International Journal of Computational Geometry & Applications

Algorithms and a library for the exact computation of the cumulative distribution function of the Euclidean distance between a point and a random variable uniformly distributed in disks, balls, or polygones and application to Probabilistic Seismic Hazard Analysis

Vincent Guigues

School of Applied Mathematics, FGV
Praia de Botafogo, Rio de Janeiro, Brazil
vincent.guigues@fgv.br

Received (received date)
Revised (revised date)
Communicated by (Name)

ABSTRACT

We consider a random variable expressed as the Euclidean distance between an arbitrary point and a random variable uniformly distributed in a closed and bounded set of a three-dimensional Euclidean space. Four cases are considered for this set: a union of disjoint disks, a union of disjoint balls, a union of disjoint line segments, and the boundary of a polyhedron. In the first three cases, we provide closed-form expressions of the cumulative distribution function and the density. In the last case, we propose two algorithms with complexity $O(n \ln n)$, n being the number of edges of the polyhedron, that computes exactly the cumulative distribution function. An application of these results to probabilistic seismic hazard analysis and extensions are discussed. Finally, we present an open source library, available at `https://github.com/vguigues/Areas_Library`, that implements the algorithms presented in this paper.

Keywords: Computational Geometry; Geometric Probability; Distance to a random variable; Uniform distribution; Green's theorem; PSHA.

MSC2010 subject classifications: 60D05, 65D99, 51N20, 65D30, 86A15.

1. Introduction

Consider a closed and bounded set $\mathcal{S} \subset \mathbb{R}^3$ and a random variable $X : \Omega \to \mathcal{S}$ uniformly distributed in $\mathcal{S}$. Given an arbitrary point $P \in \mathbb{R}^3$, we study the distribution of the Euclidean distance $D : \Omega \to \mathbb{R}_+$ between P and X defined by $D(\omega) = \|\overrightarrow{PX(\omega)}\|_2$ for any $\omega \in \Omega$.

Denoting respectively the density and the cumulative distribution function (CDF) of D by $f_D(\cdot)$ and $F_D(\cdot)$, we have $f_D(d) = F_D(d) = 0$ if $d < \min_{Q \in \mathcal{S}} \|\overrightarrow{PQ}\|_2$ while

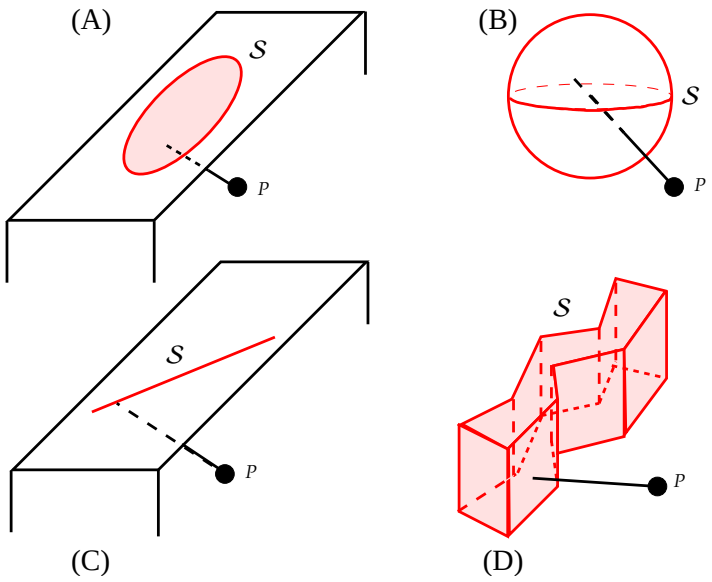

Fig. 1. Different supports $\mathcal{S}$ for random variable X.

$f_D(d) = 0$ and $F_D(d) = 1$ if $d > \max_{Q \in \mathcal{S}} \|\overrightarrow{PQ}\|_2$. For $\min_{Q \in \mathcal{S}} \|\overrightarrow{PQ}\|_2 \leq d \leq \max_{Q \in \mathcal{S}} \|\overrightarrow{PQ}\|_2$, we have

$$F_D(d) = \mathbb{P}(D \leq d) = \frac{\mu(\mathcal{B}(P,d) \cap \mathcal{S})}{\mu(\mathcal{S})}$$

where $\mu(A)$ is the Lebesgue measure of the set A and $\mathcal{B}(P,d)$ is the ball of center P and radius d. As a result, the computation of the CDF of D amounts to a problem of computational geometry, namely computing the Lebesgue measures of $\mathcal{S}$ and of $\mathcal{B}(P,d) \cap \mathcal{S}$ for any $d \in \mathbb{R}_+$.

We consider four cases for $\mathcal{S}$, represented in Figure 1 and denoted by (A), (B), (C), and (D) in this figure: (A) a disk, (B) a ball, (C) a line segment, and (D) the boundary of a polyhedron. The cases where $\mathcal{S}$ is a union of disks, a union of balls, or a union of line segments are straightforward extensions of cases (A), (B), and (C).

The study of these four cases is useful for Probabilistic Seismic Hazard Analysis (PSHA) to obtain the distribution of the distance between a given location on earth and the epicenter of an earthquake which, in a given seismic zone, is usually assumed to have a uniform distribution in that zone modelled as a union of disks, a union of balls, a union of line segments, or the boundary of a polyhedron in $\mathbb{R}^3$. This application, which motivated this study, is described in Section 2 following the lines of the seminal papers Cornell [1968], McGuire [1976], which paved the way for PSHA. PSHA involves several approximations and models and there-

fore, as in Guibas et al. [1989], Kostitsyna et al. [2015], Löffler [2010], Myers and Joskowicz [2008, 2009], Ostrovsky-Berman and Joskowicz [2005], Stewart [1991], our algorithms perform geometric computations over inexact inputs.

In this context, the outline of the paper is as follows. In Section 2, we provide an overview on PSHA. In Section 3, we consider case (A), the case where $\mathcal{S}$ is a disk. In Section 4 and Subsection 5.1, we consider respectively case (B), where $\mathcal{S}$ is a ball, and case (C), where $\mathcal{S}$ is a line segment. In these three cases (A), (B), and (C), we obtain closed-form expressions for the CDF and the density of D. The main mathematical contributions of this paper are Sections 5.2 and 5.3 which provide for case (D), i.e., the case where $\mathcal{S}$ is the boundary of a polyhedron, two algorithms with complexity $O(n \ln n)$ where n is the number of edges of the polyhedron, that computes exactly the CDF of D. An approximate density for D can then be obtained. We are not aware of other papers with these results. However, particular cases have been discussed: in Baker [2008], cases (A) and (C) are considered taking for P respectively the center of the disk and a point on the perpendicular bisector of the line segment. In Stewart and Zhang [2013], as a particular case of (D), a rectangle is considered for $\mathcal{S}$ while P is the center of the rectangle. In the case where $\mathcal{S}$ is the boundary of a polyhedron, to our knowledge, the current versions of the most popular softwares for PSHA (OPENQUACK, CRISIS 2012 Ordaz et al. [2012]) do not compute exactly the CDF of D. For instance, CRISIS 2012 uses an approximate algorithm that performs a spatial integration subdividing the boundary of the polyhedron into small triangles. We also implemented a Matlab library available at `https://github.com/vguigues/Areas_Library` with the algorithms presented in this paper. This library as well as numerical experiments are presented in Section 6 while extensions of our results, in particular to handle the case of a general polyhedron and the case where the ℓ_2-norm is replaced by either the ℓ_1-norm or the ℓ_∞-norm, are discussed in the last Section 7.

Throughout the paper, we use the following notation. For a point A in $\mathbb{R}^3$, we denote its coordinates with respect to a given Cartesian coordinate system by x_A, y_A, and z_A. For two points $A, B \in \mathbb{R}^3$, $\overline{AB}$ is the line segment joining points A and B, i.e., $\overline{AB} = \{tA + (1-t)B \ : \ t \in [0,1]\}$, $(AB) = \{tA + (1-t)B \ : \ t \in \mathbb{R}\}$ is the line passing through A and B, and $\overrightarrow{AB}$ is the vector whose coordinates are $(x_B - x_A, y_B - y_A, z_B - z_A)$. Given two vectors $x, y \in \mathbb{R}^3$, we denote the usual scalar product of x and y in $\mathbb{R}^3$ by $\langle x, y \rangle = x^T y$. For $P \in \mathbb{R}^2$, we denote the circle and the disk of center P and radius d by respectively $\mathcal{C}(P, d)$ and $\mathcal{D}(P, d)$.

