The Mathematica® Journal

Computing Exact Closed-Form Distance Distributions in Arbitrarily Shaped Polygons with Arbitrary Reference Point

Ross Pure
Salman Durrani

We propose and implement an algorithm to compute the exact cumulative density function (CDF) of the distance from an arbitrary reference point to a randomly located node within an arbitrarily shaped (convex or concave) simple polygon. Using this result, we also obtain the closed-form probability density function (PDF) of the Euclidean distance between an arbitrary reference point and its i^{th} neighbor node when N nodes are uniformly and independently distributed inside the arbitrarily shaped polygon. The implementation is based on the recursive approach proposed by Ahmadi and Pan [1] in order to obtain the distance distributions associated with arbitrary triangles. The algorithm in [1] is extended for arbitrarily shaped polygons by using a modified form of the shoelace formula. This modification allows tractable computation of the overlap area between a disk of radius r centered at the arbitrary reference point and the arbitrarily shaped polygon, which is a key part of the implementation. The obtained distance distributions can be used in the modeling of wireless networks, especially in the context of emerging ultra-dense small cell deployment scenarios, where network regions can be arbitrarily shaped. They can also be applied in other branches of science, such as forestry, mathematics, operations research, and material sciences.

1. Introduction

Wireless networks are generally modeled as a finite region in Euclidean space (this article considers those regions that are simple polygons in two-dimensional Euclidean space $\mathbb{R}^2$) with nodes independently and uniformly distributed throughout the region. The random distances between nodes or users, therefore, significantly affect the modeling of the wireless network, since the received signal power and interference depend critically on the distances between the transmitter and the receiver nodes [2]. As shown recently in [3], when nodes are independently and uniformly distributed within the network, the important distance distribution is the cumulative density function (CDF) of the distance from an arbitrary reference point to a randomly located node within the polygon (in this article, we use the phrase *distance distribution* to denote this CDF distribution). It can be obtained by finding the ratio of the overlap area between a disk of radius r centered at the arbitrary reference point and the area of the polygon. This CDF can then be used to obtain the probability density function (PDF) of the Euclidean distance between an arbitrary reference point and its i^{th} neighbor node when N nodes are uniformly and independently distributed inside the polygon.

Recently, there has been increasing interest in wireless communications to model the distance distributions in polygonal regions. In traditional cellular networks, the network region is often modeled as an equilateral triangle, a square, a hexagon, or a disk. For regular n-sided polygons, the distance distributions were obtained for the special case when the arbitrary reference point is located at the center of the polygon in [4] and for the general case when the arbitrary reference point is located anywhere inside the polygon in [3]. Note that a *Mathematica* implementation of the algorithm in [4] is available [5]. In emerging wireless network paradigms, such as ultra-dense small cell deployments, the network regions can be arbitrarily shaped. For arbitrarily shaped convex polygons, when the arbitrary reference point is located anywhere inside the polygon, an algorithm to obtain the distance distributions was proposed in [6]. For triangular regions, when the arbitrary reference point is located anywhere, an algorithm to obtain the distance distributions was proposed in [1]. The authors in [1] argued that since any polygon can be triangulated (i.e., broken up into non-overlapping triangles), their approach in principle could be applied to determine the distance distributions for the general case of arbitrary reference point location and arbitrarily shaped (convex or concave) polygons.

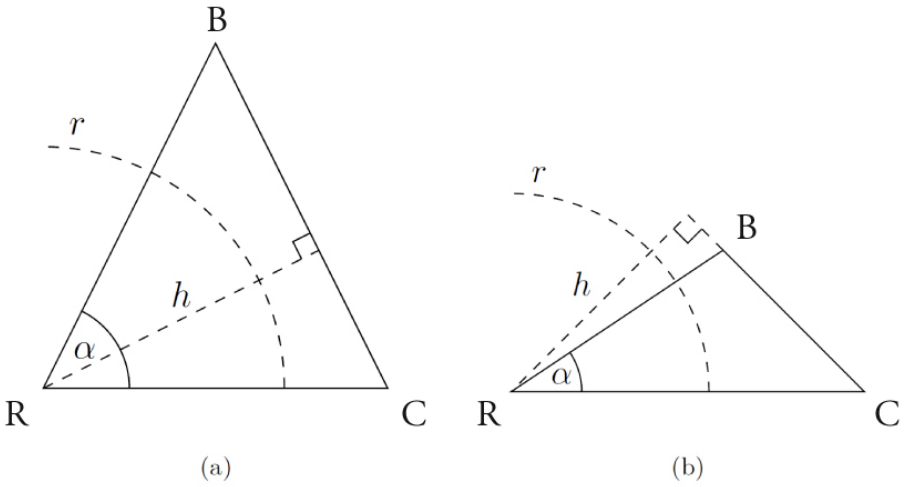
In this article, we extend the algorithm in [1] for arbitrarily shaped polygons by using a modified form of the shoelace formula. The shoelace formula, also known as Gauss's area formula, is a well-known mathematical method to determine the area of a polygon whose vertices are described by ordered pairs in $\mathbb{R}^2$ [7]. Our modification of the shoelace formula allows tractable computation of the overlap area between a disk of radius r centered at the arbitrary reference point and the arbitrarily shaped polygon, allowing the algorithm in [1] to be generalized and implemented.

This article is organized as follows. In the remainder of Section 1, we briefly summarize the algorithm in [1] and define the commonly used notation. In Section 2, we discuss the shoelace formula and its proposed modification. In Section 3, we present the proposed algorithm and its *Mathematica* implementation. The simulation results, which are used to verify the analytical results, are discussed in Section 4. An example illustrating the proposed *Mathematica* implementation is discussed in Section 5. Finally, conclusions are presented in Section 6.

1.1 Overview of Algorithm in [1] for Triangle Regions

Calculating the distance distribution (i.e. the CDF) evaluated at r is equivalent to finding the ratio of the area within the polygon that is less than a distance r from the reference point to the area of the polygon. The latter area is readily calculated if the polygon vertices are known. (Generally the polygon defining the network area has known coordinates, so the area may be calculated.) Hence, the challenge is calculating the former area.

It is clear that the CDF has an obvious geometric interpretation; if we let P be a polygon and $B_r(\mathrm{R})$ be the disc of radius r centered at some reference point R, then the CDF is the area of $P \bigcap B_r(\mathrm{R})$ divided by the area of the polygon. Ahmadi and Pan [1] perform this calculation for an arbitrary triangle and arbitrary reference point by first establishing the case for which the reference point is one of the triangle vertices, as illustrated in Figure 1.



▲ **Figure 1.** Depiction of the two characteristic cases considered by Ahmadi and Pan [1]. (a) shows the case that the altitude from R to the side BC is inside the triangle. (b) shows the case that the altitude from R to the side BC is outside the triangle.

They assume, without loss of generality, that the side length of RB is less than or equal to the side length of RC. The possible cases are then characterized by whether the altitude from R to the side BC is inside or outside the triangle, and considered separately (see Figure 1). For a disc centered at R, the area of intersection of the triangle △RBC and the disc as a function of the radius r is derived for each case. The result for the former case is

$$|\triangle \mathrm{RBC} \bigcap B_r(\mathrm{R})| =$$

$$\begin{cases} \frac{\alpha}{2} r^2 & 0 \le r \le h \\ h\sqrt{r^2 - h^2} + \frac{\alpha - 2\cos^{-1}\left(\frac{h}{r}\right)}{2} r^2 & h \le r \le |\mathrm{RB}| \\ \frac{h\left(\sqrt{|\mathrm{RB}|-h^2} + \sqrt{r^2-h^2}\right)}{2} + \frac{r^2}{2}\left(\alpha - \cos^{-1}\left(\frac{h}{|\mathrm{RB}|}\right) - \cos^{-1}\left(\frac{h}{r}\right)\right) & |\mathrm{RB}| \le r \le |\mathrm{RC}| \\ |\triangle \mathrm{RBC}| & |\mathrm{RC}| \le r \end{cases} \quad (1)$$

