewriting the Stefan problem in a form with the signed-distance function can help solve the problem, tough some points can give problems in numerical methods. These are the points that have multiple minimal distances to the boundary. These are points of the so called medial axis.

2.3 Medial axis

The medial axis, sometimes referred to as the skeleton, of a shape or object is all the points which have two closest points on the boundary Γ, so formally:

Definition 1 (medial axis)**.** *The medial axis is the collection of points c for which there exists an r such that the ball $B(c, r)$ touches points $p, q \in \Gamma$ and contains no other points of Γ.*

The medial axis preserves the topology of the object and it does so in an easier way, since the dimension of the medial axis is always at least one dimension smaller than the original object. Another property of the medial axis is that it's sensitive to small perturbations in the boundary, where just a small dent can add a new branch to the medial axis. As shown in figure 1, only a small dent can add a whole new branch to the medial axis.

The medial axis for convex figures is always a tree within the figure, but in the case of concave figure there will be parts of the medial axis on the outside. For instance a star shape will have medial axis lines on the outside between two peaks.

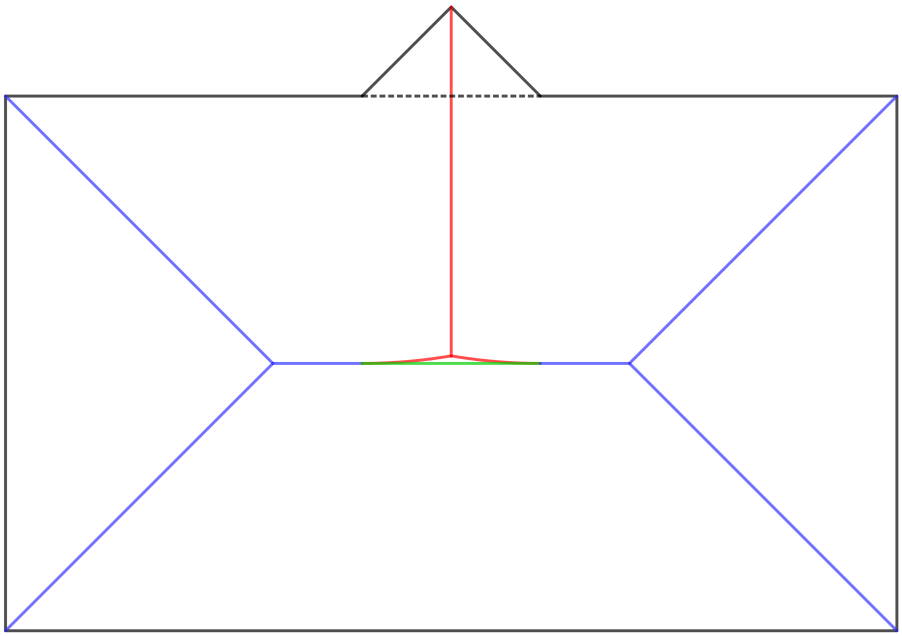


Figure 1: Medial axis of a regular rectangle (blue and green) and the medial axis after a small pertubation of the boundary (blue and red).

3 Computing the medial axis

3.1 Different approaches

For computing the medial axis there are two main approaches, a direct approach or an indirect approach via the Voronoi diagram. One of the direct approaches is the shrinking ball algorithm like the one used in [2]. In short, this algorithm uses a sample of boundary points V and their inner normal vectors N, then for each point it draws a ball touching that point with the center in the direction of the normal vector. The ball is then shrunk until it touches another sample point and there are no points inside the ball. The resulting ball is then a medial ball and its center is on the medial axis. The main weakness of this algorithm is that it's sensitive to irregularities on the boundary, which in case of the level-set method is not uncommon since you often approximate the level curve. The algorithm performs in $O(V^2)$ time in theory, but in practice its performance is better. Also the algorithm can easily be distributed over multiple processor cores, each core calculates the medial balls for a part of V.

The other approach for calculating the medial axis, is by first constructing a Voronoi diagram and deriving the medial axis from it. The Voronoi cell in a space Ω with distance function d is defined as:

Definition 2 (Voronoi cell)**.** *Let $P = (p_1, ..., p_n)$ be a collection of points in Ω. Then the Voronoi cell R_k corresponding to point p_k is*

$$R_k = x \in X | d(x, p_k) \leq d(x, p_j) \text{ for all } j \neq k.$$

The collection of all Voronoi cells of P together with vertices and edges separation the cells, form the Voronoi diagram. If the points P of the Voronoi diagram are chosen on the boundary of an object, the edges of the Voronoi diagram will contain an approximation for the medial axis. An example of this is shown in figure 2, where a set of points is chosen such that the Voronoi diagram includes the whole

medial axis. To calculate the Voronoi diagram multiple algorithms are available such as the Bowyer-Watson algorithm [3] and Fortune's algorithm [4], which will be discussed in the next section.

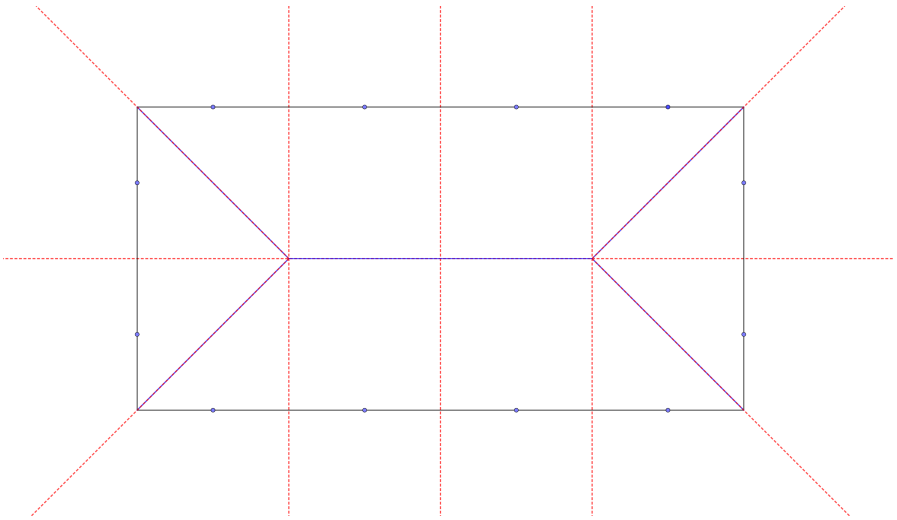


Figure 2: Voronoi diagram of some points on a rectangle with the red dotted lines seperating the Voronoi areas. The medial axis is drawn here in blue and overlaps with the Voronoi diagram.

3.2 Fortune's algorithm

For computation of the medial axis I have chosen to first calculate the Voronoi diagram and prune it later. Fortune's algorithm is mainly chosen because it has $O(n\log n)$ runtime. In short the algorithm can be visualized using a horizontal sweep line, which descends over a given set of points. When the sweep line passes these points the algorithm forms a so called beach line consisting of parts of parabolas with the points as their focuses and the sweep line as the directrix. As the sweepline progresses, the intersections of these parabolas will trace the Voronoi edges.
In more detail, the algorithm uses a priority queue to store the two types of events, the site events and the circle events. The site events are at the starting set of points, in order of their y-coordinate. For the first site event we simply add the point to a binary-search tree which stores the structure of the beach line, so we store one point as a focus of a parabola. For the next site events we look at the intersection points with the existing beach line, as is shown in figure 3. The new parabola will split the old (partial) parabola and create three arcs, a left and a right arc from the intersection points onwards and a new middle arc inbetween the intersection points. Additionally we search for triples of consecutive arcs on the beach line which use the new arc. For each of these triples we look if the intersection points on this triple converge. If they do we evaluate the circle going through the original focuses of the three arcs and add the bottommost point of this circle to the event queue as a potential circle event. We also look in the event queue for any circle event from a triple of the now split arc and remove it since the triple that created the event is now altered.