2. Overview of the four steps of PSHA

An important problem in civil engineering is to determine the level of ground shaking a given structure can withstand. In regions with high levels of seismic activity, it makes sense to invest in structures able to resist high levels of ground shaking. On the contrary, in regions without seismic activity during the structure lifetime, we should not invest in such structures. More precisely, it would be reasonable to

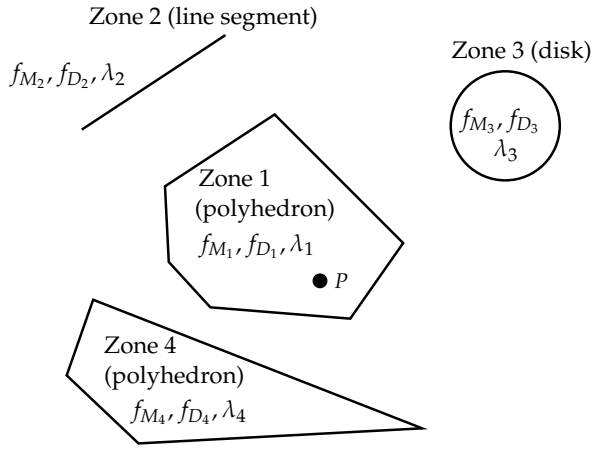

Fig. 2. Seismic zones around a given point P.

design structures able to resist up to a Peak Ground Acceleration $A^* m.s^{-2}$ that is very rarely exceeded, say with a small probability ε, over a given time window. This approach is used in PSHA: the confidence level ε and the time window being fixed (say of t years), the main task of PSHA is to estimate at a given location P, the Peak Ground Acceleration (PGA) A^* such that the probability of the event

$$E_t(A^*, P) = \{\text{There is at least an earthquake causing a PGA greater than } A^* \text{ at } P \text{ in the next } t \text{ years}\} \tag{1}$$

is ε. We present the approach introduced by Cornell [1968], McGuire [1976], to model and solve this problem. In this approach, we consider the seismic zones that could have an impact on the PGA at P (see Figure 2 for an example of 4 zones with P belonging to one of these zones). These zones are bounded sets that do not overlap: typically disks, line segments, or simple polygones. The number of earthquakes provoking PGAs at P greater than A^* over the next t years depends on the frequency of earthquakes in each zone. As for the ground acceleration at P provoked by the earthquakes of a given zone, it will depend on the magnitudes of these earthquakes, which are random, and the locations of their epicenters, which are random too. To take these factors into account, PSHA uses a four-step process (see Figure 2):

(i) in zone i, the process of earthquake arrivals is modelled as a Poisson process with rate λ_i. We will assume that the earthquake arrival processes in the different zones are independent.
(ii) In zone i, the magnitude of earthquakes is modelled as a random variable M_i with density $f_{M_i}(\cdot)$.
(iii) The distance between P and the epicenter of the earthquakes of zone i is modelled as a random variable D_i with density $f_{D_i}(\cdot)$.

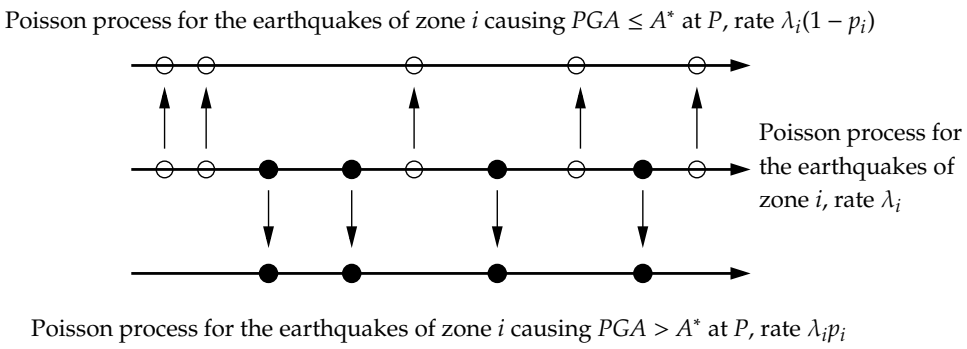

Fig. 3. Splitting of the process of earthquake arrivals in zone i into a process of earthquakes causing $PGA > A^*$ at P (arrivals represented by black balls) and a process of earthquakes causing $PGA \leq A^*$ at P.

(iv) A ground motion prediction model is chosen expressed as a regression of the ground acceleration on magnitude, distance, and possibly other factors.

We now detail these steps and explain how to combine them to achieve the main task of PSHA: compute the probability of event (1) for any A^*. The ability to compute this probability for any A^* makes possible the estimation, by dichotomy, of an acceleration A^* satisfying $\mathbb{P}\Big(E_t(A^*, P)\Big) = \varepsilon$.

From (i), we obtain that the distribution of the number of earthquakes N_{ti} in zone i on a time window of t time units is given by

$$\mathbb{P}(N_{ti} = k) = e^{-\lambda_i t}\frac{(\lambda_i t)^k}{k!},\ k \in \mathbb{N},$$

where the rate λ_i represents the mean number of earthquakes in zone i per time unit, say per year. From now on, we fix an acceleration A^* and introduce the event

$$E(A^*, P, i) = \{\text{An earthquake from zone } i \text{ causes a PGA greater than } A^* \text{ at } P\} \tag{2}$$

with its probability $p_i = \mathbb{P}\Big(E(A^*, P, i)\Big)$. For each earthquake in zone i, either event $E(A^*, P, i)$ occurs for this earthquake, i.e., this earthquake causes a PGA greater than A^* at P, or not. As a result, we can define two new counting processes for zone i: the process $\tilde{N}_{ti}$ counting the earthquakes causing $PGA > A^*$ at P (events represented by black balls in Figure 3) and the process counting the earthquakes causing $PGA \leq A^*$ at P. To proceed, we need the following well-known lemma:

Lemma 1. *Consider a Poisson process N_t with arrival rate λ. Assume that arrivals are of two types I and II: type I with probability p and type II with probability $1-p$. We also assume that the arrival types are independent. Then the process $\tilde{N}_t$ of type I arrivals is a Poisson process with rate λp.*

Proof. We compute for every $k \in \mathbb{N}$,

$$\begin{aligned}\mathbb{P}\big(\tilde{N}_t = k\big) &= \sum_{j=k}^{+\infty} \mathbb{P}\big(\tilde{N}_t = k | N_t = j\big)\mathbb{P}\big(N_t = j\big) \quad \text{[Total Probability Theorem]} \\ &= \sum_{j=k}^{+\infty} C_j^k p^k (1-p)^{j-k} e^{-\lambda t} \frac{(\lambda t)^j}{j!} \\ &= e^{-\lambda t}\frac{(\lambda p t)^k}{k!} \sum_{j=0}^{+\infty} \frac{\big[\lambda(1-p)t\big]^j}{j!} = e^{-\lambda p t}\frac{(\lambda p t)^k}{k!},\end{aligned}$$

which shows that $\tilde{N}_t$ is a Poisson random variable with parameter $\lambda p t$. We conclude using the independence of the arrival types on disjoint time windows. □

This lemma shows that the process $(\tilde{N}_{ti})_t$ is a Poisson process with rate $\lambda_i p_i$. Denoting by $\mathcal{N}$ the number of zones, it follows that the probability to have k earthquakes causing a PGA greater than A^* at P over the next time window of t years is

$$\begin{aligned}\mathbb{P}\Big(\sum_{i=1}^{\mathcal{N}} \tilde{N}_{ti} = k\Big) &= \sum_{x_1+\ldots+x_{\mathcal{N}}=k} \mathbb{P}\big(\tilde{N}_{t1} = x_1; \ldots; \tilde{N}_{t\mathcal{N}} = x_{\mathcal{N}}\big) \\ &= \sum_{x_1+\ldots+x_{\mathcal{N}}=k} \prod_{i=1}^{\mathcal{N}} \mathbb{P}\big(\tilde{N}_{ti} = x_i\big) \\ &= \sum_{x_1+\ldots+x_{\mathcal{N}}=k} \prod_{i=1}^{\mathcal{N}} e^{-\lambda_i p_i t}\frac{(\lambda_i p_i t)^{x_i}}{x_i!}\end{aligned}$$

where for the second equality we have used the independence of $\tilde{N}_{t1}, \ldots, \tilde{N}_{t\mathcal{N}}$. Taking $k = 0$ in the above relation, we obtain

$$1 - \mathbb{P}(E_t(A^*, P)) = \mathbb{P}(\overline{E_t(A^*, P)}) = e^{-(\sum_{i=1}^{\mathcal{N}} \lambda_i p_i)t}. \tag{3}$$

Setting $\tilde{N}_t = \sum_{i=1}^{\mathcal{N}} \tilde{N}_{ti}$, the expectation of $\tilde{N}_t$ which is the mean number of earthquakes causing a PGA greater than A^* at P over the next t years, can be expressed as

$$\lambda_t(A^*, P) = \mathbb{E}\big[\tilde{N}_t\big] = \sum_{i=1}^{\mathcal{N}} \mathbb{E}\big[\tilde{N}_{ti}\big] = (\sum_{i=1}^{\mathcal{N}} \lambda_i p_i)t. \tag{4}$$

Using this relation and (3), the probability of event $E_t(A^*, P)$ can be rewritten

$$\mathbb{P}(E_t(A^*, P)) = 1 - e^{-\lambda_t(A^*, P)}$$

with $\lambda_t(A^*, P)$ given by (4).