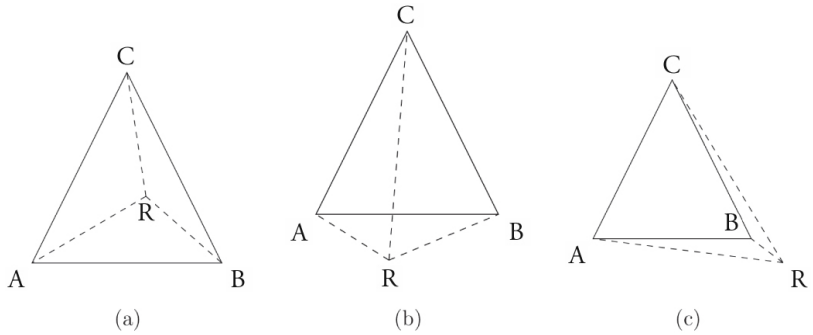
and for the latter case the result is

$$|\triangle \mathrm{RBC} \bigcap B_r(\mathrm{R})| = \begin{cases} \frac{\alpha}{2} r^2 & 0 \le r \le |\mathrm{RB}| \\ \frac{h\left(\sqrt{r^2-h^2} - \sqrt{|\mathrm{RB}|-h^2}\right)}{2} + \frac{\sin^{-1}\left(\frac{h}{r}\right) - \gamma}{2} r^2 & |\mathrm{RB}| \le r \le |\mathrm{RC}| \\ |\triangle \mathrm{RBC}| & |\mathrm{RC}| \le r \end{cases} \quad (2)$$

All the symbols in (1) and (2) are defined in Figure 1. The equations (1) and (2) are extended to an arbitrary triangle with an arbitrary reference point using triangle decomposition and adding and subtracting the areas appropriately [1]. The three possible cases are that the reference point is inside the triangle, the reference point is outside the triangle and in the area formed by the extension of two edges from a vertex, or otherwise (see Figure 2). For these three cases, the area $|\triangle \mathrm{ABC} \bigcap B_r(\mathrm{R})|$ is given by

$$|\triangle \mathrm{ABC} \bigcap B_r(\mathrm{R})| = |\triangle \mathrm{RAB} \bigcap B_r(\mathrm{R})| + |\triangle \mathrm{RBC} \bigcap B_r(\mathrm{R})| + |\triangle \mathrm{RCA} \bigcap B_r(\mathrm{R})|,$$
$$|\triangle \mathrm{ABC} \bigcap B_r(\mathrm{R})| = |\triangle \mathrm{RCA} \bigcap B_r(\mathrm{R})| - |\triangle \mathrm{RAB} \bigcap B_r(\mathrm{R})| + |\triangle \mathrm{RBC} \bigcap B_r(\mathrm{R})|,$$
$$|\triangle \mathrm{ABC} \bigcap B_r(\mathrm{R})| = |\triangle \mathrm{RCA} \bigcap B_r(\mathrm{R})| + |\triangle \mathrm{RBC} \bigcap B_r(\mathrm{R})| - |\triangle \mathrm{RAB} \bigcap B_r(\mathrm{R})|,$$

respectively, where the individual intersection areas can be found using (1) and (2) appropriately.



▲ **Figure 2.** Possible cases for triangle decompositions given an arbitrary triangle and reference point. (a) is the case of an interior reference point. (b) and (c) show the case for an exterior reference point. (c) is the case that the reference point is in the area formed by the extension of the edges from one vertex.

Using this result, an algorithm to compute the CDF for the general case of an arbitrary polygon and arbitrary reference point is proposed and implemented. This is achieved by first establishing a modification of the shoelace formula, which is described and proved in Section 2.

1.2 Definitions

In this subsection, we define some functions and notation that are used throughout this article.

Definition 1

The area of a region $S \subset \mathbb{R}^2$ is denoted by $|S|$. In the case of a triangle with vertices $T = \{(x_1, y_1), (x_2, y_2), (x_3, y_3)\}$, the area can be calculated as in [8].

$$|T| = \frac{1}{2}\text{abs}\left(\det\begin{pmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{pmatrix}\right). \tag{3}$$

Definition 2

The signed area of a triangle T with vertices $T = \{(x_1, y_1), (x_2, y_2), (x_3, y_3)\}$ is denoted by $|T|_s$, and is given by

$$|T|_s = \frac{1}{2}\det\begin{pmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{pmatrix}. \tag{4}$$

(The subscript s stands for "signed".) We note that from the above definition, $|T|_s$ has the same magnitude as the area of the triangle, $|T|$, but is positive if the vertices are given in counterclockwise order, and negative if the vertices are given in clockwise order.

Definition 3

The signed area of the region defined by the intersection of a triangle T and a region $S \subset \mathbb{R}^2$, that is, $T \bigcap S$, is denoted $|T \bigcap S|_s$ and is given by

$$|T \bigcap S|_s = \text{sgn}(|T|_s)\,|T \bigcap S|, \tag{5}$$

where $\text{sgn}(x)$ *is the signum function.*

Essentially, this says that $|T \bigcap S|_s = \pm|T \bigcap S|$, where $|T \bigcap S|_s$ is positive if the signed area of T is positive and negative if the signed area of T is negative.

2. Shoelace Formula and Its Modification

The shoelace formula is a useful formula for calculating the area of a polygon (see the illustration in [6] for an explanation of the name). It is stated in the theorem below [9].

Theorem 1

The shoelace formula: Let $P = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ *be an ordered set of vertices in the plane that define an n-sided polygon. Define the triangles* $T_1, T_2, \ldots, T_n$ *as the triangles formed from two adjacent points from P and the origin. That is,*

$$T_1 = \{(0, 0), (x_1, y_1), (x_2, y_2)\},$$
$$T_2 = \{(0, 0), (x_2, y_2), (x_3, y_3)\},$$
$$\ldots$$
$$T_n = \{(0, 0), (x_n, y_n), (x_1, y_1)\}.$$

Then the area of the polygon is given by

$$|P| = \frac{1}{2}\,\text{abs}\left(\sum_{j=1}^{n} x_j\, y_{j+1} - x_{j+1}\, y_j\right), \tag{6}$$

where for simplicity we let $x_{n+1} = x_1$ *and* $y_{n+1} = y_1$.

The shoelace formula holds if instead of using the origin to define each T_j, we used an arbitrary point in $\mathbb{R}^2$.

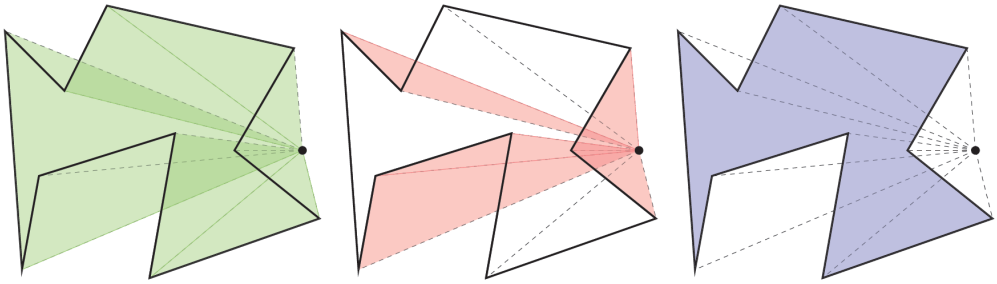
Notice that

$$|T_j|_s = \frac{1}{2}\det\begin{pmatrix} 0 & 0 & 1 \\ x_j & y_j & 1 \\ x_{j+1} & y_{j+1} & 1 \end{pmatrix} =$$
$$\frac{1}{2}\left(0\det\begin{pmatrix} y_j & 1 \\ y_{j+1} & 1 \end{pmatrix} - 0\det\begin{pmatrix} x_j & 1 \\ x_{j+1} & 1 \end{pmatrix} + 1\det\begin{pmatrix} x_j & y_j \\ x_{j+1} & y_{j+1} \end{pmatrix}\right) = \frac{1}{2}(x_j\, y_{j+1} - x_{j+1}\, y_j).$$

Hence, the shoelace formula can alternatively be stated as

$$|P| = \frac{1}{2}\,\text{abs}\left(\sum_{j=1}^{n} |T_j|_s\right). \tag{7}$$

A visual illustration of (7) is shown in Figure 3. The triangles with positively signed area are shaded in green and shown in (a), and the triangles with negatively signed area are shaded in orange and shown in (b). In both cases, the darker regions indicate where triangles overlap. We can thus think of the shoelace formula as stating that if we add the green regions and subtract the orange regions, we obtain the region defined by the polygon (shown in (c)). In the given example in Figure 3 we can see this visually because the green regions outside the polygon "cancel" with the orange regions outside the polygon, and the dark green regions inside the polygon are "canceled" by orange regions inside the polygon.