Eventually the beach line encounters a circle event that did not get removed. This indicates that we found a point where three parabolas intersect with eachother and will give us one of the Voronoi vertices, since the point is equidistant to three sites. For the beach line the middle arc will disappear and the two intersection points will become one. An example of this process is shown in figure 4. Since the beach line changed, any new triples will have to be checked and the potential circle events have to be added. Similarly any circle events still in the queue which use the disappearing middle arc need to be removed.

Apart from executing all the events as described there are some minor details to complete the algorithm. For instance the halflines on the outside of the Voronoi will not be correctly be traced since they only have one endpoint. To counteract this one usually uses a bounding box around all the points and detect when an intersection point hits this bounding box. Additionally there are some subtleties such as that occasionally two events can have the same y-coordinate, in which case one can choose freely which one to handle first.

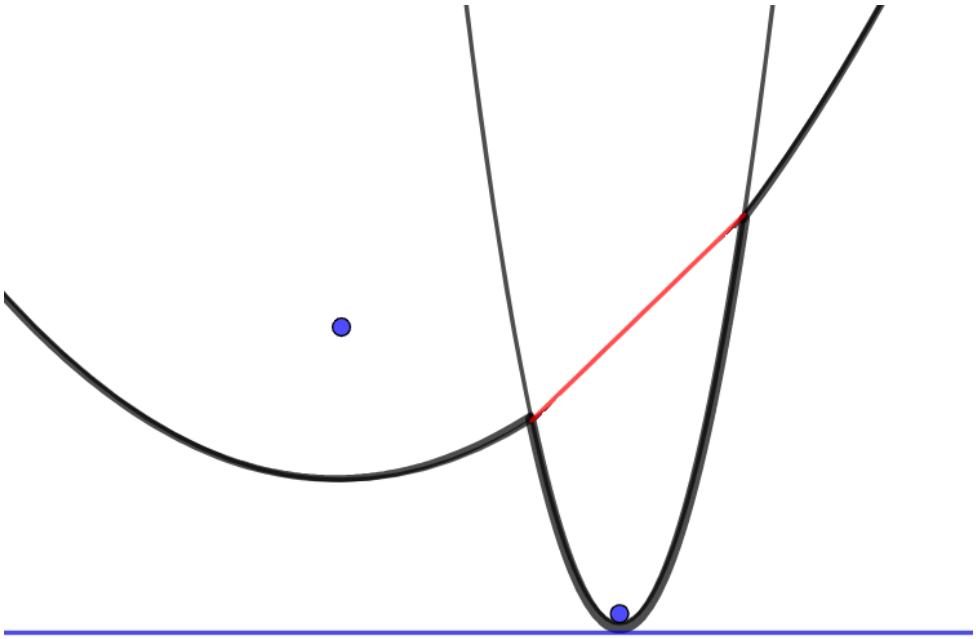


Figure 3: A site event. The sweepline in blue and the thick black line is the beach line. In between the intersetion points a Voronoi edge is starting to form in red.

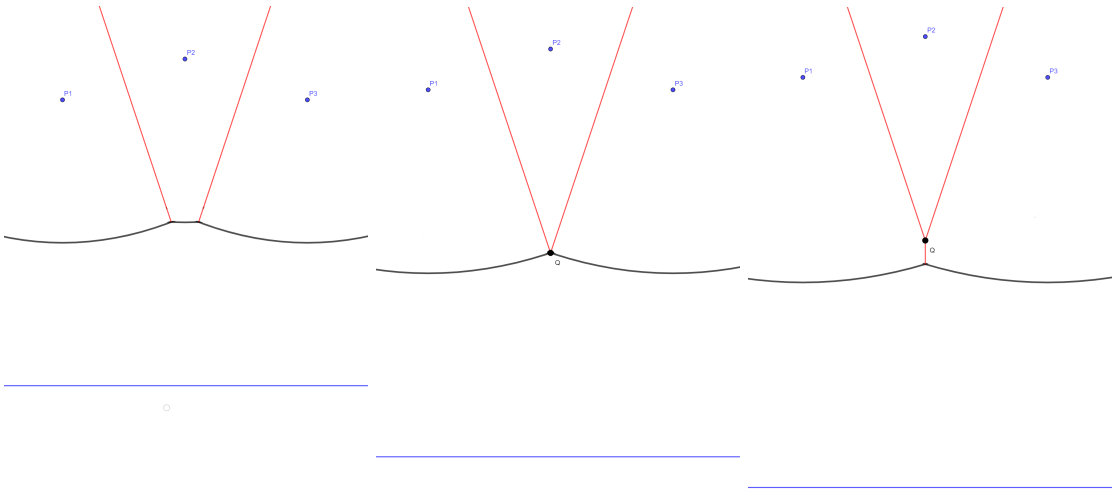


Figure 4: A circle event. The arcs of the beach line (thick black line) come together in point q, the center of the circle through all three sites. The three picture show the situation before, during and after the circle event.

If the explanation above isn't completely clear, below is the Algorithm in pseudocode and in the appendix there is a step-by-step example of Fortune's Algorithm on a set of points.

Data: set of point
Result: Voronoi diagram
initialization; Create an event queue Q and with all points as site event, create an empty tree T for the beachline and a graph G.
while Q *is not empty* **do**
  Remove the event with the highest y-coordinate from Q.
  if *the event is a site event* p_i **then** HandleSiteEvent(p_i);
  else HandleCircleEvent((c_i);
end

HandleSiteEvent(p_i)
if T *is empty* **then** add p_i to T, return;
Find the arc of p_j directly above p_i in T. Replace p_j in T with two new arcs p_j, middle arc p_i, breakpoints (p_j, p_i) and (p_i, p_j)
Remove any circle events in Q corresponding to the old arc of p_j. Look for any triples on the beachline involing p_i, look if their breakpoints converge, if so add that triple as a circle event to Q. Add an edge to G between the two breakpoints (p_j, p_i) and (p_i, p_j). Return
HandleCircleEvent(c_i)
The circle event is defined by a triple of consecutive arcs p_a, p_b, p_c and breakpoints (p_a, p_b), (p_b, p_c). Remove these in T and replace them with arcs of p_a and p_c and breakpoint (p_a, p_c).
Remove any circle events from Q that use p_b. Look for new triples with converging breakpoints containing p_a and p_c and add their circle events to Q.
Add the center of the circle to G as a vertex. Return

3.3 Triangular mesh

Before we can employ the Fortune algorithm on the test cases, we need to get a finite set of points that are relevant for the level curve. While we could use a parametrization of the test functions, like using polar coordinates for the donut test function. However, in general such nice parametrizations might not exists. Therefore we will instead use a standard triangular mesh. This mesh is constructed such that each side of the triangles is relatively close to a given value h. We assume the value of the level-set function is known at every vertex of the mesh. Naturally none of these vertices will give exactly 0 when plugged into the level-set function. Thus to approximate a set of points we will interpolate linearly on all the edges which have endpoints with opposite sign. For instance if the level-set function gives values $\frac{-1}{10}$ and $\frac{1}{5}$ at the endpoints, the approximation will be at $\frac{1}{3}$ along the length of the edge starting from the negative valued vertice.

3.4 Thinning the Voronoi diagram to the medial axis

After using Fortune's algorithm on a set of boundary points we obtain the Voronoi diagram of this set. To obtain an approximation of the medial axis some Voronoi edges need to be removed. The edges to be removed are the ones which cross the boundary of the object, because while all the points on these edges are equidistant to two boundary points, the circle touching these boundary points is typically not contained by the figure. To filter out these unwanted edges we could look at which boundary points are next to each other on the boundary and remove edges that seperate these boundary points. In the case that there exists a level-set function however finding these edges is much easier. In this case the endpoints of an edge going through the boundary will give values with opposing signs when plugged into the level-set function.