It remains to explain how the probability p_i of event (2) is computed. This computation is based on a ground motion prediction model (step (iv) above) which is a

regression equation representing the PGA induced by an earthquake of magnitude M at distance D of its epicenter. This relation takes the form

$$\ln PGA = \overline{\ln PGA}(M, D, \theta) + \sigma(M, D, \theta)\varepsilon. \tag{5}$$

In this relation, $\overline{\ln PGA}(M, D, \theta)$ (resp. $\sigma(M, D, \theta)$) is the conditional mean (resp. standard deviation) of $\ln PGA$ given the magnitude M and distance D to the epicenter while ε is a standard Gaussian random variable. We see that the PGA depends on the magnitude, the distance to the epicenter and other parameters, generally referred to as θ (such as the ground conditions). More precisely, the mean $\overline{\ln PGA}(M, D, \theta)$ should increase with M (the higher the magnitude, the higher the PGA) and decrease with D (the larger the distance, the lower the PGA). As an example, the ground motion prediction model in Cornell [1968] is of the form

$$\ln PGA = 0.152 + 0.859M - 1.803 \ln(D + 25) + 0.57\varepsilon$$

which amounts to take $\overline{\ln PGA}(M, D, \theta) = 0.152 + 0.859M - 1.803 \ln(D + 25)$ and $\sigma(M, D, \theta) = 0.57$.

The density $f_{M_i}(\cdot)$ used for the distribution of the magnitude of the earthquakes of zone i depends on the history of the magnitudes of the earthquakes of that zone. For a large number of seismic zones, the density proposed by Gutenberg and Richter [1944] has shown appropriate. It is of the form

$$f_{M_i}(m) = \frac{\beta_i e^{-\beta_i(m - M_{\min}(i))}}{1 - e^{-\beta_i(M_{\max}(i) - M_{\min}(i))}}$$

for some parameter $\beta_i > 0$ where the support of M_i is $[M_{\min}(i), M_{\max}(i)]$.

In each zone, the epicenter has a uniform distribution in that zone. The seismic zones usually considered in PSHA are disks, balls, line segments, or the boundary of a polyhedron. As a result, the determination of the density $f_{D_i}(\cdot)$ of the distance D_i between P and the epicenter in zone i can be determined analytically or approximately using Sections 3, 4, 5.1, and 5.2.

Gathering the previous ingredients, assuming that D_i and M_i are independent, and using the Total Probability Theorem, we obtain

$$p_i = \int_{m_i = M_{\min}(i)}^{M_{\max}(i)} \int_{x_i = 0}^{\infty} \mathbb{P}\Big(PGA > A^* | M_i = m_i; D_i = x_i\Big) f_{M_i}(m_i) f_{D_i}(x_i) dm_i dx_i$$

where $\mathbb{P}\Big(PGA > A^* | M_i = m_i; D_i = x_i\Big)$ is given by the ground motion prediction model (5). For implementation purposes, the above integral is generally estimated discretizing the continuous distributions of magnitude $M_i, i = 1, \ldots, \mathcal{N}$, and distance $D_i, i = 1, \ldots, \mathcal{N}$.

Finally, we mention the existence of an alternative, zoneless approach to PSHA introduced by Frankel [1995] and Woo [1996].

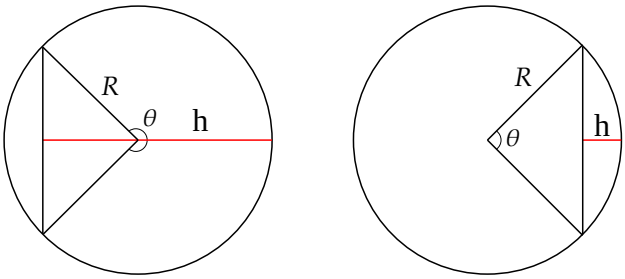

Fig. 4. Lenses of height h in a disk of radius R.

3. Distance to a random variable uniformly distributed in a disk

Let $\mathcal{S} = \mathcal{D}(S_0, R_0)$ be a disk of center S_0 and radius $R_0 > 0$ and let P be a point in the plane containing $\mathcal{S}$ at Euclidean distance R_1 of S_0. We first consider the case where $R_1 = 0$. If $0 \leq d \leq R_0$, we get $F_D(d) = \frac{\pi d^2}{\pi R_0^2} = (d/R_0)^2$ and $f_D(d) = 2\frac{d}{R_0^2}$, if $d > R_0$ we have $F_D(d) = 1$ and $f_D(d) = 0$ while if $d < 0$ we have $F_D(d) = f_D(d) = 0$. Let us now consider the case where $R_1 \geq R_0$. If $d > R_1 + R_0$ we have $F_D(d) = 1$ and $f_D(d) = 0$ while if $d < R_1 - R_0$ we have $F_D(d) = f_D(d) = 0$. Let us now take $R_1 - R_0 \leq d \leq R_1 + R_0$. The intersection of the disks $\mathcal{D}(S, R_0)$ and $\mathcal{D}(P, d)$ is the union of two lenses having a line segment $\overline{AB}$ in common (see Figures 4 and 5). Without loss of generality, assume that (S_0P) is the x-axis and that the equations of the boundaries of the disks are given by $x^2 + y^2 = R_0^2$ and $(x - R_1)^2 + y^2 = d^2$. From these equations, we obtain that the abscissa of the intersection points A and B of the boundaries of the disks is $x^* = \frac{R_0^2 + R_1^2 - d^2}{2R_1}$. Note that $A = B$ if and only if $d = R_1 \pm R_0$. In Figure 5, we represented a situation where $x^* \geq 0$ and a situation where $x^* < 0$. In both cases, $\mathcal{D}(S_0, R_0) \cap \mathcal{D}(P, d)$ is the union of a lens of height $h_1(d)$ in a disk of radius d (the disk $\mathcal{D}(P, d)$) and of a lens of height $h_2(d)$ in a disk of radius R_0 (the disk $\mathcal{D}(S_0, R_0)$) where

$$\begin{aligned} h_1(d) &= d - R_1 + x^* = d - R_1 + \tfrac{R_0^2 + R_1^2 - d^2}{2R_1} \text{ and} \\ h_2(d) &= R_0 - x^* = R_0 - \tfrac{R_0^2 + R_1^2 - d^2}{2R_1}. \end{aligned} \tag{6}$$

Recall that the area $\mathbb{A}(R, h)$ of a lens of height h contained in a disk of radius R (see Figure 4) is $\mathbb{A}(R, h) = R^2\frac{\theta}{2} - R^2\sin(\frac{\theta}{2})\cos(\frac{\theta}{2})$ with $\cos(\frac{\theta}{2}) = \frac{R-h}{R}$, i.e.,

$$\mathbb{A}(R, h) = R^2 \text{Arccos}\left(\frac{R-h}{R}\right) - (R-h)\sqrt{R^2 - (R-h)^2}. \tag{7}$$

In the sequel, we will denote by $\mathcal{A}(\mathbb{S})$ the area of a surface $\mathbb{S}$. With this notation, it follows that

$$\mathcal{A}(\mathcal{D}(S_0, R_0) \cap \mathcal{D}(P, d)) = \mathbb{A}(d, h_1(d)) + \mathbb{A}(R_0, h_2(d)) \tag{8}$$

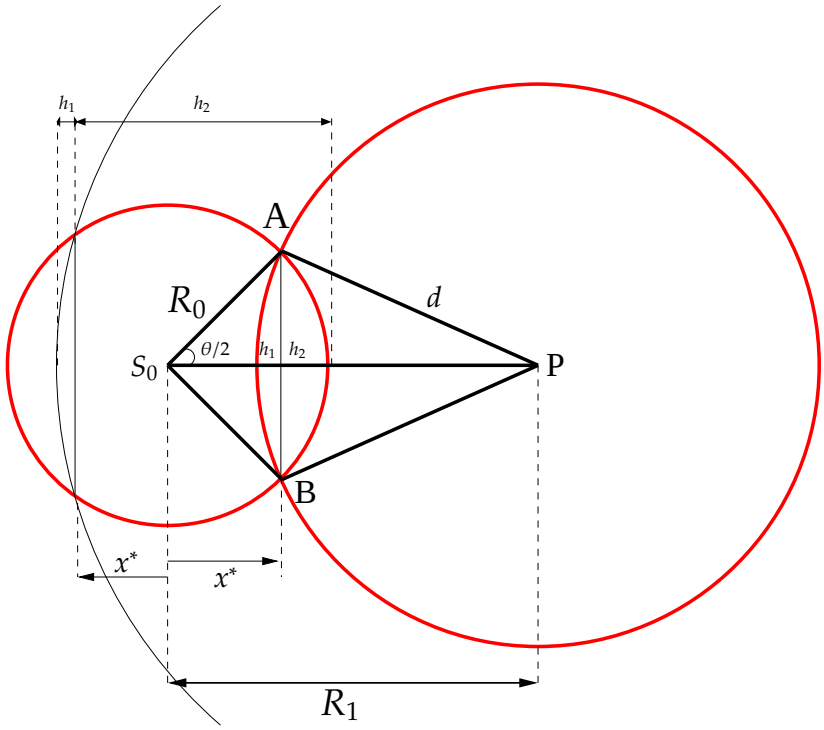

Fig. 5. Random variable X uniformly distributed in a ball of radius R_0 and center S_0. Case where $R_1 \geq R_0 > 0$.