▲ **Figure 3.** Visual illustration of the shoelace formula for calculating the area of a polygon.

We now build upon the shoelace formula in (7) to obtain a useful modification.

Theorem 2

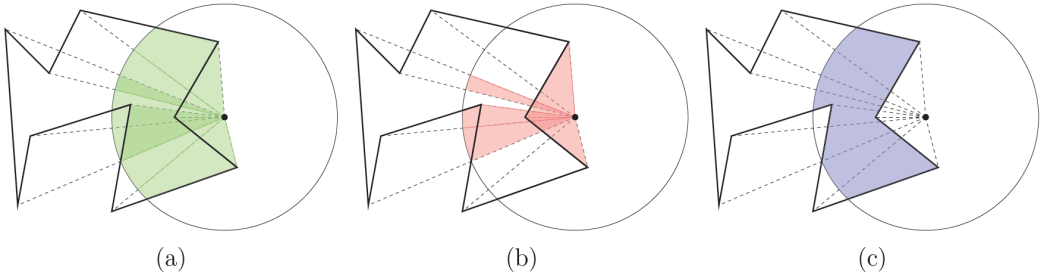
The modified shoelace formula: Let $P = \{p_1, p_2, \ldots, p_n\}$ *be an ordered set of vertices in the plane that define an n-sided polygon. Let* $B_r(\mathrm{R}) \subset \mathbb{R}^2$ *be a disc of radius* $r \geq 0$ *centered at the point* R. *Define the triangles* $T_1, T_2, \ldots, T_n$ *as the triangles formed from two adjacent points from P and the reference point* R. *That is,*

$T_1 = \{\mathrm{R}, p_1, p_2\}$,
$T_2 = \{\mathrm{R}, p_2, p_3\}$,
…
$T_n = \{\mathrm{R}, p_n, p_1\}$.

Then

$$|B_r(\mathrm{R}) \bigcap P| = \mathrm{abs}\left(\sum_{j=1}^{n} |B_r(\mathrm{R}) \bigcap T_j|_{\mathrm{s}}\right). \tag{8}$$

Thus, just as the area of the polygon was equal to the sum of the signed areas of the triangles $T_1, T_2, \ldots, T_n$ in the original shoelace formula, we have that the area of intersection of the polygon and a disc is given by the sum of the signed areas of intersection of each $T_1, T_2, \ldots, T_n$ and the disc in the modified shoelace formula. A visual illustration of this modified formula is shown in Figure 4. If we consider the same example as in Figure 3 but with the addition of a disc, as depicted in Figure 4, then the orange regions (areas with negative signed area, shown in (b)) "cancel" the green regions (areas with positive signed area, shown in (a)) that are outside the polygon and "cancel" the darker regions inside the polygon, giving the desired area of intersection, shown in (c). We prove theorem 2 using induction.

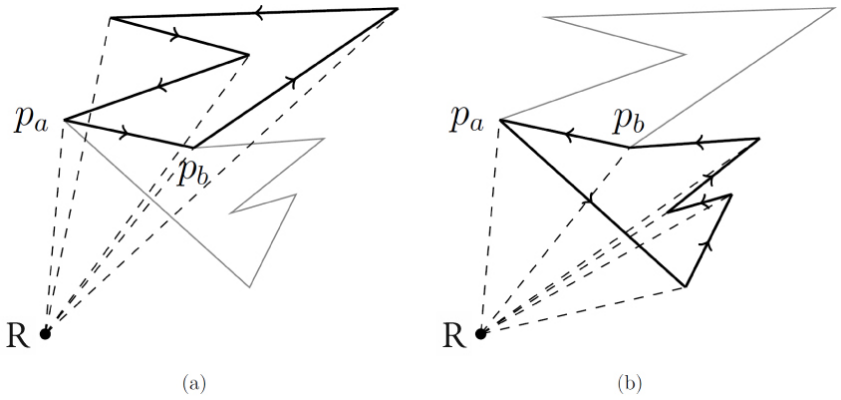


▲ **Figure 4.** Visual illustration of the modified shoelace formula for calculating overlap areas.

Proof of Theorem 2

We use strong induction on n (the number of sides of the polygon). The base case $n = 3$ is immediate, as Ahmadi and Pan [1] show exhaustively in their paper. We now assume that the result is true for all $n \leq k$ for some $k \geq 3$, and let P_{k+1} be an arbitrary $(k+1)$-sided polygon. We may choose two vertices v_1, v_2 of the polygon P_{k+1} such that the straight line joining v_1 and v_2 is contained within the polygon P_{k+1} (for a proof of this fact, see [10]).

To complete the induction step, let $P_{k+1} = \{p_1, p_2, \ldots, p_{k+1}\}$ and pick any diagonal $p_a p_b$ contained within P_{k+1}, where p_a appears before p_b in the list. Then let $P_{k+1} = S_1 \bigcup S_2$, where S_1 and S_2 are the two polygons formed by adding the diagonal. We can thus write $S_1 = \{p_1, p_2, \ldots, p_{a-1}, p_a, p_b, p_{b+1}, \ldots, p_{k+1}\}$ and $S_2 = \{p_a, p_{a+1}, \ldots, p_{b-1}, p_b\}$. These two polygons correspond to (a) and (b), respectively, in the example given in Figure 5.



▲ **Figure 5.** Example polygon decomposition given the diagonal $p_a p_b$. The arrows denote the order of the vertices.

Since both S_1 and S_2 have at most k sides (say p and q sides, respectively), then for any R and any $r \geq 0$, we know that for the disc of radius r centered on R (denoted $B_r(\mathrm{R})$), we have

$$|B_r(\mathrm{R}) \bigcap S_1| = \mathrm{abs}\left(\sum_{j=1}^{p} \left|B_r(\mathrm{R}) \bigcap T_j^{(1)}\right|_{\mathrm{s}}\right),$$

$$|B_r(\mathrm{R}) \bigcap S_2| = \mathrm{abs}\left(\sum_{j=1}^{q} \left|B_r(\mathrm{R}) \bigcap T_j^{(2)}\right|_{\mathrm{s}}\right),$$

where $T_1^{(1)}, \ldots, T_p^{(1)}$ and $T_1^{(2)}, \ldots, T_q^{(2)}$ are the triangles defined in the same way as in the statement of the theorem for the polygons S_1 and S_2, respectively. From their construction, the vertices in S_1 and S_2 are either both in clockwise order, or both in counterclockwise order. So $|B_r(\mathrm{R}) \bigcap S_1|$ and $|B_r(\mathrm{R}) \bigcap S_2|$ are either both positive or both negative. Hence

$$\begin{aligned} &\mathrm{abs}\left(\sum_{j=1}^{p} \left|B_r(\mathrm{R}) \bigcap T_j^{(1)}\right|_{\mathrm{s}}\right) + \mathrm{abs}\left(\sum_{j=1}^{q} \left|B_r(\mathrm{R}) \bigcap T_j^{(2)}\right|_{\mathrm{s}}\right) = \\ &\mathrm{abs}\left(\sum_{j=1}^{p} \left|B_r(\mathrm{R}) \bigcap T_j^{(1)}\right|_{\mathrm{s}} + \sum_{j=1}^{q} \left|B_r(\mathrm{R}) \bigcap T_j^{(2)}\right|_{\mathrm{s}}\right). \end{aligned} \tag{9}$$