4 Numerical analysis

4.1 Introduction

In this section we will introduce three functions, which will act as our level-set functions. For these functions we will use the zero contour as our level-set curve and approximate the medial axis of this zero contour. We will use a dumbbell and a star shaped test function, similar as in [5] and a donut (or an annulus) test function. As our space we will use the square $\Omega = [-1, 1] \times [-1, 1]$ For every function we will give a function definition and a contour plot.

4.2 Star test case

The star function can be defined as:

$$f(x, y) = \frac{1}{2} + \frac{3}{10} \cos\left(6 \arctan\left(\frac{y}{x}\right)\right)^2 - (x^2 + y^2).$$

A plot of the contours of this function is shown in figure 5.
While this function is not a level-set function, since the absolute values are not equal to the shortest distance to the zero contour. However it mimics the behavior of a level-set function well. The value of the function is increasing or decreasing the further away you go from the zero contour and the signs are also correct. The expected medial axis has a inner medial axis of straight lines from the origin into the 'points' of the star and an outer medial axis of straight lines in between the 'points' of the star.

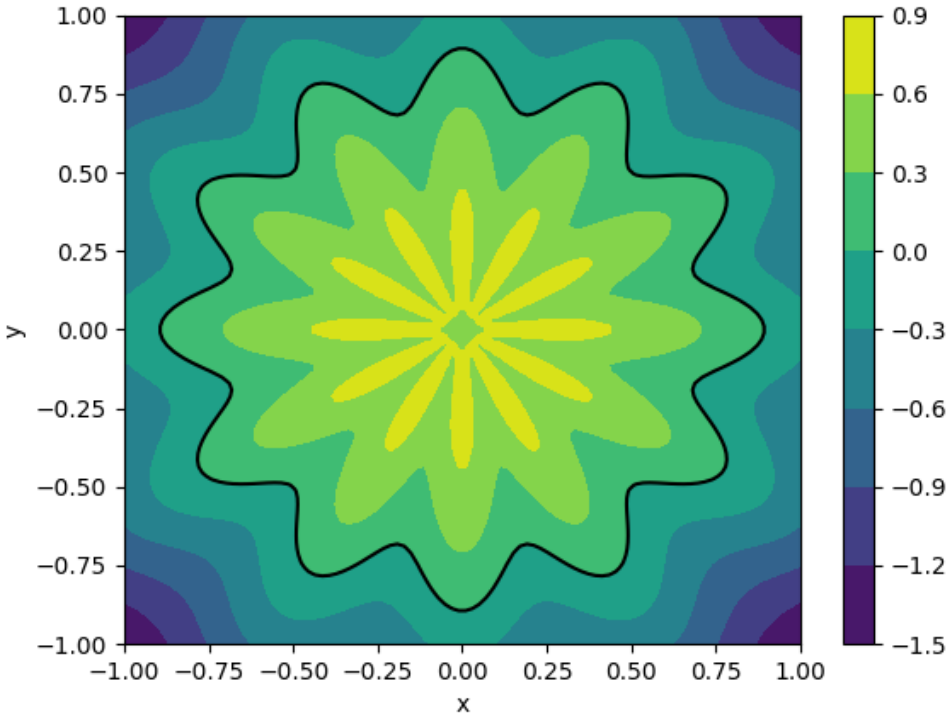


Figure 5: A contour plot of the star test function.

4.3 Dumbbell test case

The dumbbell function can be defined as:

$$f(x,y) = \begin{cases} \max\left(\frac{3}{10} - \sqrt{(x-\frac{1}{2})^2+y^2}, \sqrt{(x+\frac{1}{2})^2+y^2}, \frac{1}{10}|y|\right), & \frac{-1}{2} < x < \frac{1}{2}, \\ \max\left(\frac{3}{10} - \sqrt{(x-\frac{1}{2})^2+y^2}, \sqrt{(x+\frac{1}{2})^2+y^2}\right), & \text{otherwise.} \end{cases}$$

A plot of the contours of this function is shown in figure 6.
For the dumbbell we expect the inner medial axis to be a straight line segment between the centers of the two circles. The outer medial axis lines should originate from the corners where the bar connects to the circles.

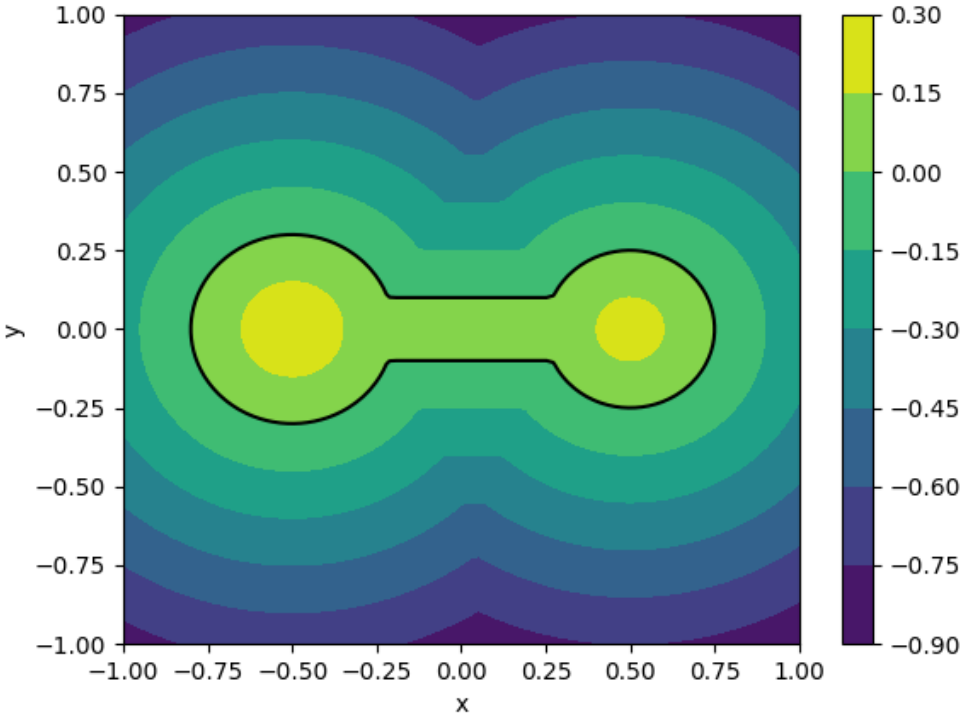


Figure 6: A contour plot of the dumbbell test function.

4.4 Donut test case

The donut function can be defined as:

$$f(x,y) = \min\left(-\sqrt{x^2+y^2} + \frac{3}{5}, \sqrt{x^2+y^2} - \frac{2}{5}\right).$$

A plot of the contours of this function is shown in figure 7.
The medial axis of the donut has an inner medial axis which is a circle which radius is the average of the radii of the two boundary circles, in this case a radius of $\frac{1}{2}$. The outer medial axis is simply the center of the circles.