where

$$\mathbb{A}(d, h_1(d)) = d^2 \text{Arccos}\left(\frac{d^2 + R_1^2 - R_0^2}{2R_1 d}\right) - \frac{d^2 + R_1^2 - R_0^2}{2R_1}\sqrt{d^2 - \left(\frac{d^2 + R_1^2 - R_0^2}{2R_1}\right)^2} \quad (9)$$

and

$$\mathbb{A}(R_0, h_2(d)) = R_0^2 \text{Arccos}\left(\frac{R_0^2 + R_1^2 - d^2}{2R_0 R_1}\right) - \frac{R_0^2 + R_1^2 - d^2}{2R_1}\sqrt{R_0^2 - \left(\frac{R_0^2 + R_1^2 - d^2}{2R_1}\right)^2}. \quad (10)$$

For $R_1 - R_0 \leq d \leq R_1 + R_0$, we obtain $F_D(d) = \frac{\mathbb{A}(d,h_1(d)) + \mathbb{A}(R_0,h_2(d))}{\pi R_0^2}$ where $\mathbb{A}(d, h_1(d))$ and $\mathbb{A}(R_0, h_2(d))$ are given by (9) and (10). The density is

$$f_D(d) = \frac{1}{\pi R_0^2}\Big[h_2'(d)\frac{\partial \mathbb{A}(R_0, h_2(d))}{\partial h} + \frac{\partial \mathbb{A}(d, h_1(d))}{\partial R} + h_1'(d)\frac{\partial \mathbb{A}(d, h_1(d))}{\partial h}\Big] \quad (11)$$

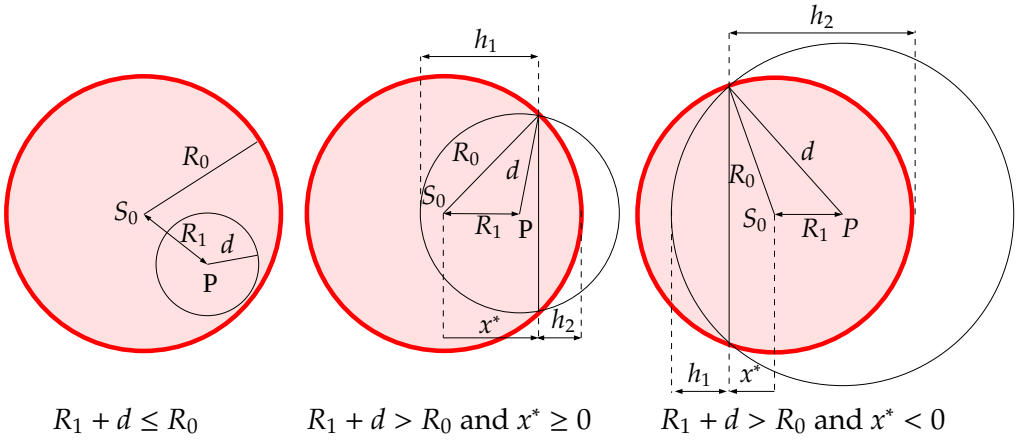

Fig. 6. Random variable X uniformly distributed in a ball of radius R_0 and center S_0. Case where $0 < R_1 < R_0$.

where $h_1'(d) = 1 - \frac{d}{R_1}$, $h_2'(d) = \frac{d}{R_1}$, and

$$\begin{aligned} \frac{\partial \mathbb{A}(R,h)}{\partial R} &= 2R\text{Arccos}\left(1 - \tfrac{h}{R}\right) - 2\sqrt{h(2R-h)}, \\ \frac{\partial \mathbb{A}(R,h)}{\partial h} &= 2\sqrt{h(2R-h)}. \end{aligned} \tag{12}$$

We now consider the case where $R_1 > 0$ and $R_1 < R_0$ (see Figure 6). If $0 \leq d \leq R_0 - R_1$, we obtain $F_D(d) = \frac{\pi d^2}{\pi R_0^2} = \frac{d^2}{R_0^2}$, if $d < 0$ we have $F_D(d) = f_D(d) = 0$ while if $d > R_0 + R_1$ we have $F_D(d) = 1$ and $f_D(d) = 0$ (see Figure 6). If $R_1 + R_0 \geq d > R_0 - R_1$, both in the case where the abscissa x^* of the intersection points between the boundaries of $\mathcal{D}(S_0, R_0)$ and $\mathcal{D}(P, d)$ is positive and negative, we check (see Figure 6) that the area of $\mathcal{D}(S_0, R_0) \cap \mathcal{D}(P, d)$ is still given by (8) with $\mathbb{A}(d, h_1(d))$ and $\mathbb{A}(R_0, h_2(d))$ given respectively by (9) and (10). Summarizing, if $0 < R_1 < R_0$ then if $R_1 + R_0 \geq d > R_0 - R_1$, the density of D at d is given by (11) and if $0 \leq d \leq R_0 - R_1$, we have $f_D(d) = \frac{2d}{R_0^2}$. The density of D when X is uniformly distributed in a disk is given for some examples in Figure 7.

Finally, we consider the case where $\mathcal{S}$ is a disk $\mathcal{D}$ and $P \in \mathbb{R}^3$ is not contained in the plane $\mathcal{P}$ containing this disk. Let S_0 be the center of $\mathcal{S}$ and let S_1, S_2 be two points of the boundary of the disk such that $\overrightarrow{S_0S_1}$ and $\overrightarrow{S_0S_2}$ are linearly independent. We introduce the projection $P_0 = \pi_{\mathcal{P}}[P] = \text{argmin}_{Q \in \mathcal{P}} \|\overrightarrow{PQ}\|_2$ of P onto $\mathcal{P}$. Since vectors $\overrightarrow{S_0S_1}$ and $\overrightarrow{S_0S_2}$ are linearly independent, if A is the $(3,2)$ matrix $[\overrightarrow{S_0S_1}, \overrightarrow{S_0S_2}]$ whose first column is $\overrightarrow{S_0S_1}$ and whose second column is $\overrightarrow{S_0S_2}$, then the matrix $A^\top A$ is invertible. It follows that the projection $P_0 = \pi_{\mathcal{P}}[P]$ of P onto $\mathcal{P}$ can be expressed as $\overrightarrow{S_0P_0} = A(A^\top A)^{-1}A^\top \overrightarrow{S_0P}$. With this notation, the intersection of $\mathcal{P}$ and the ball $\mathcal{B}(P, d)$

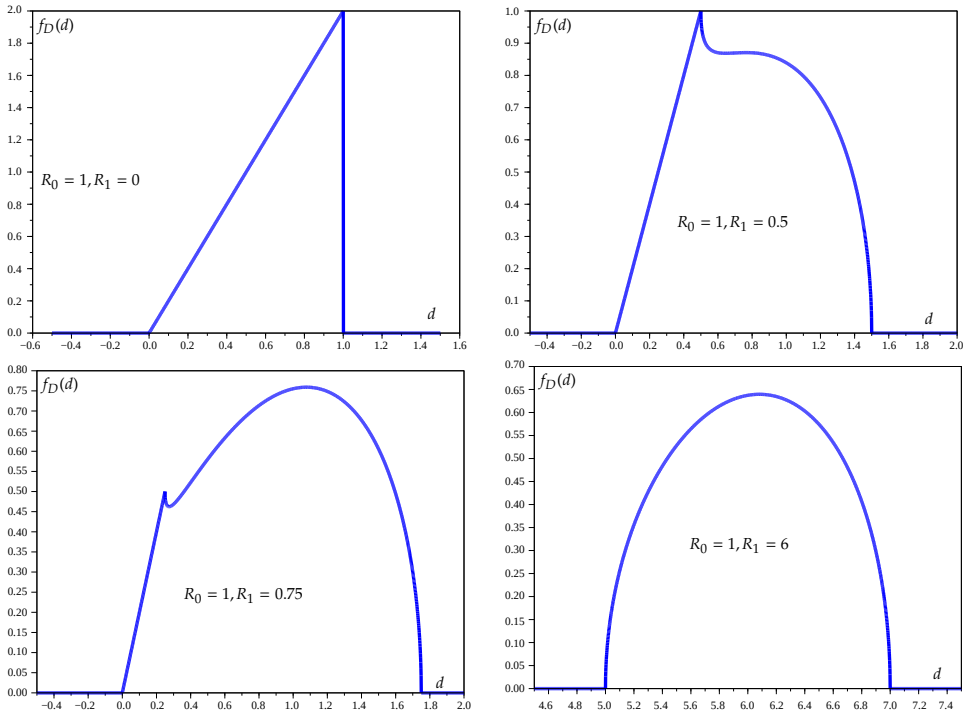

Fig. 7. Density of D where X is uniformly distributed in a disk of radius $R_0 = 1$: some examples. Top left: $R_1 = 0$, top right: $R_1 = 0.5$, bottom left: $R_1 = 0.75$, bottom right: $R_1 = 6$.