Using (7) we deduce that

$$|B_r(\mathrm{R}) \bigcap P_{k+1}| =$$

$$|B_r(\mathrm{R}) \bigcap S_1| + |B_r(\mathrm{R}) \bigcap S_2| = \mathrm{abs}\left(\sum_{j=1}^{p} \left|B_r(\mathrm{R}) \bigcap T_j^{(1)}\right|_{\mathrm{s}} + \sum_{j=1}^{q} \left|B_r(\mathrm{R}) \bigcap T_j^{(2)}\right|_{\mathrm{s}}\right).$$

But all of the triangles $T_i^{(1)}, T_j^{(2)}$ are precisely the analogous triangles $T_1, T_2, \ldots, T_{k+1}$ for the polygon P_{k+1}, and also include the triangles added by the diagonal, namely $\triangle \mathrm{R}\, p_a\, p_b$ and $\triangle \mathrm{R}\, p_b\, p_a$. Thus

$$|B_r(\mathrm{R}) \bigcap P_{k+1}| = \mathrm{abs}\left(\sum_{j=1}^{p} \left|B_r(\mathrm{R}) \bigcap T_j^{(1)}\right|_{\mathrm{s}} + \sum_{j=1}^{q} \left|B_r(\mathrm{R}) \bigcap T_j^{(2)}\right|_{\mathrm{s}}\right) =$$

$$\mathrm{abs}\left(\sum_{j=1}^{k+1} |B_r(\mathrm{R}) \bigcap T_j|_{\mathrm{s}} + |B_r(\mathrm{R}) \bigcap \triangle \mathrm{R}\, p_a\, p_b|_{\mathrm{s}} + |B_r(\mathrm{R}) \bigcap \triangle \mathrm{R}\, p_b\, p_a|_{\mathrm{s}}\right).$$

But $\triangle \mathrm{R}\, p_a\, p_b$ and $\triangle \mathrm{R}\, p_b\, p_a$ have opposite orientation, so $|B_r(\mathrm{R}) \bigcap \triangle \mathrm{R}\, p_a\, p_b|_{\mathrm{s}} + |B_r(\mathrm{R}) \bigcap \triangle \mathrm{R}\, p_b\, p_a|_{\mathrm{s}} = 0$, so that

$$|B_r(\mathrm{R}) \bigcap P_{k+1}| = \mathrm{abs}\left(\sum_{j=1}^{k+1} |B_r(\mathrm{R}) \bigcap T_j|_{\mathrm{s}}\right),$$

as required. □

3. Proposed Algorithm and Implementation

In this section, we extend the algorithm in [1] for arbitrarily shaped polygons by using the modified form of the shoelace formula in theorem 2. Since any polygon can be triangulated, the area of intersection of a disc of radius r centered at a reference point R and the polygon can be found by summing the areas of the intersections of the disc and each triangle of the triangulation. Since the latter areas can be found using theorem 2, the generalization follows.

3.1 Algorithm

We know that given a polygon P and a reference point R, the CDF $F(r)$ is

$$F(r) = \frac{|P \bigcap B_r(\mathrm{R})|}{|P|}. \tag{10}$$

Using the modified shoelace formula, we may write (10) as

$$F(r) = \frac{1}{|P|}\,\mathrm{abs}\left(\sum_{j=1}^{n} |T_j \bigcap B_r(\mathrm{R})|_{\mathrm{s}}\right), \tag{11}$$

where $P = \{p_1, p_2, \ldots, p_n\}$ and

$$T_1 = \{\mathrm{R}, p_1, p_2\},$$
$$T_2 = \{\mathrm{R}, p_2, p_3\},$$
$$\ldots$$
$$T_n = \{\mathrm{R}, p_n, p_1\}.$$

In (11), we can calculate $|P|$ using the shoelace formula in theorem 1. The method to calculate each $|T_j \bigcap B_r(\mathrm{R})|_{\mathrm{s}}$ was given in [1] and is summarized in Section 1.1. Once the CDF $F(r)$ is found, the corresponding PDF $f(r)$ can be found by differentiation.

3.2 Implementation

In this section, we describe the implementation of the algorithm. The functions in (1) and (2) are implemented as the functions `InsideAltitudeArea` and `OutsideAltitudeArea`, respectively. These functions return the piecewise functions (1) and (2) as a function of the argument r. Each function returns a list of sublists; here each sublist is of the form $\{g(r), a, b\}$, where $g(r)$ is the piece of the function that corresponds to the range $a \le r \le b$. The function (1) has four pieces and so `InsideAltitudeArea` returns a 4×3 array. Similarly, `OutsideAltitudeArea` returns a 3×3 array, as it corresponds to (2), which has three pieces.

```
InsideAltitudeArea[R_, B_, C_, r_] := Module[
  {RB, BC, RC, s, tRBC, h, α},
  RB = Norm[R - B];
  BC = Norm[B - C];
  RC = Norm[R - C];
  If[
   R == B || R == C || B == C,
   {{0, 0, ∞}},
   (* else *)
   s = (RB + BC + RC)/2;
   tRBC = √(s (s - RB) (s - BC) (s - RC));
   h = (2 tRBC)/BC;
   α = Norm[VectorAngle[B - R, C - R]];
   {
    {α/2 r^2, 0, h},
    {h √(r^2 - h^2) + (α/2 - ArcCos[h/r]) r^2, h, RB},
    {h/2 (√(RB^2 - h^2) + √(r^2 - h^2)) +
      r^2/2 (α - (ArcCos[h/RB] + ArcCos[h/r])), RB, RC},
    {tRBC, RC, ∞}
   }
  ]
 ]
```

```
OutsideAltitudeArea[R_, B_, C_, r_] := Module[
  {RB, BC, RC, s, tRBC, h},
  If[R == B || R == C || B == C,
   {{0, 0, ∞}},
   (* else *)
   RB = Norm[R - B];
   BC = Norm[B - C];
   RC = Norm[R - C];
   s = (RB + BC + RC)/2;
   tRBC = √(s (s - RB) (s - BC) (s - RC));
   h = (2 tRBC)/BC;
   {
    {(r^2/2) Norm[VectorAngle[B - R, C - R]], 0, RB},
    {(h/2) (√(r^2 - h^2) - √(RB^2 - h^2)) +
       (r^2/2) (ArcSin[h/r] - Norm[VectorAngle[R - C, B - C]]),
     RB, RC},
    {tRBC, RC, ∞}
   }
  ]
 ]
```

The function `PolygonArea`, given the vertices, calculates the area of a polygon with the shoelace formula.

```
PolygonArea[P_] :=
 1 / 2 Total[Det /@ Partition[Append[P, First@P], 2, 1]]
```