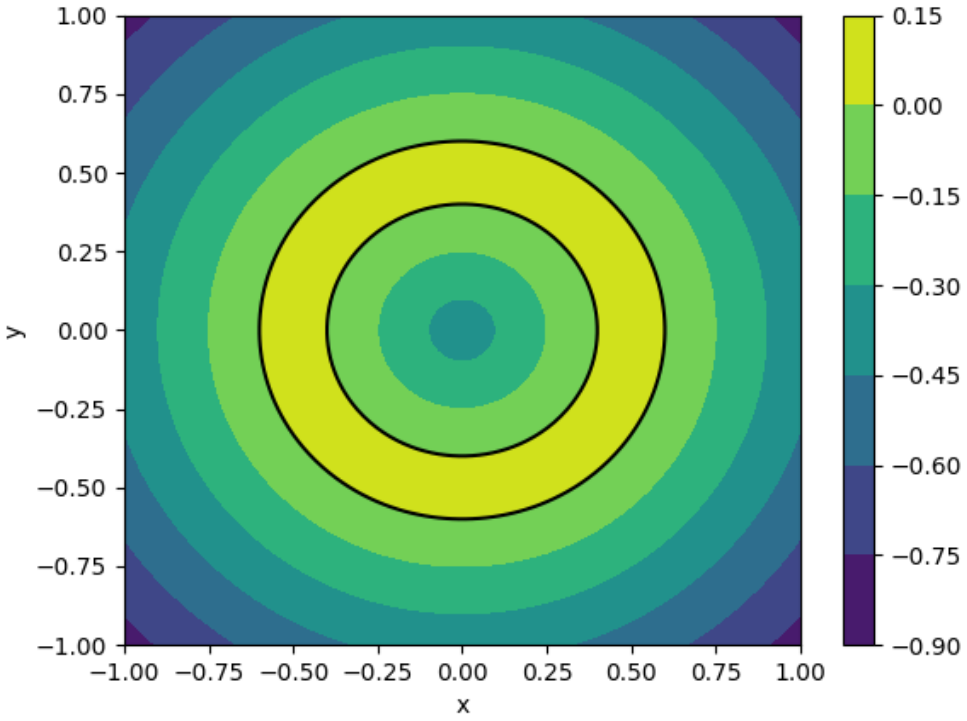


Figure 7: A contour plot of the donut test function.

4.5 Medial axis approximation

In this section we will show the results of the discussed methods used on the three test functions. First we will use a mesh to generate a set of points, then we will use Fortune's Algorithm and lastly we will filter anything we do not want and leave only the approxiamtion of the medial axis. We will discuss in short how well the method performs and what is the cause of any imperfections.

First we use the mesh with a triangle side length of $\frac{1}{8}$ and interpolate to recieve a set of points. From figure 8, we can see that the interpolation doesn't perform well in all places, escpecially in the areas where the star curves a lot. We can also see that the points are not evenly distributed, since the mesh sometimes doesn't line up with the star. Sometimes the mesh vertices are close to the star function and thus there are multiple interpolated points nearby and sometimes the opposite happens.

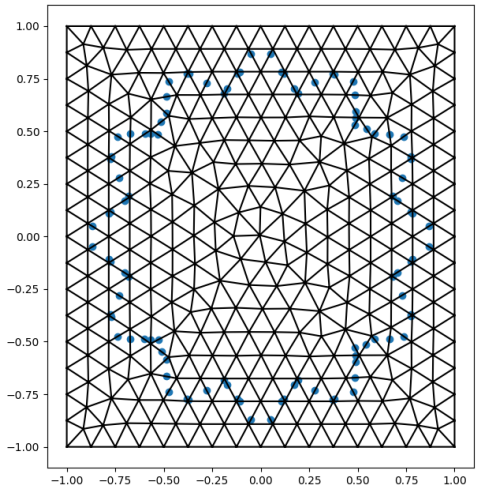


Figure 8: The mesh with the approximated level-set points in blue for the star test function. Mesh size is $h = \frac{1}{8}$.

When we use Fortune's Algorithm and thin the Voronoi diagram to a medial axis, shown in figure 9, we see that the general shape we expected is present. The medial axis lines are mostly straight lines and the inner medial axis lines come together in the origin, this is because the mesh is almost symmetrical in the x- and y-axis. However mostly at the ends of the medial axis lines there are some irregularities caused by irregularities in the boundary points. To try and achieve a better result we can try and use mesh with smaller edge size, now $h = \frac{1}{16}$ instead of $h = \frac{1}{8}$. After doing that we see that the medial axis looks slightly better, as can be seen in figure 10, especially at the ends of the medial axis lines. However there are still some anomalies caused by boundary points which are relatively close to each other and have a small approximation error within them.

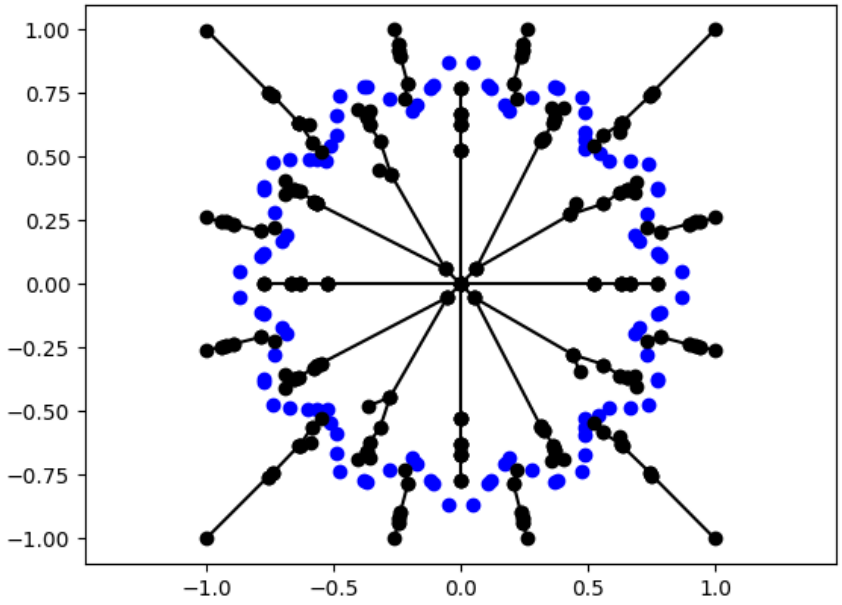


Figure 9: The medial axis approximation of the star test function. Original mesh size used was $h = \frac{1}{8}$.

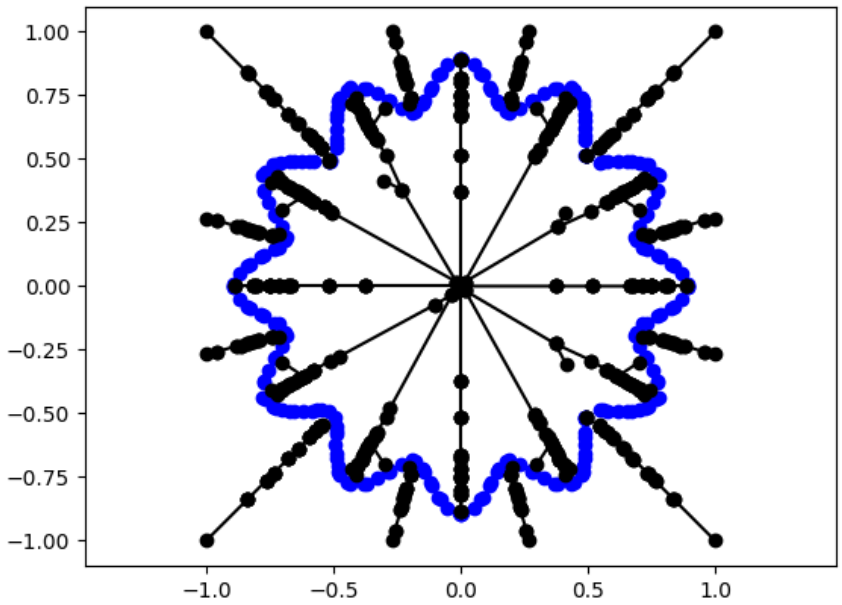


Figure 10: The medial axis approximation of the star test function. Original mesh size used was $h = \frac{1}{16}$.

In the case of the dumbbell test function using the precision of $h = \frac{1}{16}$ we see a better result in figure 11. The middle looks particularly good, because the dumbbell is linear in the middle so the linear

interpolation on the mesh gives good results. In the two circles however the methods don't perform well, the medial axis points are scattered around the center of the circles.