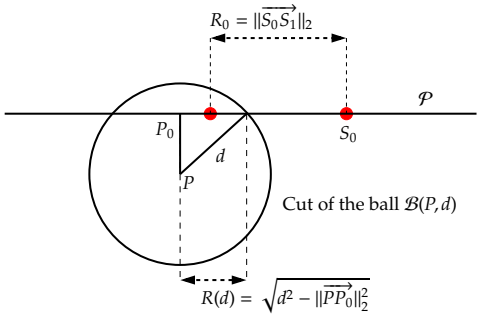

Fig. 8. Euclidean distance to a point uniformly distributed in a disk.

of center P and radius d is either empty or it is a disk of center P_0 and radius

$$R(d) = \sqrt{d^2 - \|\overrightarrow{PP_0}\|_2^2} \tag{13}$$

(see Figure 8). In the latter case, denoting this disk by $\mathcal{D}(P_0, R(d))$ and using the fact

that $\mathcal{D} = \mathcal{D} \cap \mathcal{P}$ (recall that $\mathcal{D} \subset \mathcal{P}$), we obtain

$$\mathcal{D} \cap \mathcal{B}(P, d) = \mathcal{D} \cap \mathcal{P} \cap \mathcal{B}(P, d) = \mathcal{D} \cap \mathcal{D}(P_0, R(d)).$$

Since $\mathcal{D}$ and $\mathcal{D}(P_0, R(d))$ are disks contained in the plane $\mathcal{P}$, setting $R_0 = \|\overrightarrow{S_0S_1}\|_2$ and $R_1 = \|\overrightarrow{S_0P_0}\|_2$, the previous results provide the area of their intersection and the following CDFs and densities for D:

Case where $P_0 = S_0$: The CDF and density of D are given by

$$\begin{cases} F_D(d) = f_D(d) = 0 & \text{if } d < \|\overrightarrow{S_0P}\|_2, \\ \left.\begin{cases} F_D(d) = \frac{R(d)^2}{\|\overrightarrow{S_0S_1}\|_2^2} = \frac{d^2 - \|\overrightarrow{S_0P}\|_2^2}{\|\overrightarrow{S_0S_1}\|_2^2} \\ f_D(d) = \frac{2d}{\|\overrightarrow{S_0S_1}\|_2^2} \end{cases}\right\} & \text{if } \|\overrightarrow{S_0P}\|_2 \le d \le \sqrt{\|\overrightarrow{S_0P}\|_2^2 + \|\overrightarrow{S_0S_1}\|_2^2}, \\ F_D(d) = 1 \text{ and } f_D(d) = 0 & \text{if } d > \sqrt{\|\overrightarrow{S_0P}\|_2^2 + \|\overrightarrow{S_0S_1}\|_2^2}. \end{cases}$$

Case where $0 < \|\overrightarrow{S_0P_0}\|_2 < \|\overrightarrow{S_0S_1}\|_2$: Setting

$$\begin{aligned} d_{\min} &= \sqrt{\|\overrightarrow{PP_0}\|_2^2 + (\|\overrightarrow{S_0S_1}\|_2 - \|\overrightarrow{S_0P_0}\|_2)^2} \text{ and} \\ d_{\max} &= \sqrt{\|\overrightarrow{PP_0}\|_2^2 + (\|\overrightarrow{S_0S_1}\|_2 + \|\overrightarrow{S_0P_0}\|_2)^2}, \end{aligned} \tag{14}$$

the CDF of D is given by

$$\begin{cases} (a)\ F_D(d) = 0 & \text{if } d < \|\overrightarrow{PP_0}\|_2, \\ (b)\ F_D(d) = \frac{R(d)^2}{\|\overrightarrow{S_0S_1}\|_2^2} = \frac{d^2 - \|\overrightarrow{PP_0}\|_2^2}{\|\overrightarrow{S_0S_1}\|_2^2} & \text{if } \|\overrightarrow{PP_0}\|_2 \le d \le d_{\min}, \\ (c)\ F_D(d) = \frac{\mathbb{A}(R(d), h_1(R(d))) + \mathbb{A}(\|\overrightarrow{S_0S_1}\|_2, h_2(R(d)))}{\pi \|\overrightarrow{S_0S_1}\|_2^2} & \text{if } d_{\min} \le d \le d_{\max}, \\ (d)\ F_D(d) = 1 & \text{if } d > d_{\max}, \end{cases} \tag{15}$$

where the expression of $\mathbb{A}$ is given by (7) and, where, using the expressions of h_1 and h_2 and recalling that $R_0 = \|\overrightarrow{S_0S_1}\|_2$ and $R_1 = \|\overrightarrow{S_0P_0}\|_2$,

$$\begin{aligned} h_1(R(d)) &= \sqrt{d^2 - \|\overrightarrow{PP_0}\|_2^2} - \|\overrightarrow{S_0P_0}\|_2 + \frac{\|\overrightarrow{S_0S_1}\|_2^2 + \|\overrightarrow{S_0P_0}\|_2^2 + \|\overrightarrow{PP_0}\|_2^2 - d^2}{2\|\overrightarrow{S_0P_0}\|_2}, \\ h_2(R(d)) &= \|\overrightarrow{S_0S_1}\|_2 - \frac{\|\overrightarrow{S_0S_1}\|_2^2 + \|\overrightarrow{S_0P_0}\|_2^2 + \|\overrightarrow{PP_0}\|_2^2 - d^2}{2\|\overrightarrow{S_0P_0}\|_2}. \end{aligned} \tag{16}$$

It follows that $f_D(d) = 0$ if $d < \|\overrightarrow{PP_0}\|_2$ or $d > d_{\max}$ while $f_D(d) = \frac{2d}{\|\overrightarrow{S_0S_1}\|_2^2}$ if $\|\overrightarrow{PP_0}\|_2 \le d \le d_{\min}$. Finally, if $d_{\min} \le d \le d_{\max}$, we have

$$\begin{aligned} f_D(d) = \frac{1}{\pi\|\overrightarrow{S_0S_1}\|_2^2} &\left[\frac{d}{\|\overrightarrow{S_0P_0}\|_2} \frac{\partial \mathbb{A}(\|\overrightarrow{S_0S_1}\|_2, h_2(R(d)))}{\partial h} + \frac{d}{\sqrt{d^2 - \|\overrightarrow{PP_0}\|_2^2}} \frac{\partial \mathbb{A}(R(d), h_1(R(d)))}{\partial R} \right] \\ &+ \frac{d}{\pi\|\overrightarrow{S_0S_1}\|_2^2} \left(\frac{1}{\sqrt{d^2 - \|\overrightarrow{PP_0}\|_2^2}} - \frac{1}{\|\overrightarrow{S_0P_0}\|_2} \right) \frac{\partial \mathbb{A}(R(d), h_1(R(d)))}{\partial h} \end{aligned} \tag{17}$$

where the expressions of $\frac{\partial \mathbb{A}(R,h)}{\partial R}$ and $\frac{\partial \mathbb{A}(R,h)}{\partial h}$ are given by (12).

Case where $\|\overrightarrow{S_0P_0}\|_2 \geq \|\overrightarrow{S_0S_1}\|_2$**:** With the definitions (14) of $d_{\min}$ and $d_{\max}$, if $d < d_{\min}$ then $F_D(d) = f_D(d) = 0$, if $d > d_{\max}$ then $F_D(d) = 1$ and $f_D(d) = 0$, while if $d_{\min} \leq d \leq d_{\max}$, $f_D(d)$ is given by (17) and $F_D(d)$ is given by (15)-(c) with $h_1(R(d))$ and $h_2(R(d))$ given by (16).

4. Distance to a random variable uniformly distributed in a ball

Let $\mathcal{S} = \mathcal{B}(S_0, R_0)$ be a ball of radius $R_0 > 0$ and center S_0 in $\mathbb{R}^3$ and let P be at Euclidean distance R_1 of S_0. The computations are identical to those of the previous section replacing two dimensional lenses and disks by three dimensional caps and balls. If $R_1 = 0$ then if $d > R_0$, we have $f_D(d) = 0$ and $F_D(d) = 1$, if $d < 0$, we have $f_D(d) = F_D(d) = 0$ while if $0 \leq d \leq R_0$, we obtain $F_D(d) = \frac{(4/3)\pi d^3}{(4/3)\pi R_0^3}$, i.e., $f_D(d) = 3\frac{d^2}{R_0^3}$ (see Figure 6). If $0 < R_1 < R_0$, then if $d > R_0 + R_1$, we have $F_D(d) = 1$ and $f_D(d) = 0$, if $d < 0$, we have $F_D(d) = f_D(d) = 0$ while if $0 \leq d \leq R_0 - R_1$, we have $F_D(d) = \frac{(4/3)\pi d^3}{(4/3)\pi R_0^3}$, i.e., $f_D(d) = 3\frac{d^2}{R_0^3}$ (see Figure 6). If $R_1 \geq R_0$ then if $d > R_0 + R_1$, we have $f_D(d) = 0$ and $F_D(d) = 1$ and if $d < R_1 - R_0$, we have $f_D(d) = F_D(d) = 0$. If $0 < R_1 < R_0$ and $R_0 - R_1 < d \leq R_0 + R_1$ or if $R_1 \geq R_0$ and $R_1 - R_0 \leq d \leq R_1 + R_0$, then $\mathcal{B}(S_0, R_0) \cap \mathcal{B}(P, d)$ is the union of a spherical cap of height $h_1(d)$ contained in a ball of radius d (the ball $\mathcal{B}(P, d)$) and of a spherical cap of height $h_2(d)$ contained in a ball of radius R_0 (the ball $\mathcal{B}(S_0, R_0)$) where the expressions (6) for $h_1(d)$ and $h_2(d)$ are still valid. Now recall that the volume of a spherical cap (see Figure 4 for a cut of this cap) of height h contained in a ball of radius R in $\mathbb{R}^3$ is

$$\mathbb{V}(R, h) = \int_{x=R-h}^{R} \pi r^2(x)dx = \int_{x=R-h}^{R} \pi[R^2 - x^2]dx = \frac{\pi h^2}{3}(3R - h). \tag{18}$$