`CombinePieces` is responsible for simplifying a "pseudo-piecewise" function by determining the distinct ranges of the equivalent piecewise function, sorting these ranges, and finding the corresponding function for each range. This piecewise function is then converted to either the CDF or the PDF, depending on the argument `case`. The function takes four arguments: f is a "pseudo-piecewise" function of the form that is returned by `InsideAltitudeArea` and `OutsideAltitudeArea`, namely, a list of sublists, where each sublist is of the form $\{g(r), a, b\}$ where $g(r)$ is the piece of the function that corresponds to the range $a \le r \le b$. The argument `area` is set equal to the area of the polygon. The returned CDF is a function of the argument r. Finally, `case` is a Boolean argument that determines whether the CDF or PDF is returned; when it is true the function returns the CDF, and when it is false the function returns the PDF.

```
CombinePieces[f_, area_, r_, case_] := Module[
  {t, indices, Additions, Subtractions, cFunction},
  (*Identify the different ranges for the piecewise
   function*)
  indices = Partition[
    Sort[DeleteDuplicates[Flatten[{#[[2]], #[[3]]} & /@ f]],
     Less],
    2, 1
   ];
  (*
    Take the function bounds from the list t.
    Remove the duplicate elements.
    Sort the elements in ascending order.
    Make ranges from adjacent elements.
  *)

  (* Create the list describing when to add to the
   cumulative function, cFunction. *)
  Additions = {#[[1]], #[[2]]} & /@
    Sort[f, #1[[2]] < #2[[2]] &];
  (*
    Sort the list according to the range lower bound.
    Remove the range upper bound from each element.
  *)

  (* Create the list describing when to subtract
   from the cumulative function, cFunction. *)
  Subtractions = {-#[[1]], #[[3]]} & /@
    Sort[f, #1[[3]] < #2[[3]] &];
  (*
    Sort the list according to the range upper bound.
    Remove the range lower bound from each element
   and make the function negative.
  *)

  (* Construct the cumulative function, cFunction. *)
  cFunction =
   Accumulate[
    Total /@
     Map[
      #[[1]] &,
      GatherBy[
       Sort[
        Join[Additions, Subtractions],
        #1[[2]] < #2[[2]] &
       ],
       #[[2]] &
      ],
```

```
    {2}
   ]
 ];
(*
  Collect all of the addition and subtraction
 information.
  Sort the elements in ascending order by the
 value at which they are added/subtracted.
  Group the functions by the value at which they
 are added/subtracted.
  Find the sum of all previous functions at each
 range boundary.
  Delete the FoldList artifact.
*)

(* Convert to CDF in Mathematica Piecewise form. *)
Piecewise[
 DeleteCases[
  If[
     case,
     {#[[1]], #[[2, 1]] ≤ r ≤ #[[2, 2]]},
     {D[#[[1]], r], #[[2, 1]] ≤ r ≤ #[[2, 2]]}
    ] & /@ Partition[
    Riffle[FullSimplify[cFunction/area], indices],
    2],
  x_ /; x[[2, 1]] == x[[2, 3]]
 ]
]
(*
  Convert the area function to the CDF.
  Gather the functions and their valid ranges.
  Group the functions with the value at which they
 become active.
  Convert to Mathematica Piecewise form.
  Manually delete pieces with zero range that
 GatherBy misses.
*)
]
```

This gives the sign of a triangle's area.

```
SignTriangleArea[{R_, A_, B_}] :=
 Sign@Det[{Flatten[{1, R}], Flatten[{1, A}],
    Flatten[{1, B}]}]
```

`AltitudeInsideQ` returns `True` if the altitude of the triangle intersects the opposite edge inside the triangle.

```
AltitudeInsideQ[R_, A_, B_] := Module[{AB = B - A},
  (* Test if the altitude lies inside the triangle
   by checking that the component of the side vectors
   from R in the direction of AB are both less than
   the length of AB. *)
  Norm[Dot[A - R, AB]] < Dot[AB, AB] &&
   Norm[Dot[B - R, AB]] < Dot[AB, AB]
 ]
```

`PolygonCDF` is the main function used to compute the CDF; it uses the method outlined in Subsection 3.1.

```
PolygonCDF[P_, R_, r_, case_] := Module[{Q, f, i, A, B},
  (* Make P cyclic. *)
  Q = Append[P, First[P]];
  (* Initialize f. *)
  f = {};
  (* Loop over each triangle formed by the reference
   point and adjacent vertices of the polygon,
  obtaining the area function for each. *)
  For[i = 1, i ≤ Length[P], i++,
   (* Create triangle and side vectors. *)
   {A, B} = Q[[{i, i + 1}]];
   (* Calculate the appropriate area function given
    the characteristics of the triangle. *)
   f = Join[f, Map[
      If[
       SignTriangleArea[{R, A, B}] == 1,
       Identity,
       (* The sign of the area of the triangle is
        positive. *)
       {-1, 1, 1} # &
       (*
          The sign of the area of the triangle is
         negative.
          Multiply the function but not the ranges by -1.
       *)
      ],
      Apply[
       If[AltitudeInsideQ[R, A, B], InsideAltitudeArea,
```

```
        OutsideAltitudeArea],
       If[Norm[A - R] > Norm[B - R], {R, B, A, r}, {R, A, B, r}]
      ]
     ]]
  ];
  (*Convert the area functions for Piecewise. *)
  CombinePieces[FullSimplify[f], PolygonArea[P], r, case]
 ]
```

4. Simulation Aspects

This section discusses how to generate the simulation results, which are used to verify the analytical results. In general, we need uniformly distributed points inside arbitrarily shaped polygons.

Generating uniformly distributed points inside a triangle is straightforward and can be accomplished in a number of ways [11, 12]. The method used here selects two numbers at random from (0, 1) to measure off lengths on two sides of the triangle to use as weights on the vertices. `RandomPointsTriangle` has two arguments: `T` is a list of three vertices describing a triangle and `n` is the number of points to generate.

```
RandomPointsTriangle[{a_, b_, c_}, n_] := Module[{u, v},
  {u, v} = Transpose[Sort /@ RandomReal[1, {n, 2}]];
  Map[# a &, u] + Map[# b &, (v - u)] + Map[# c &, (1 - v)]]
```

The method for triangles is extended to arbitrary polygons by triangulating the polygon P, uniformly picking a triangle, and then generating a point in that triangle. Uniformly picking a triangle means choosing each triangle with probability such that the points generated are uniform for the whole polygon; this means that the probability of picking a triangle T_j must be proportional to its area. Precisely, the probability is $|T_j| / |P|$. This is done efficiently using `TriangulateMesh`.

```
RandomPointsPolygon[P_, n_] := Module[
  {p, s, t},
  p = N[P];
  (* Numerically approximate vertices for faster
   computation. *)
  s =
   First /@
    MeshPrimitives[
     TriangulateMesh[
      DiscretizeGraphics@Graphics[Polygon@p],
      MaxCellMeasure → Infinity],
     2];
  t = Accumulate[PolygonArea[#] & /@ s / PolygonArea[p]];
```

```
(*
  Triangulate the polygon.
  Calculate the area of the polygon.
  Associate each triangle with its fraction of area
 of the polygon.
  Associate each triangle with a range calculated
 as the sum of all previous
  area fractions. This allows a triangle to be
 picked from a random variable
  generated between 0 and 1.
*)
Select[
 Map[
  Function[
   x,
   Flatten@RandomPointsTriangle[
     s[[First@FirstPosition[x < # & /@ t, True]]], 1]
  ],
  RandomReal[1, n]
 ],
 # ≠ {} &]
(*
   Random points:
   Generate n random numbers between 0 and 1.
   For each random number,
 pick the corresponding triangle and generate a
  point in it.
*)
]
```

5. Examples

For convenience, here is a list of the different special examples that have been considered in the literature that can be verified using our implementation. The polygon vertices must be specified in clockwise order.