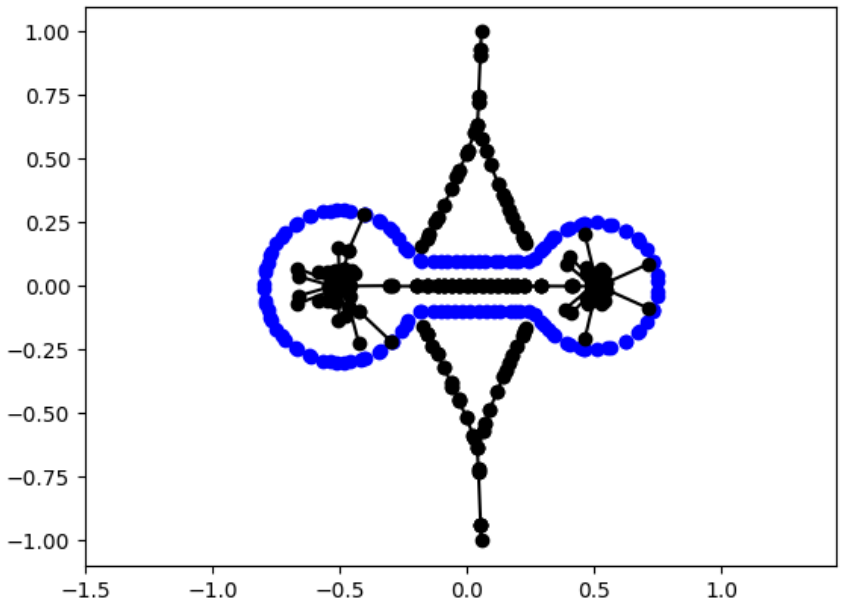


Figure 11: The medial axis approximation of the donut test function. Original mesh size used was $h = \frac{1}{16}$.

The donut test case is interesting since it has two boundaries, this benefits the inner medial axis which is a circle in between the boundaries. However, in figure 12 we also see that the outer medial axis is a bit messy. The outer medial axis however looks a bit messy. With the mesh edge length of $h = \frac{1}{8}$ we encounter three points outside the donut, but they luckily disappear when the edge mesh gets denser, as shown in figure 13. However the medial axis inside the inner circle doesn't improve all that much and there are still anomalies present no matter how tight we make the mesh.

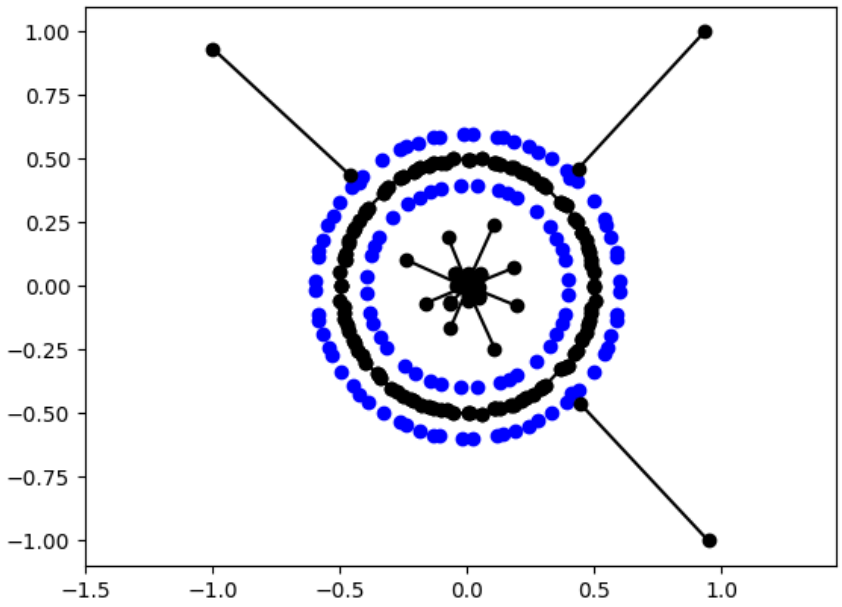


Figure 12: The medial axis approximation of the dumbbell test function. Original mesh size used was $h = \frac{1}{16}$.

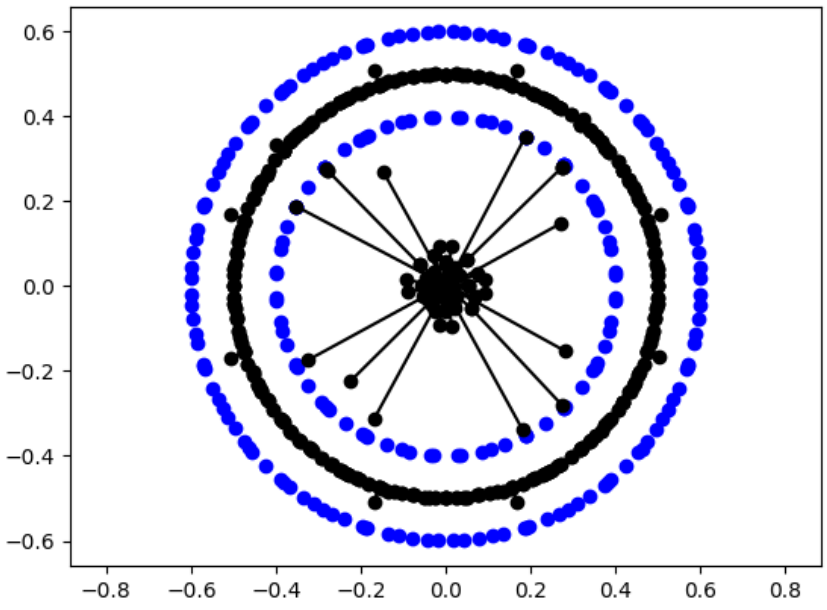


Figure 13: The medial axis approximation of the donut test function. Original mesh size used was $h = \frac{1}{16}$.

5 Discussion

The results from the previous section are far from perfect and in this section we will discuss some ways to improve these results. We will also discuss the limitations of the methods and how they might be altered for future research.

5.1 Improving by meshfitting

Most of the problems in the approximation of the medial axis were caused by irregularities in the set of boundary points. This is due to the mesh not lining up with the level-set function. As an example, shown in figure 14, if we distribute the points on the outer and inner circle of the donut test function by giving neighboring points a consistent angle in polar coordinates, the method will give a near perfect result.

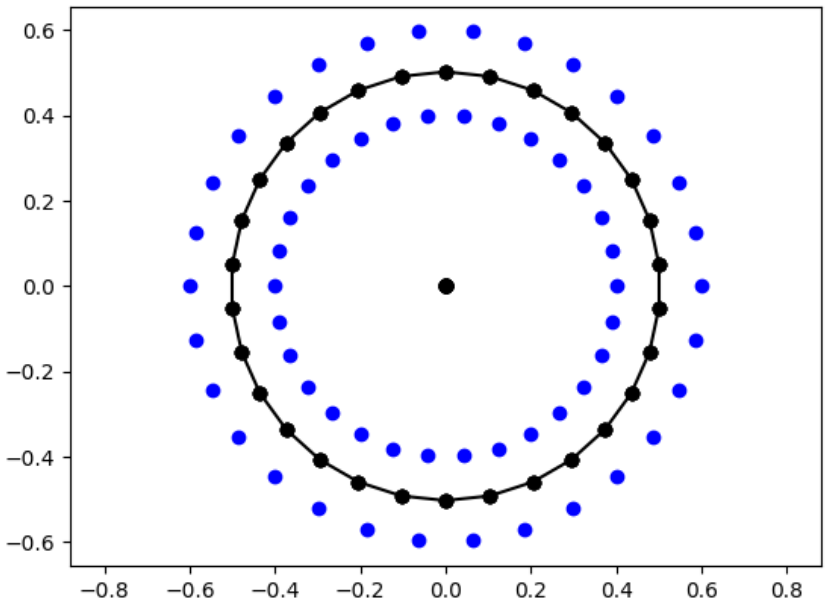


Figure 14: The medial axis approximation of the donut test function with the point collection of the donut nicely distributed.

To improve the distribution of the boundary points we can use a so called mesh fitting algorithm like those decribed in [5] and [6]. These algorithms change the mesh points such that there is a set of mesh points on the level-set while still aiming to have the edge length in the mesh consistent. After using the right meshfitting algorithm you don't even need to interpolate, since you can use the mesh points that are now placed on the level-set curve.