It follows that if $0 < R_1 < R_0$ and $R_0 - R_1 < d \leq R_0 + R_1$ or if $R_1 \geq R_0$ and $R_1 - R_0 \leq d \leq R_1 + R_0$, we have

$$\begin{aligned} F_D(d) &= \tfrac{3}{4\pi R_0^3}\left[\mathbb{V}(d, h_1(d)) + \mathbb{V}(R_0, h_2(d))\right] \\ &= \tfrac{1}{4R_0^3}\Big[h_1^2(d)(3d - h_1(d)) + h_2^2(d)(3R_0 - h_2(d))\Big] \end{aligned}$$

where we recall that $h_1(d)$ and $h_2(d)$ are given by (6) and the density is

$$f_D(d) = \frac{3}{4\pi R_0^3}\left[\frac{\partial \mathbb{V}}{\partial R}(d, h_1(d)) + h_1'(d)\frac{\partial \mathbb{V}}{\partial h}(d, h_1(d)) + h_2'(d)\frac{\partial \mathbb{V}}{\partial h}(R_0, h_2(d))\right]$$

where

$$\frac{\partial \mathbb{V}(R, h)}{\partial R} = \pi h^2 \text{ and } \frac{\partial \mathbb{V}(R, h)}{\partial h} = \pi h(2R - h).$$

The density of D when X is uniformly distributed in a ball is given for some examples in Figure 9.

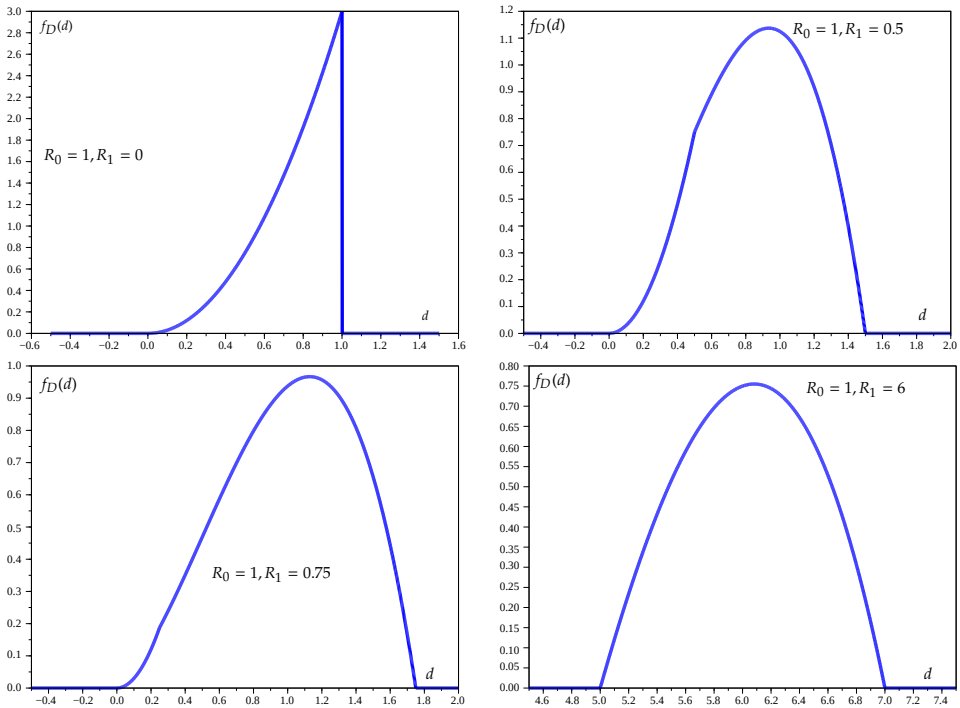

Fig. 9. Density of D when X is uniformly distributed in a ball of radius $R_0 = 1$: some examples. Top left: $R_1 = 0$, top right: $R_1 = 0.5$, bottom left: $R_1 = 0.75$, bottom right: $R_1 = 6$.

5. Distance to a random variable uniformly distributed in a polygone

5.1. *Distance to a random variable uniformly distributed on a line segment*

Let $\mathcal{S} = \overline{AB}$ be a line segment in $\mathbb{R}^3$ with $A \neq B$ and let $P \in \mathbb{R}^3$. We introduce the projection P_0 of P onto line (AB):

$$P_0 = A + \frac{\langle \overrightarrow{AB}, \overrightarrow{AP} \rangle}{\|\overrightarrow{AB}\|_2^2} \overrightarrow{AB}.$$

This projection P_0 belongs to line segment $\overline{AB}$ if and only if $\langle \overrightarrow{P_0A}, \overrightarrow{P_0B} \rangle \leq 0$ (see Figure 10). In this case, setting $d_{\min} = \min(\|\overrightarrow{PA}\|_2, \|\overrightarrow{PB}\|_2)$ and $d_{\max} = \max(\|\overrightarrow{PA}\|_2, \|\overrightarrow{PB}\|_2)$,

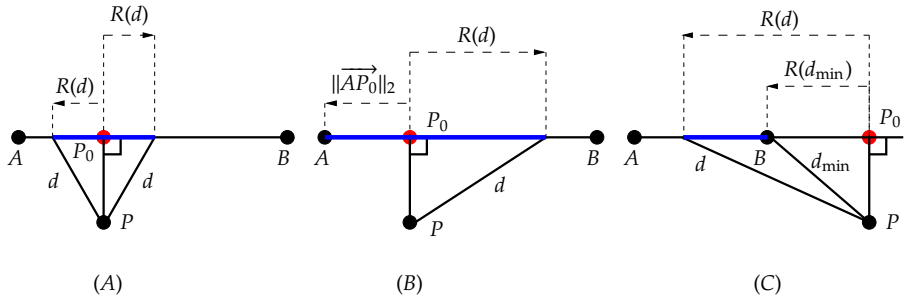

Fig. 10. Distance to a random variable uniformly distributed on a line segment.

we obtain the following CDF for D (see Figure 10):

$$\begin{cases} F_D(d) = 0 & \text{if } d < \|\overrightarrow{PP_0}\|_2, \\ F_D(d) = \frac{2R(d)}{\|\overrightarrow{AB}\|_2} = \frac{2\sqrt{d^2 - \|\overrightarrow{PP_0}\|_2^2}}{\|\overrightarrow{AB}\|_2} & \text{if } \|\overrightarrow{PP_0}\|_2 \leq d \leq d_{\min}, \\ F_D(d) = \frac{\min(\|\overrightarrow{P_0A}\|_2, \|\overrightarrow{P_0B}\|_2) + R(d)}{\|\overrightarrow{AB}\|_2} & \\ \qquad = \frac{\min(\|\overrightarrow{P_0A}\|_2, \|\overrightarrow{P_0B}\|_2) + \sqrt{d^2 - \|\overrightarrow{PP_0}\|_2^2}}{\|\overrightarrow{AB}\|_2} & \text{if } d_{\min} \leq d \leq d_{\max}, \\ F_D(d) = 1 & \text{if } d > d_{\max}. \end{cases} \tag{19}$$

If P_0 does not belong to $\overline{AB}$, i.e., if $\langle \overrightarrow{P_0A}, \overrightarrow{P_0B} \rangle > 0$, we obtain the following CDF for D (see Figure 10):

$$\begin{cases} F_D(d) = 0 & \text{if } d < d_{\min}, \\ F_D(d) = \frac{R(d) - R(d_{\min})}{\|\overrightarrow{AB}\|_2} = \frac{\sqrt{d^2 - \|\overrightarrow{PP_0}\|_2^2} - \sqrt{d_{\min}^2 - \|\overrightarrow{PP_0}\|_2^2}}{\|\overrightarrow{AB}\|_2} & \text{if } d_{\min} \leq d \leq d_{\max}, \\ F_D(d) = 1 & \text{if } d > d_{\max}. \end{cases} \tag{20}$$

An analytic expression of the density can be obtained deriving the above CDF. The density of D when X is uniformly distributed in a line segment is given for two examples in Figure 11.