1. Example 1 in [1]: equilateral triangle with interior reference point

```
P1 = {{0, 0}, {1, 0}, {1/2, Sqrt[3]/2}};
R1 = {0, 0};
```

2. Example 2 in [1]: equilateral triangle with exterior reference point.

```
P2 = {{0, 0}, {1, 0}, {1/2, Sqrt[3]/2}};
R2 = {1/2, -Sqrt[3]/2};
```

3. Section 2.5.1, p. 263 in [13]: triangle with reference point at a vertex.

```
P3 = {{0, 0}, {1, 0}, {1/3, 1/2}};
R3 = {0, 0};
```

4. Example in [3]: square with reference point on the boundary.

```
P4 = {{1, 0}, {0, 1}, {-1, 0}, {0, -1}};
R4 = {1/2, -1/2};
```

5. Example in [4]: hexagon with reference point at the center.

```
P5 = {{1, 0}, {1/2, Sqrt[3]/2}, {-1/2, Sqrt[3]/2}, {-1, 0}, {-1/2, -Sqrt[3]/2},
    {1/2, -Sqrt[3]/2}};
R5 = {0, 0};
```

6. Example in [6], [14], and [15]: arbitrarily shaped convex polygon.

```
P6 = {{0, 0}, {Sqrt[3], 0}, {Sqrt[3] - Sqrt[3]/Sqrt[2], Sqrt[3]/Sqrt[2]}, {0, 1}};
R6 = {Sqrt[3], 0};
```

7. Example in [18]: arbitrarily shaped convex polygon.

```
P7 = {
    0.7 {Cos[50 Pi / 180], Sin[50 Pi / 180]},
    1.0 {Cos[80 Pi / 180], Sin[80 Pi / 180]},
    0.2 {Cos[230 Pi / 180], Sin[230 Pi / 180]},
    0.65 {Cos[320 Pi / 180], Sin[320 Pi / 180]},
    0.6 {Cos[340 Pi / 180], Sin[340 Pi / 180]}
   };
R7 = {0, 0};
```

8. Example in this report: arbitrarily shaped polygon shaped like the letter N.

```
P8 = {{0, 0}, {0.2, 0}, {0.2, 0.7}, {0.8, 0}, {1, 0},
    {1, 1}, {0.8, 1}, {0.8, 0.3}, {0.2, 1}, {0, 1}};
R8 = {0.2, 1.5};
```

9. Example in this report: star-shaped (concave) polygon region with reference point at the center.

$$\texttt{P9 = With}\left[\left\{\texttt{a} = \sqrt{\frac{5}{8} - \frac{\sqrt{5}}{8}}\,,\ \texttt{b} = \frac{1}{4}\left(-1 - \sqrt{5}\right),\right.\right.$$

$$\texttt{c} = \frac{1}{4}\left(-1 + \sqrt{5} - \frac{2\sqrt{5\left(5 + \sqrt{5}\right)}}{\sqrt{5 - \sqrt{5}} + \sqrt{5 + \sqrt{5}}}\right),\ \texttt{d} = \frac{1}{2}\sqrt{\frac{1}{2}\left(5 - \sqrt{5}\right)}\,,$$

$$\left.\texttt{e} = \frac{1}{2}\sqrt{\frac{1}{2}\left(5 + \sqrt{5}\right)}\,,\ \texttt{f} = \sqrt{\frac{5}{8} + \frac{\sqrt{5}}{8}}\,,\ \texttt{g} = \frac{1}{4}\left(-1 + \sqrt{5}\right)\right\},$$

```
    {{-a, b}, {0, c}, {a, b}, {- e c, g c}, {f, g},
     {- d c, b c}, {0, 1}, { d c, b c}, {-f, g}, { e c, g c}}];
R9 = {0, 0};
```

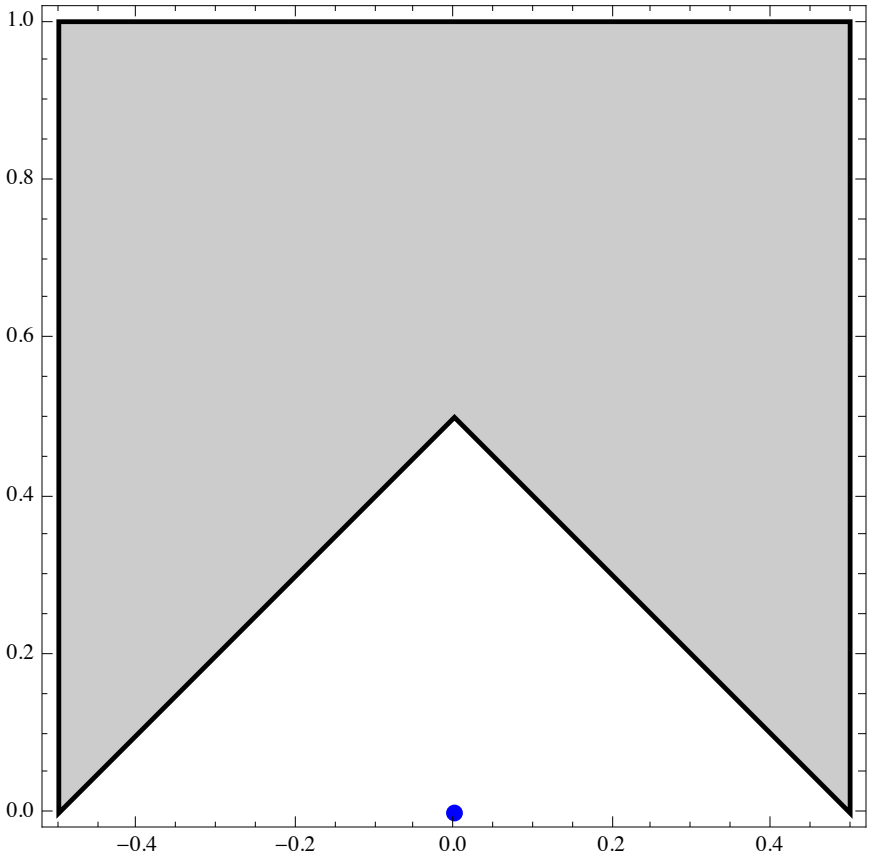
10. Example in this report: arbitrarily shaped concave polygon with exterior reference point.

$$\texttt{P10} = \left\{\left\{-\frac{1}{2}, 0\right\}, \left\{0, \frac{1}{2}\right\}, \left\{\frac{1}{2}, 0\right\}, \left\{\frac{1}{2}, 1\right\}, \left\{-\frac{1}{2}, 1\right\}\right\};$$

```
R10 = {0, 0};
```

We illustrate using Example 10.

```
Poly = P10;
ReferencePoint = R10;

Graphics[{EdgeForm[Thick], Lighter[Gray, 0.6],
  Polygon[Poly],
  Blue, PointSize[Large], Point[ReferencePoint]},
 Frame → True]
```

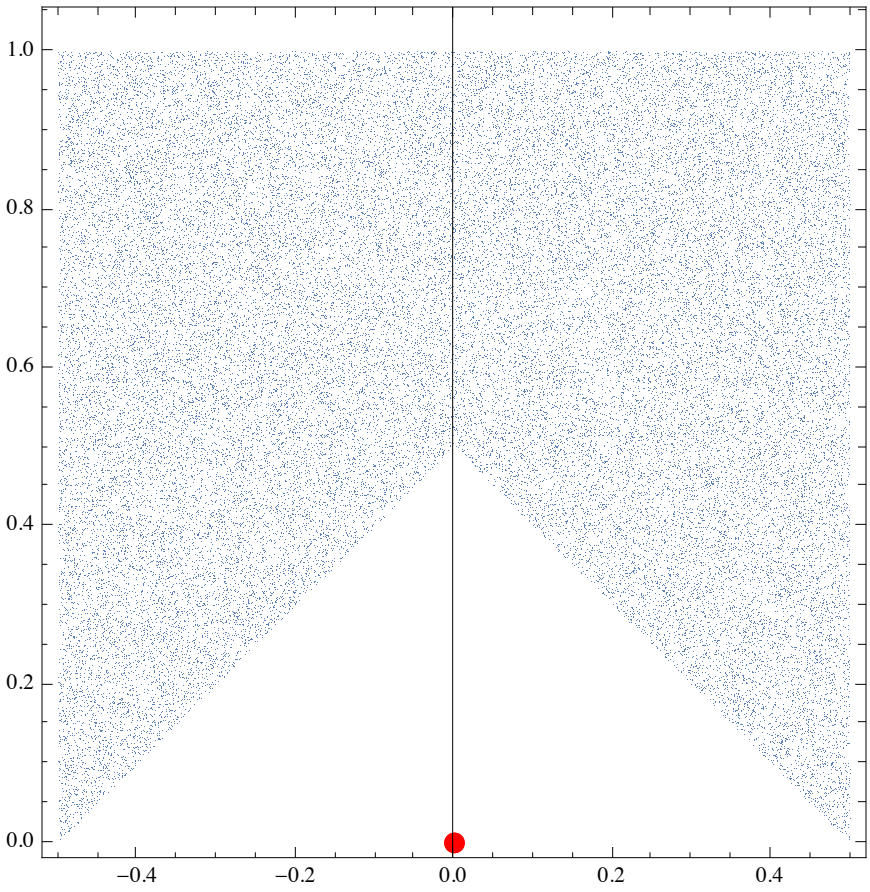


The simulation data is calculated. The code checks to see if the polygon is convex. If it is, the points are simulated using `RandomPointsConcave`, which is faster than `RandomPointsPolygon`. Otherwise, the code uses `RandomPointsPolygon`.

```
n = 50 000;
data = RandomPointsPolygon[Poly, n];
```