5.2 Improving after the algorithm

Another way one could try to improve the results is by trying to smooth out the medial axis. In the star test case for instance you could try to straigthen the lines of the medial axis. For the circular cases where the medial axis is supposed to be a single point one could consider using statistical methods to filter any outliers of the point cloud we recieved. The main problem however is that in general you don't know what the medial axis should look like, so one would have to find a way to distinguish between different cases.

5.3 Three dimensional medial axis

While all these methods have been in the plane, the medial axis also exists in three dimensinal objects. The step of interpolating on the mesh, should be simple to extend to three dimensions, as well as converting a 3D Voronoi diagram to a medial axis. The difficulty comes with using Fortune's Algorithm, since it isn't easily extended to three dimensions. You could try changing the parabolas to paraboloids and use a sweeping plane, but this will certainly not be straightforward and likely cause the algorithm to have a lot of edge cases. It may be easier to use another algorithm instead such as the Bowyer-Watson algorithm [3].

References

[1] den Ouden D, Vermolen FJ, Zhao L, Vuik C, Sietsma J (2011); Modelling of precipitate nucleation and growth in binary alloys under elastic deformation: an application to a Cu-0.95%Co alloy. Comput Mater Sci 50:23972410

[2] MA J., BAE S. W., CHOI S.; 3D medial axis point approximation using nearest neighbors and the normal field. The Visual Computer 28, 1 (2012), 719

[3] A. Bowyer, Computing Dirichlet tessellations, The Computer Journal, Volume 24, Issue 2, 1981, Pages 162166, https://doi.org/10.1093/comjnl/24.2.162

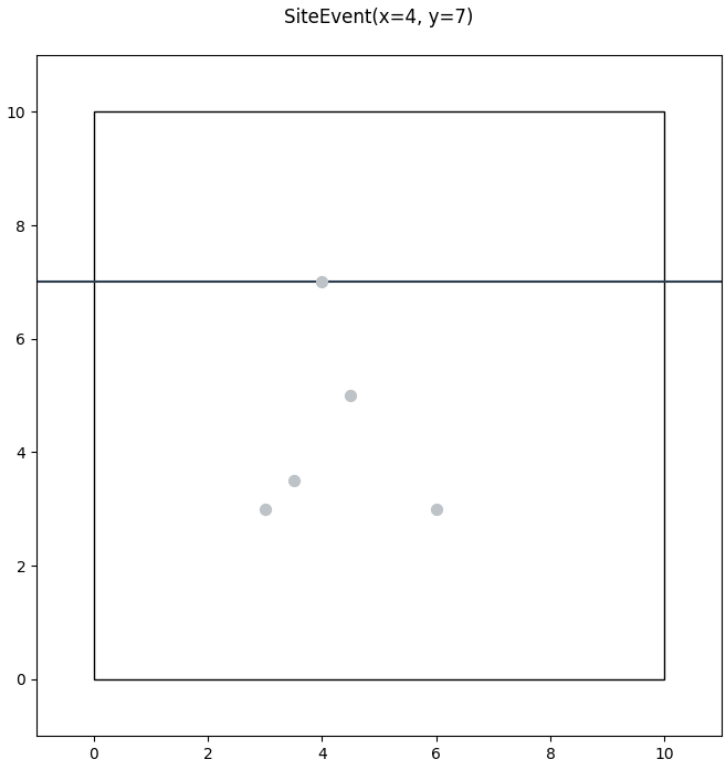
[4] Fortune, S. A sweepline algorithm for Voronoi diagrams. Algorithmica 2, 153 (1987). https://doi.org/10.1007/BF01840357

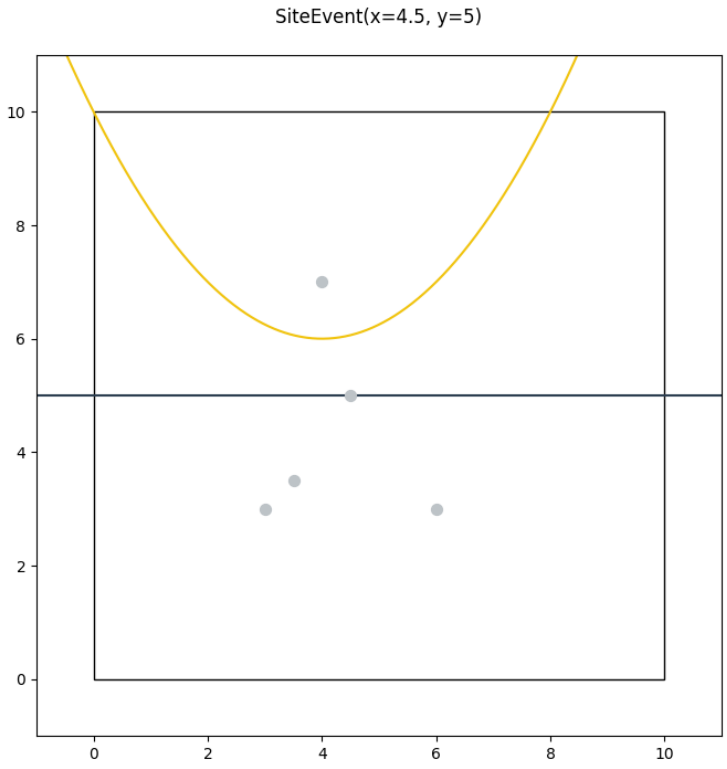
[5] W. M. den Hertog. An elastic model for meshfitting for the Level-set Method. Delft: Technische Universiteit Delft, 2019.

[6] T. E. Wortelboer. Meshfitting for the Level-set Method. Delft: Technische Universiteit Delft, 2018.

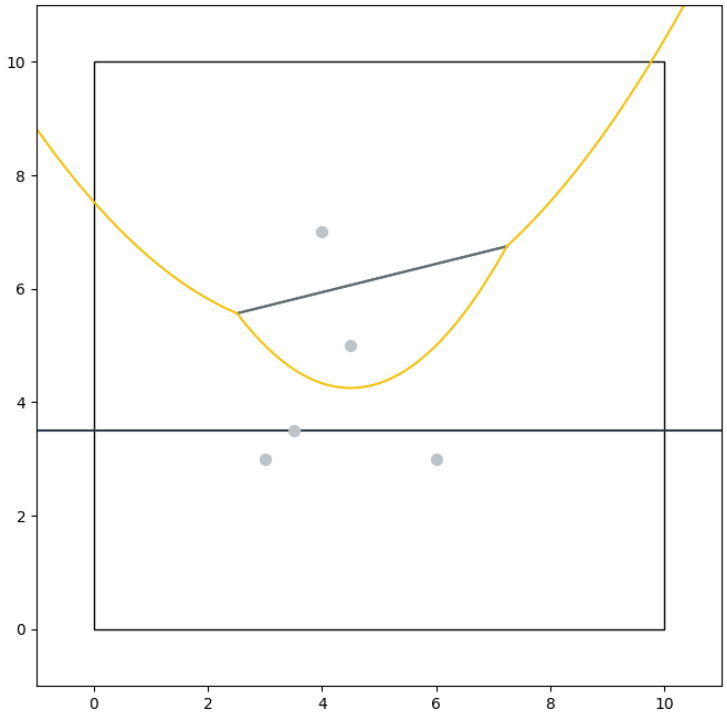
A Full example of Fortune's Algorithm

Below is a step-by-step example of Fortune's Algorithm. The set of points is $S = \{(3,3),(3.5,3.5),(4,7),(4.5,5),(6,3)\}$ and the bounding box is a square with corners $(0,0),(10,0),(10,10),(0,10)$. All of the site events and circle events are done in order. After each site event we see an extra arc was added to the beachline (yellow). For the circle events, the circle is drawn in blue and the center is added to the Voronoi diagram as a vertice. At the last step the Voronoi diagram is fit in the bounding box.