5.2. *Simple polygone: an algorithm based on Green's theorem*

Let $\mathcal{S}$ be a simple polygone contained in a plane given by its extremal points $\{S_1, S_2, \ldots, S_n\}$ where the boundary of $\mathcal{S}$ is $\cup_{i=1}^{n} \overline{S_iS_{i+1}}$ with the convention that $S_{n+1} = S_1$ and where $S_i \neq S_j$ for $i \neq j$ with $1 \leq i, j \leq n$. We assume that when travelling on the boundary of $\mathcal{S}$ from S_1 to S_2, then from S_2 to S_3 and so on until the last line segment $\overline{S_nS_1}$, one always has the relative interior of $\mathcal{S}$ to the left (see Figure 12). Let P be a point in the plane $\mathcal{P}$ containing $\mathcal{S}$. $F_D(d)$ is the area of the intersection of $\mathcal{S}$ and the disk $\mathcal{D}(P, d)$ of center P and radius d divided by the area of $\mathcal{S}$. These areas will be computed making use of a special case of Green's theorem: if $\mathbb{D}$ is a

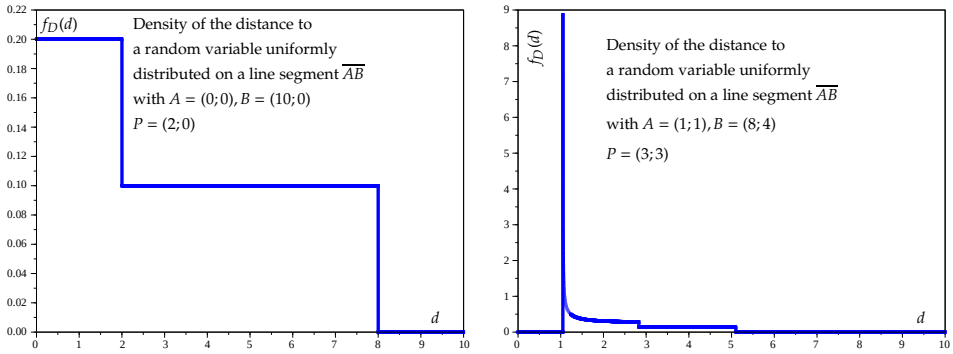

Fig. 11. Density of D when X is uniformly distributed on a line segment $\overline{AB}$: some examples.

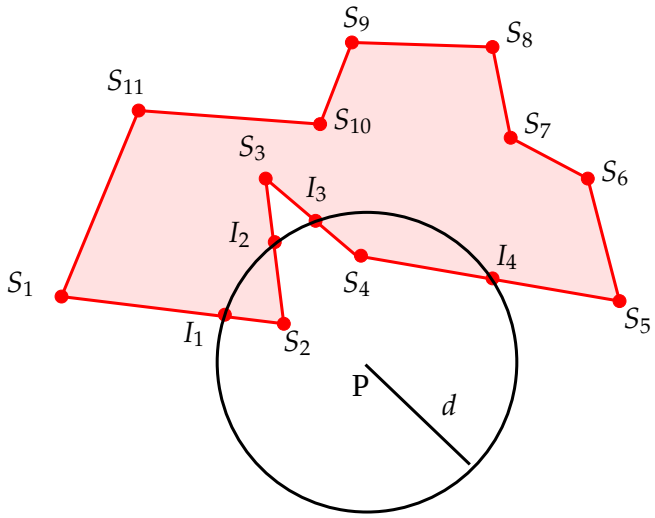

Fig. 12. Random variable uniformly distributed in a polyhedron in the plane.

closed and bounded region in the plane then the area $\mathcal{A}(\mathbb{D})$ of $\mathbb{D}$ can be expressed as a line integral over the boundary $\partial\mathbb{D}$ of D:

$$\mathcal{A}(\mathbb{D}) = \frac{1}{2}\oint_{\partial\mathbb{D}} [xdy - ydx]. \tag{21}$$

Since the boundary of $\mathcal{S}$ is a union of line segments and the boundary of $\mathcal{S} \cap \mathcal{D}(P,d)$ is made of line segments and arcs, we need to compute $\int_C [xdy - ydx]$ with C a line segment or an arc. If $C = \overline{AB}$ is a line segment, denoting respectively the coordinates of A and B by (x_A, y_A) and (x_B, y_B), we obtain

$$\mathcal{I}_{\overline{AB}} := \int_{\overline{AB}} [xdy - ydx] = y_B x_A - y_A x_B. \tag{22}$$

Now let $C = \widehat{AB}_{R_0,P}$ be an arc starting at $A = (x_A, y_A)$ and ending at $B = (x_B, y_B)$ with A and B belonging to the circle of center $P = (x_P, y_P)$ and radius $R_0 > 0$. We assume that when travelling along the arc from A to B, the relative interior of the disk is to the left. If $\theta(A, B)$ is the angle $\angle APB$, using (21) we obtain

$$\frac{R_0^2\theta(A,B)}{2} = \frac{1}{2}\Big(\mathcal{I}_{\widehat{AB}_{R_0,P}} + \mathcal{I}_{\overline{PA}} + \mathcal{I}_{\overline{BP}}\Big)$$

where $\mathcal{I}_{\widehat{AB}_{R_0,P}} := \displaystyle\int_{\widehat{AB}_{R_0,P}} [xdy - ydx]$. Using (22), the above relation can be written

$$\mathcal{I}_{\widehat{AB}_{R_0,P}} = R_0^2\theta(A,B) + x_P(y_B - y_A) - y_P(x_B - x_A). \tag{23}$$

We introduce the function $\texttt{Angle}$ defined on the boundary of $\mathcal{D}(P,d)$ taking values in $[0, 2\pi[$ and given by

$$\begin{array}{l}\texttt{Angle}(x,y) = \mathrm{Arccos}\left(\frac{x-x_P}{R_0}\right) \text{ if } y \geq y_P \text{ and}\\ \texttt{Angle}(x,y) = 2\pi - \mathrm{Arccos}\left(\frac{x-x_P}{R_0}\right) \text{ if } y < y_P.\end{array} \tag{24}$$

This function associates to a point of the boundary of $\mathcal{D}(P,d)$ its angle. With this notation, for two points $A = (x_A, y_A)$ and $B = (x_B, y_B)$ of the boundary of $\mathcal{D}(P,d)$, we have

$$\begin{array}{l}\theta(A,B) = \texttt{Angle}(x_B,y_B) - \texttt{Angle}(x_A,y_A) \text{ if } \texttt{Angle}(x_A,y_A) \leq \texttt{Angle}(x_B,y_B)\\ \theta(A,B) = 2\pi + \texttt{Angle}(x_B,y_B) - \texttt{Angle}(x_A,y_A) \text{ otherwise}\end{array}$$

and formula (23) can be written

$$\begin{cases}\mathcal{I}_{\widehat{AB}_{R_0,P}} = R_0^2(\texttt{Angle}(x_B,y_B) - \texttt{Angle}(x_A,y_A)) + x_P(y_B - y_A) - y_P(x_B - x_A)\\ \text{if } \texttt{Angle}(x_A,y_A) \leq \texttt{Angle}(x_B,y_B) \text{ and}\\ \mathcal{I}_{\widehat{AB}_{R_0,P}} = R_0^2(2\pi + \texttt{Angle}(x_B,y_B) - \texttt{Angle}(x_A,y_A)) + x_P(y_B - y_A)\\ \qquad\qquad -y_P(x_B - x_A) \text{ if } \texttt{Angle}(x_A,y_A) > \texttt{Angle}(x_B,y_B).\end{cases} \tag{25}$$

To compute the area of the intersection $\mathcal{S} \cap \mathcal{D}(P,d)$, we need to determine the intersections between the boundary of $\mathcal{S}$ and the circle $\mathcal{C}(P,d)$ of center P and radius d. This will be done using Algorithm 1 which computes the intersection between a given line segment $\overline{AB}$ with $A \neq B$ and the sphere of center P and radius d in $\mathbb{R}^3$. When this intersection is nonempty, let $I_1(d)$ and $I_2(d)$ be the intersection points (eventually $I_1(d) = I_2(d)$). Writing $I_i(d)$ as

$$I_i(d) = A + t_i\overrightarrow{AB}, \tag{26}$$

t_i solves $\|\overrightarrow{PA} + t_i\overrightarrow{AB}\|_2^2 = d^2$. Introducing

$$\Delta = \langle\overrightarrow{PA}, \overrightarrow{AB}\rangle^2 - \|\overrightarrow{AB}\|_2^2(\|\overrightarrow{PA}\|_2^2 - d^2), \tag{27}$$

if $\Delta < 0$ then the boundary of $\mathcal{S}$ and $\mathcal{C}(P,d)$ have an empty intersection while if $\Delta \geq 0$ the intersections $I_1(d)$ and $I_2(d)$ are given by (26) where

$$t_i = \frac{-\langle \overrightarrow{PA}, \overrightarrow{AB} \rangle \pm \sqrt{\Delta}}{\|\overrightarrow{AB}\|_2^2}. \tag{28}$$

We are now in a position to write Algorithm 1, observing that $I_i(d) \in (AB)$ belongs to line segment $\overline{AB}$ if and only if $\langle \overrightarrow{I_i(d)A}, \overrightarrow{I_i(d)B} \rangle \leq 0$.

Algorithm 1: Computation of the intersection points between line segment $\overline{AB}$ with $A \neq B$ and the sphere of center P and radius d in $\mathbb{R}^3$.

Inputs: A, B, P, d.

Initialization: `N=0;` //Will store the number of intersections (0, 1, or 2).
`List_Intersections=Null;` //Will store the intersection points.