This shows the simulated uniformly distributed points for the given polygon.

```
Show[ListPlot[data, AspectRatio → Automatic],
 ListPlot[{ReferencePoint},
  PlotStyle → {Red, PointSize[0.025]}], Frame → True]
```



A function that converts the simulated points to the CDF is defined. The number of simulation trials is n.

```
DataCDF[data_, R_] :=
 Rest@FoldList[{#2, #1[[2]] + 1/Length[data]} &, {0, 0},
   Sort[Norm[# - R] & /@ data, Less]]
(*
   Convert the points to vectors from the reference point.
   Find the lengths of the vectors.
   Sort the lengths in ascending order.
   Construct pairs
 {length, position in the list scaled between 0 and 1.
   Delete Foldlist artifact.
  *)
```

The closed form of the CDF is displayed as calculated by the algorithm.

```
(* Output the CDF after forcing the evaluation of
 expressions of the form Root[] to tidy up the output. *)
FR[r_, P_, R_] :=
 Quiet@
  Chop[PolygonCDF[P, R, r, True] /.
    x_ :> N[x] /; Head[x] == Root, 10^-5]

FREx[r_] = FR[r, Poly, ReferencePoint]
```

$$\begin{cases} 0 & 0 \le r \le \frac{1}{2\sqrt{2}} \\ \frac{1}{3}\left(-\sqrt{-1+8r^2}+8r^2\,\text{ArcSec}\left[2\sqrt{2}\,r\right]\right) & \frac{1}{2\sqrt{2}} \le r \le \frac{1}{2} \\ \frac{1}{3}\left(-1+\sqrt{-1+4r^2}+2r^2\left(\text{ArcCos}\left[\frac{3}{5}\right]-2\,\text{ArcCos}\left[\frac{2}{\sqrt{5}}\right]+2\,\text{ArcCsc}[2r]\right)\right) & \frac{1}{2} \le r \le 1 \\ \frac{1}{3}\left(-1+4\sqrt{-1+r^2}+\sqrt{-1+4r^2}+2r^2\left(\text{ArcCos}\left[\frac{3}{5}\right]-2\,\text{ArcCos}\left[\frac{2}{\sqrt{5}}\right]+2\,\text{ArcCsc}[2r]-2\,\text{ArcSec}[r]\right)\right) & 1 \le r \le \frac{\sqrt{5}}{2} \\ 1 & \frac{\sqrt{5}}{2} \le r \le \infty \\ 0 & \text{True} \end{cases}$$

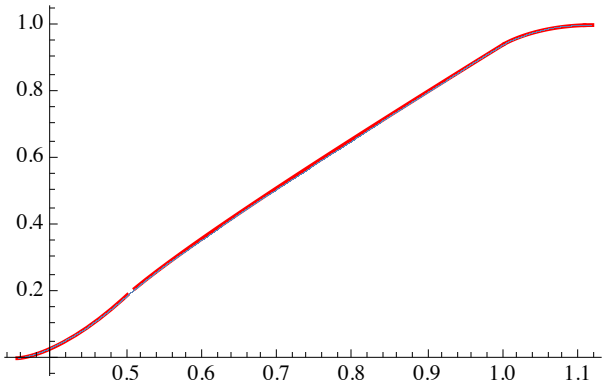
Also find the corresponding closed form of the PDF, which is needed for the i^{th} neighbor PDF.

```
fR[r_, P_, R_] :=
 Quiet@
  Chop[PolygonCDF[P, R, r, False] /.
    x_ :> N[x] /; Head[x] == Root, 10^-5]

fREx[r_] = fR[r, Poly, ReferencePoint]
```

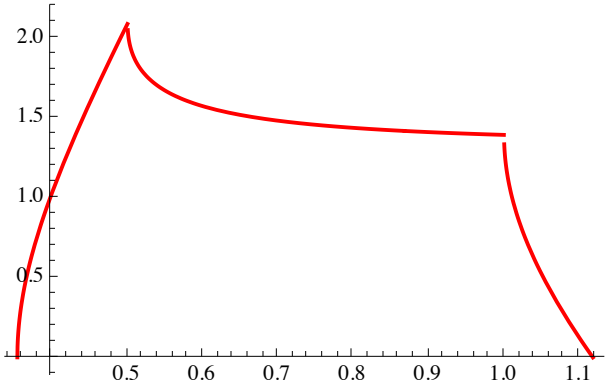
$$\begin{cases} 0 & 0 \le r \le \frac{1}{2\sqrt{2}} \\ \frac{1}{3}\left(\frac{2\sqrt{2}}{\sqrt{1-\frac{1}{8r^2}}} - \frac{8r}{\sqrt{-1+8r^2}} + 16r\,\text{ArcSec}\left[2\sqrt{2}\,r\right]\right) & \frac{1}{2\sqrt{2}} \le r \le \frac{1}{2} \\ \frac{1}{3}\left(-\frac{2}{\sqrt{1-\frac{1}{4r^2}}} + \frac{4r}{\sqrt{-1+4r^2}} + 4r\left(\text{ArcCos}\left[\frac{3}{5}\right] - 2\,\text{ArcCos}\left[\frac{2}{\sqrt{5}}\right] + 2\,\text{ArcCsc}[2r]\right)\right) & \frac{1}{2} \le r \le 1 \\ \frac{1}{3}\left(2\left(-\frac{2}{\sqrt{1-\frac{1}{r^2}}\,r^2} - \frac{1}{\sqrt{1-\frac{1}{4r^2}}\,r^2}\right)r^2 + \frac{4r}{\sqrt{-1+r^2}} + \frac{4r}{\sqrt{-1+4r^2}} + 4r\left(\text{ArcCos}\left[\frac{3}{5}\right] - 2\,\text{ArcCos}\left[\frac{2}{\sqrt{5}}\right] + 2\,\text{ArcCsc}[2r] - 2\,\text{ArcSec}[r]\right)\right) & 1 \le r \le \frac{\sqrt{5}}{2} \\ 0 & \text{True} \end{cases}$$

The analytical result for the CDF (red) is compared with the result obtained from simulations (blue).

```
minEx = If[
    FREx[r][[1, 1, 1]] === 0,
    FREx[r][[1, 1, 2, 3]],
    FREx[r][[1, 1, 2, 1]]
   ];
maxEx = FREx[r][[1, -1, 2, 1]];

Show[
 Plot[FREx[r], {r, minEx, maxEx}, PlotStyle → Red],
 ListPlot[DataCDF[data, ReferencePoint],
  PlotStyle → {PointSize[0.0001], AspectRatio → Automatic,
    Frame → True, FrameLabel → {r, "CDF"}}]
]
```




```
Plot[fREx[r], {r, minEx, maxEx}, PlotStyle → Red]
```



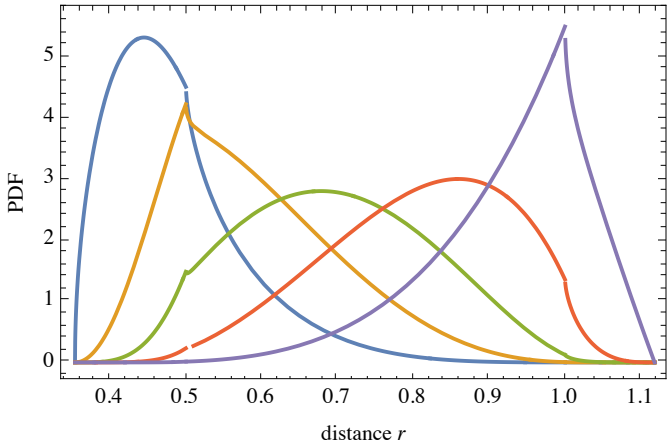
Using the CDF, the PDF of the Euclidean distance between an arbitrary reference point and its i^{th} neighbor node can be found.