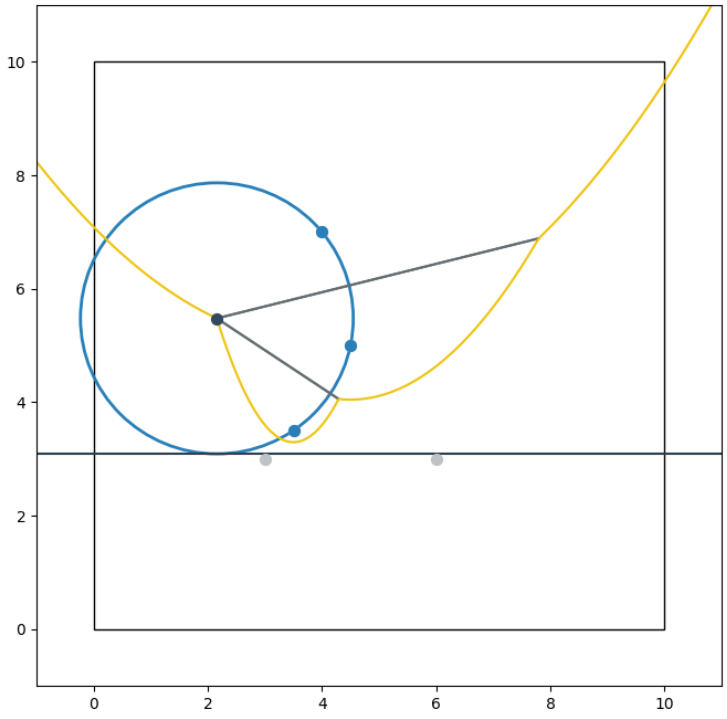

SiteEvent(x=4, y=7)
10
8
6
4
2
0
0
2
4
6
8
10

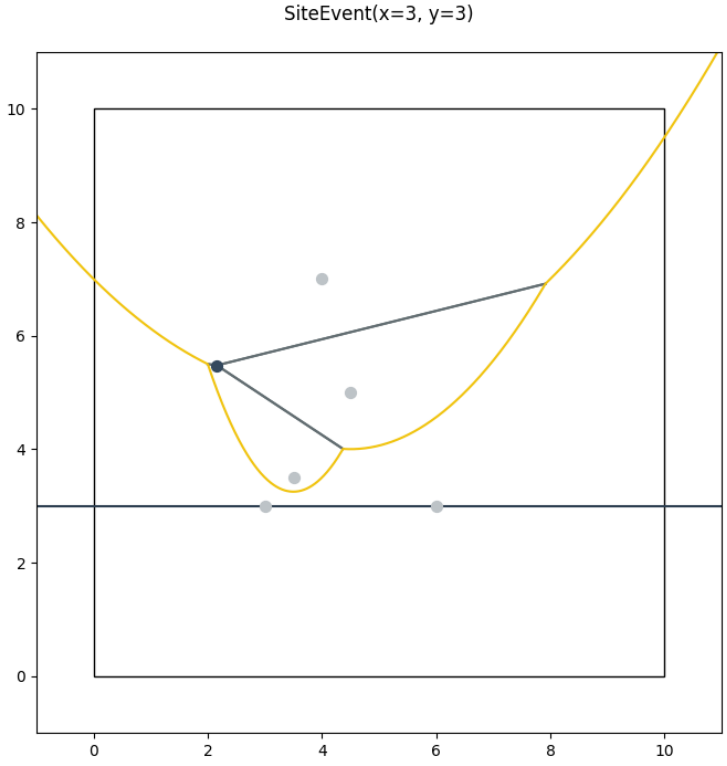

SiteEvent(x=4.5, y=5)
10
8
6
4
2
0
0
2
4
6
8
10

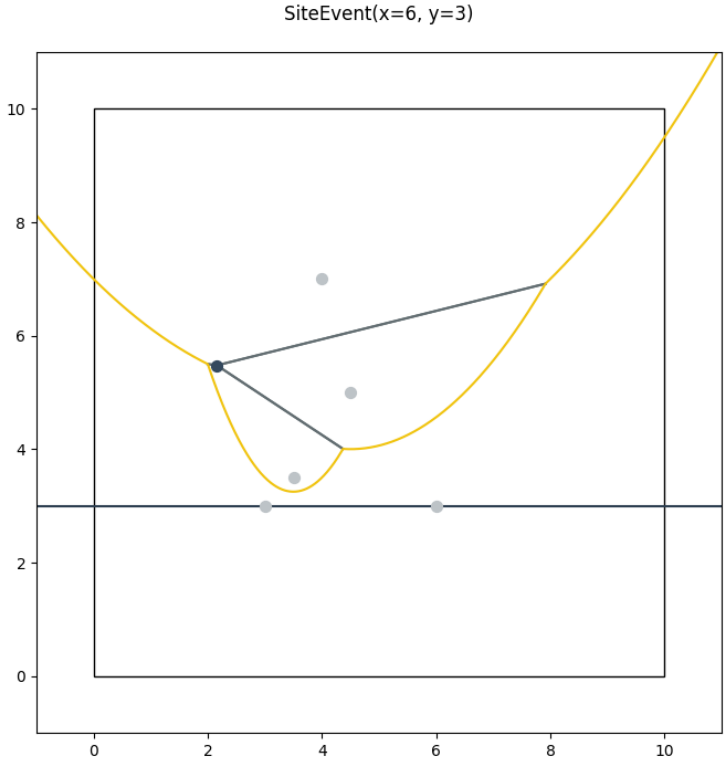
SiteEvent(x=3.5, y=3.5)



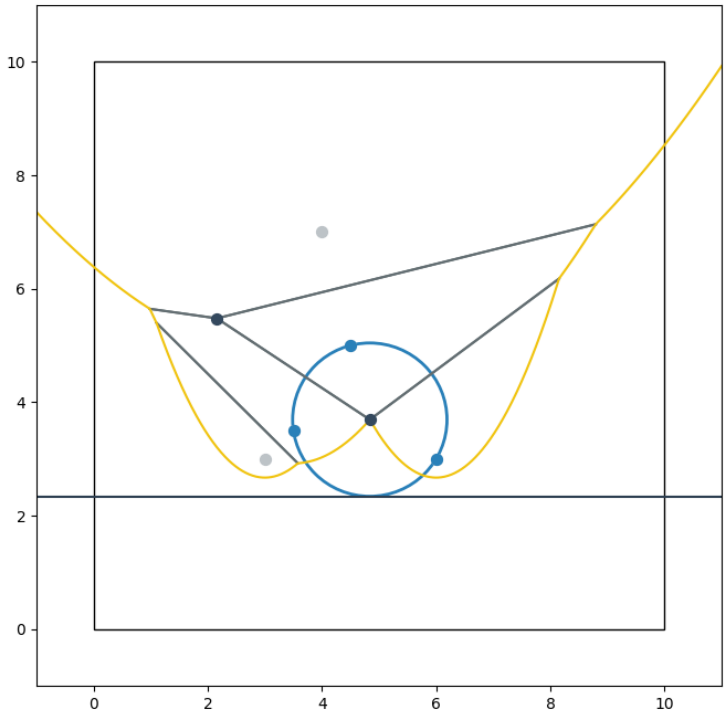
CircleEvent((P0, P1, P2), y-radius=3.09, y=5.48, radius=2.39)