Define the i^{th} neighbor PDF: equation (12) in [4].

$$\texttt{fi[m\_, i\_, r\_, F\_, f\_] := } \frac{\texttt{(1 - F[r])}^{\texttt{m-i}}\ \texttt{F[r]}^{\texttt{i-1}}}{\texttt{Beta[m - i + 1, i]}}\ \texttt{f[r]}$$

The result can be plotted as follows.

```
With[{m = 5 (* the number of nodes *)},
 Plot[Evaluate@Table[fi[m, i, r, FREx, fREx],
    {i , Range[1, m]}], {r, minEx, maxEx}, PlotRange → Full,
  Frame → True,
  FrameLabel → {Row[{"distance ", Style["r", Italic]}], PDF}]
]
```



6. Conclusion

In this article, we have reported on the use of *Mathematica* for distance distribution modeling in wireless networks. We have proposed and implemented an algorithm to compute the exact cumulative density function (CDF) of the distance from an arbitrary reference point to a randomly located node within an arbitrarily shaped (convex or concave) polygon. Examples of how the obtained distance distributions can be used in the modeling of finite-area wireless networks are provided in [6], [15–19]. The distance distribution results can also be applied in other branches of science, such as forestry, mathematics, operations research, and material sciences [13], [20].

Acknowledgments

We would like to thank the anonymous reviewer for his comments, especially for suggesting an efficient method of uniformly distributing points in a triangle. We would also like to thank Ms. Maryam Ahmadi (University of Victoria, Canada) and Prof. Jianping Pan (University of Victoria, Canada) for their constructive feedback, which helped us to further improve the implementation.

References

[1] M. Ahmadi and J. Pan, "Random Distances Associated with Arbitrary Triangles: A Recursive Approach with an Arbitrary Reference Point," UVic-PanLab-TR-14-01, 2014. dspace.library.uvic.ca//handle/1828/5134.

[2] J. G. Andrews, R. K. Ganti, M. Haenggi, N. Jindal, and S. Weber, "A Primer on Spatial Modeling and Analysis in Wireless Networks," *IEEE Communications Magazine*, **48**(11), 2010 pp. 156–163. doi:10.1109/MCOM.2010.5621983.

[3] Z. Khalid and S. Durrani, "Distance Distributions in Regular Polygons," *IEEE Transactions on Vehicular Technology*, **62**(5), 2013 pp. 2363–2368. doi:10.1109/TVT.2013.2241092.

[4] S. Srinivasa and M. Haenggi, "Distance Distributions in Finite Uniformly Random Networks: Theory and Applications," *IEEE Transactions on Vehicular Technology*, **59**(2), 2010 pp. 940–949. doi:10.1109/TVT.2009.2035044.

[5] S. Durrani, J. Guo, and Z. Khalid, "*Mathematica* and Matlab Software for Computing Distance Distributions," (May 12, 2015). users.cecs.anu.edu.au/~Salman.Durrani/software.html.

[6] J. Guo, S. Durrani, and X. Zhou, "Outage Probability in Arbitrarily-Shaped Finite Wireless Networks," *IEEE Transactions on Communications*, **62**(2), 2014 pp. 699–712. doi:10.1109/TCOMM.2013.122913.130298.

[7] Wikipedia. "Shoelace Formula." (Jun 18, 2015) en.wikipedia.org/wiki/Shoelace_formula.

[8] E. W. Weisstein. "Triangle Area" from Wolfram *MathWorld*—A Wolfram Web Resource. mathworld.wolfram.com/TriangleArea.html.

[9] E. W. Weisstein. "Polygon Area" from Wolfram *MathWorld*—A Wolfram Web Resource. mathworld.wolfram.com/PolygonArea.html.

[10] J. O'Rourke, *Art Gallery Theorems and Algorithms*, New York: Oxford University Press, 1987.

[11] E. W. Weisstein. "Triangle Point Picking" from Wolfram *MathWorld*—A Wolfram Web Resource. mathworld.wolfram.com/TrianglePointPicking.html.

[12] G. Turk, "Generating Random Points in Triangles," in *Graphics Gems* (A. S. Glassner, ed.), Boston: Academic Press, 1990 pp. 24–28. inis.jinr.ru/sl/vol1/CMC/Graphics_Gems_1,ed_A.Glassner.pdf.

[13] A. M. Mathai, *An Introduction to Geometrical Probability: Distributional Aspects with Applications*, Amsterdam: Gordon and Breach Science Publishers, 1999.

[14] J. Guo, S. Durrani, and X. Zhou, "Performance Analysis of Arbitrarily-Shaped Underlay Cognitive Networks: Effect of Secondary User Activity Protocols," *IEEE Transactions on Communications*, **63**(2), 2015 pp. 376–389. doi:10.1109/TCOMM.2014.2385065.

[15] J. Guo, S. Durrani, and X. Zhou, "Characterization of Aggregate Interference in Arbitrarily-Shaped Underlay Cognitive Networks," *Proceedings of IEEE Global Communications Conference (GLOBECOM)*, Austin, TX: IEEE, 2014. pp. 961–966. doi:10.1109/GLOCOM.2014.7036933.

[16] M. C. Valenti, D. Torrieri, and S. Talarico, "A Direct Approach to Computing Spatially Averaged Outage Probability," *IEEE Communications Letters*, **18**(7), 2014 pp. 1103–1106. doi:10.1109/LCOMM.2014.2317740.

[17] Z. Khalid, S. Durrani, and J. Guo, "A Tractable Framework for Exact Probability of Node Isolation and Minimum Node Degree Distribution in Finite Multi-hop Networks," *IEEE Transactions on Vehicular Technology*, **63**(6), 2014 pp. 2836–2847. doi:10.1109/TVT.2013.2293580.

[18] K. B. Baltzis, "Distance Distribution in Convex n-Gons: Mathematical Framework and Wireless Networking Applications," *Wireless Personal Communications*, **71**(2), 2012 pp. 1487–1503. doi:10.1007/s11277-012-0887-9.

[19] D. Moltchanov, "Distance Distribution in Random Networks," *Ad Hoc Networks*, **10**(6), 2012 pp. 1146–1166. doi:10.1016/j.adhoc.2012.02.005.

[20] U. Basel, "Random Chords and Point Distances in Regular Polygons," *Acta Mathematica Universitatis Comenianae*, **LXXXIII**(1), 2014 pp. 1–18. www.emis.de/journals/AMUC/_vol-83/_no_1/_baesel/baesel.html.

R. Pure and S. Durrani, "Computing Exact Closed-Form Distance Distributions in Arbitrarily Shaped Polygons with Arbitrary Reference Point," *The Mathematica Journal*, 2015. dx.doi.org/doi:10.3888/tmj.17-6.

About the Authors

Ross Pure is a second-year Bachelor of Engineering (Research & Development)/Bachelor of Science student in the Research School of Engineering, the Australian National University (ANU), Canberra, Australia.

Salman Durrani (SMIEEE, SFHEA, MIEAust) is a Senior Lecturer in the Research School of Engineering, ANU. He has coauthored more than 80 publications to date in refereed international journals and conferences. He currently serves as an editor of *IEEE Transactions on Communications*. His research interests include wireless and mobile communications, wireless energy harvesting, and synchronization and signal processing on the unit sphere.

Ross Pure
Research School of Engineering
The Australian National University
Canberra, Australia
u5349749@anu.edu.au

Salman Durrani
Research School of Engineering
The Australian National University
Canberra, Australia
salman.durrani@anu.edu.au