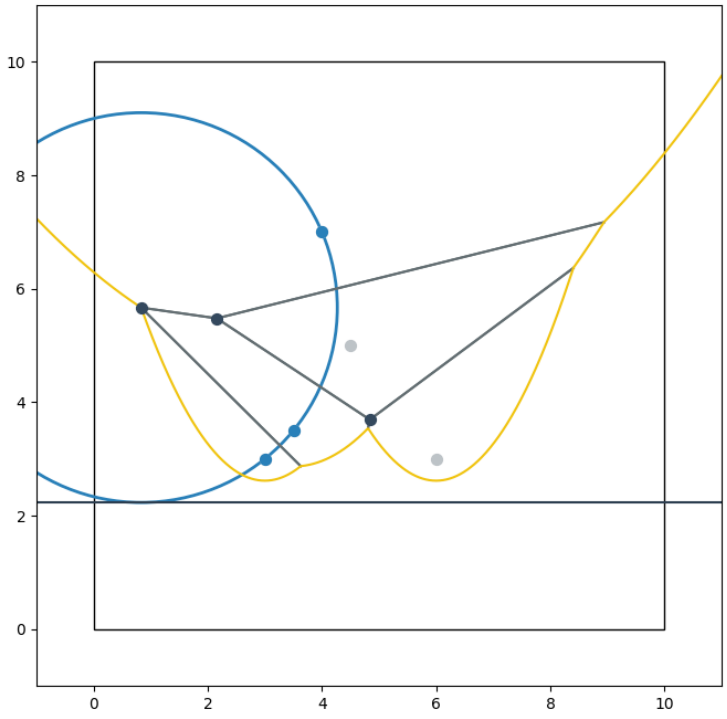

SiteEvent(x=3, y=3)
10
8
6
4
2
0
0
2
4
6
8
10


SiteEvent(x=6, y=3)
10
8
6
4
2
0
0
2
4
6
8
10

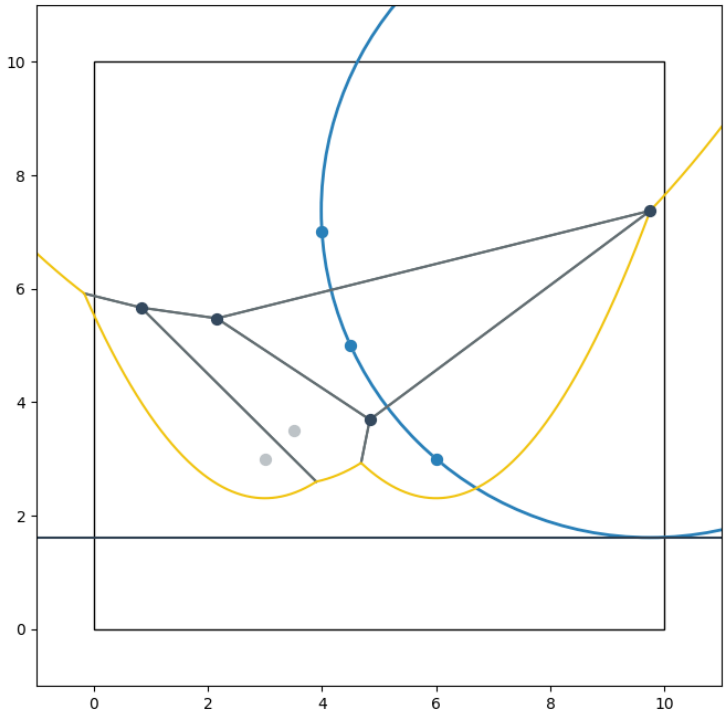
CircleEvent((P2, P1, P4), y-radius=2.34, y=3.69, radius=1.35)



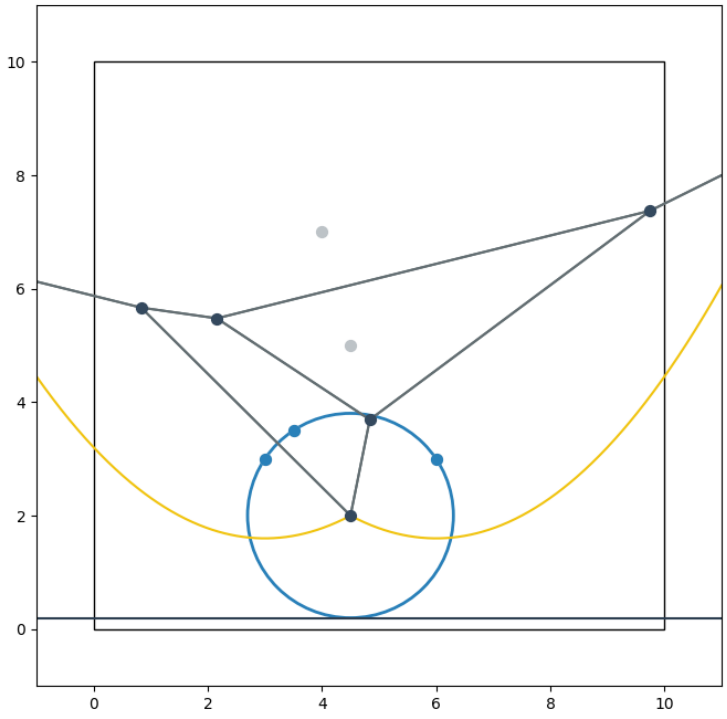
CircleEvent((P0, P2, P3), y-radius=2.23, y=5.67, radius=3.44)

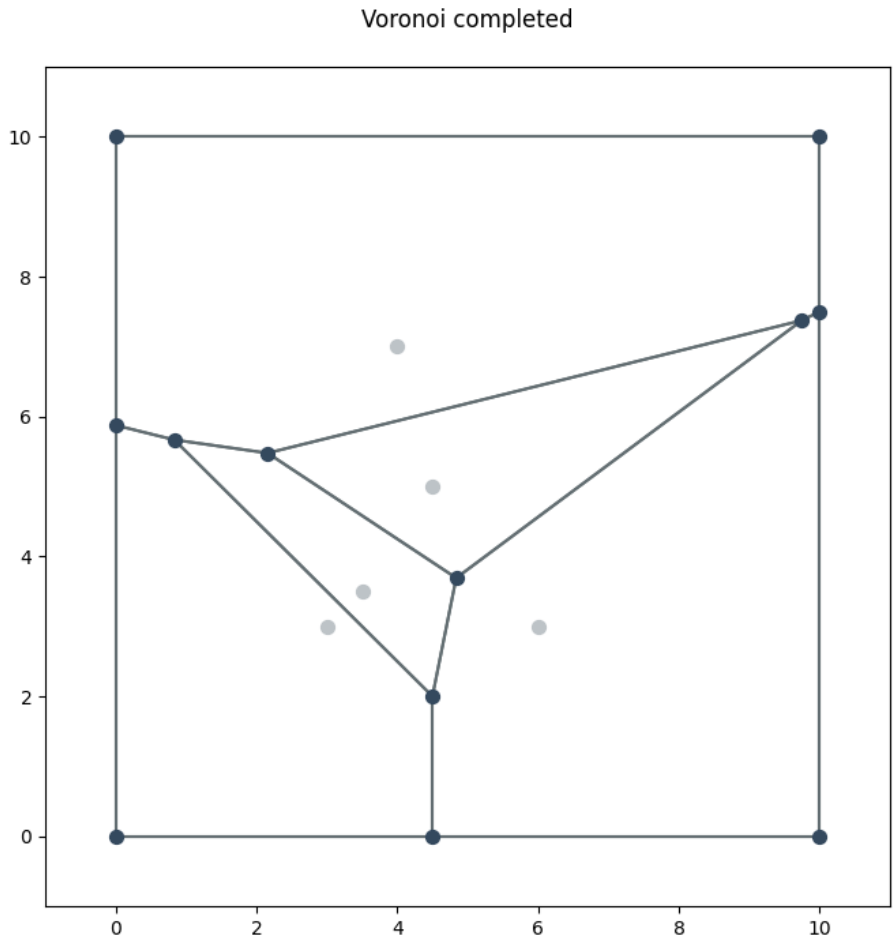


CircleEvent((P4, P1, P0), y-radius=1.61, y=7.38, radius=5.76)



CircleEvent((P3, P2, P4), y-radius=0.20, y=2.00, radius=1.80)




Voronoi completed
10
8
6
4
2
0
0
2
4
6
8
10