//Check if line (AB) and the sphere have an empty intersection or not
Compute $\Delta = \langle \overrightarrow{PA}, \overrightarrow{AB} \rangle^2 - \|\overrightarrow{AB}\|_2^2(\|\overrightarrow{PA}\|_2^2 - d^2)$.
If $\Delta \geq 0$ **then** //if $\Delta < 0$ the intersection is empty.
  If $\Delta = 0$ **then** //the intersection of (AB) and the sphere is a singleton $\{I\}$
    Compute $I = A + t\overrightarrow{AB}$ where $t = \frac{-\langle \overrightarrow{PA}, \overrightarrow{AB} \rangle}{\|\overrightarrow{AB}\|_2^2}$ (see (26), (28)) and
    check if I belongs to $\overline{AB}$:
    If $\langle \overrightarrow{IA}, \overrightarrow{IB} \rangle \leq 0$, **then** //$I$ belongs to $\overline{AB}$
      `List_Intersections=`$\{I\}$, `N=1`.
    End If
  Else
    Compute the intersections $I_1(d)$ and $I_2(d)$ of (AB) and the sphere
    given by (26), (28).
    If $\langle \overrightarrow{I_1(d)A}, \overrightarrow{I_1(d)B} \rangle \leq 0$ **then** //$I_1(d)$ belongs to $\overline{AB}$
      If $\langle \overrightarrow{I_2(d)A}, \overrightarrow{I_2(d)B} \rangle \leq 0$ **then** //$I_2(d)$ belongs to $\overline{AB}$
        //$I_1(d)$ and $I_2(d)$ belong to $\overline{AB}$
        `List_Intersections=`$\{I_1(d), I_2(d)\}$, `N=2`.
      Else //Only $I_1(d)$ belongs to the intersection
        `List_Intersections=`$\{I_1(d)\}$, `N=1`.
      End If
    Else
      If $\langle \overrightarrow{I_2(d)A}, \overrightarrow{I_2(d)B} \rangle \leq 0$ **then** //$I_2(d)$ belongs to $\overline{AB}$
        `List_Intersections=`$\{I_2(d)\}$, `N=1`.
      End If
    End If
End If

End If

Outputs: `N`, `List_Intersections`.

Algorithm 4 which computes the CDF of D will also make use of Algorithm 2 that (i) computes the minimal distance $d_{\min}$ and maximal distance $d_{\max}$ between P and the boundary of $\mathcal{S}$, (ii) computes the area of $\mathcal{S}$, and (iii) determines if P belongs to the relative interior of $\mathcal{S}$ or not. The computation of the area of $\mathcal{S}$ will be done using formula (21). To know if P belongs to the relative interior of $\mathcal{S}$ or not, we compute the *crossing number* (stored in variable `Crossing_Number` of Algorithm 2) for point P and polyhedron $\mathcal{S}$. Let R be the ray starting at P and parallel to the positive x-axis. The crossing number counts the number of times ray R crosses the boundary of $\mathcal{S}$ going either from the inside to the outside of $\mathcal{S}$ or from the outside to the inside of $\mathcal{S}$. If the crossing number is odd then P belongs to the relative interior of $\mathcal{S}$. Otherwise, the crossing number is even and P is on the boundary of $\mathcal{S}$ or outside $\mathcal{S}$.

Though the computation of the crosssing number (the value of variable `Crossing _Number` in the end of Algorithm 2) is known (see for instance O'Rourke [1998]), we recall it here for the sake of self-completeness. For each edge $\overline{S_iS_{i+1}}$ of the polygone, we consider its intersection with R. Each time a single intersection point is found that belongs to the relative interior of an edge, `Crossing_Number` increases by one. If the intersection between the edge and the ray is nonempty but is not a single point from the relative interior of the edge, then either this intersection is an extremal point or it is the whole edge. There are 8 possibles cases, denoted by A-H in Figure 13. This figure also provides the increase in the crossing number in each case. To deal with these cases, the following (known) rules are used in Algorithm 2: (a) horizontal edges (edges $\overline{S_iS_{i+1}}$ with $y_{S_i} = y_{S_{i+1}}$) are not considered, (b) for upward edges (edges $\overline{S_iS_{i+1}}$ with $y_{S_i} < y_{S_{i+1}}$), only the final vertex is counted as an intersection, and (c) for downward edges (edges $\overline{S_iS_{i+1}}$ with $y_{S_i} > y_{S_{i+1}}$), only the starting vertex is counted as an intersection.[a] The increase in the crossing number using these rules is reported for cases A-H in Figure 13. Comparing with the expected increase in the crossing number in each case, we see that variable `Crossing_Number` that is updated using these rules in Algorithm 2, will be even if and only if P is on the boundary of the polygone or outside the polygone, as expected.

Algorithm 2: Given a polygone $\mathcal{S}$ contained in a plane and a point P in that plane, the algorithm computes the area of $\mathcal{S}$, the crossing number, and the minimal and maximal distances from P to the boundary of $\mathcal{S}$.

---

[a] Alternatively, we can of course count only the starting vertices of upward edges and the final vertices of downard edges.

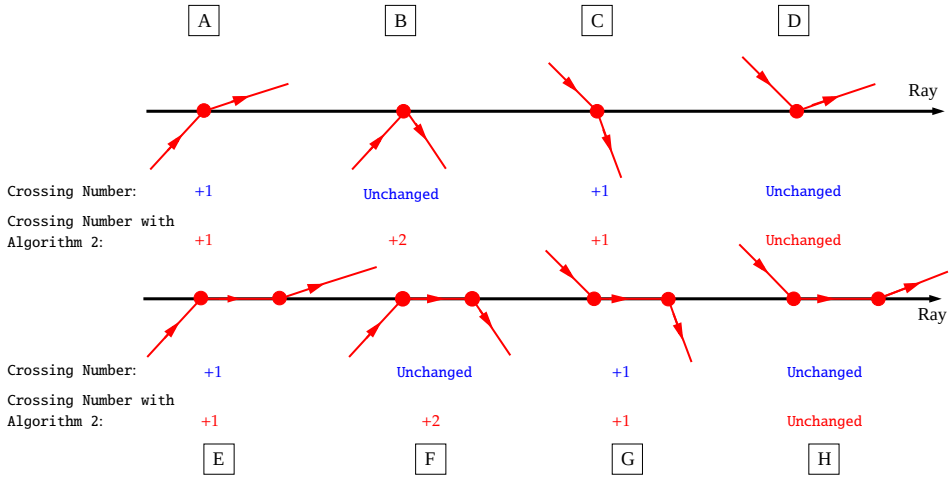

Fig. 13. Increase in the crossing number when the ray passes through an extremal point of the polygone or when an edge of the polygone is contained in the ray.

Inputs: P and the vertices $S_1, S_2, \ldots, S_n$ of a polygone contained in a plane.

Initialization: $\mathcal{L} = 0$. //Will store line integral (21) taking $\mathbb{D} = \mathcal{S}$, i.e.,
//will store $\mathcal{A}(\mathcal{S})$.
`Crossing_Number`=0. //Will store the crossing number.
$d_{\min} = +\infty$. //Will store the minimal distance from P to the boundary of $\mathcal{S}$.
$d_{\max} = 0$. //Will store the maximal distance from P to the boundary of $\mathcal{S}$.

For $i = 1, \ldots, n$,
$\mathcal{L} = \mathcal{L} + \frac{1}{2}\mathcal{I}_{\overline{S_iS_{i+1}}}$ where for a line segment $\overline{AB}$, $\mathcal{I}_{\overline{AB}}$ is given by (22).
//Computation of the crossing number
If $y_{S_i} < y_P \leq y_{S_{i+1}}$ or $y_{S_{i+1}} < y_P \leq y_{S_i}$ **then**
//Compute the abscissa x_I of the intersection I of the line $y = y_P$
//and line segment $\overline{S_iS_{i+1}}$:

$$x_I = x_{S_i} + \frac{x_{S_{i+1}} - x_{S_i}}{y_{S_{i+1}} - y_{S_i}}(y_P - y_{S_i}).$$

If $x_I > x_P$ **then**

`Crossing_Number` = `Crossing_Number` + 1

End If
End If
//Computation of the maximal distance from P to the boundary of $\mathcal{S}$
$d_{\max} = \max(d_{\max}, \|\overrightarrow{PS_i}\|_2)$
//Computation of the minimal distance from P to the boundary of $\mathcal{S}$

Compute the projection P_0 of P onto line (S_iS_{i+1}):

$$P_0 = S_i + \frac{\langle \overrightarrow{S_iP}, \overrightarrow{S_iS_{i+1}} \rangle}{\|\overrightarrow{S_iS_{i+1}}\|_2^2} \overrightarrow{S_iS_{i+1}}.$$

If $\langle \overrightarrow{P_0S_i}, \overrightarrow{P_0S_{i+1}} \rangle \leq 0$ **then**
  //P_0 belongs to $\overline{AB}$
  $d_{\min} = \min(d_{\min}, \|\overrightarrow{PP_0}\|_2)$.
Else
  $d_{\min} = \min(d_{\min}, \|\overrightarrow{PS_i}\|_2, \|\overrightarrow{PS_{i+1}}\|_2)$.
End If
End For

Outputs: `Crossing_Number`, $\mathcal{L}, d_{\min}, d_{\max}$.

The outputs of Algorithm 2 allow us to know if P belongs to $\mathcal{S}$ or not. Indeed, P belongs to $\mathcal{S}$ if and only if P belongs to the relative interior of $\mathcal{S}$, which occurs if and only if the crossing number is odd, or if P is on the boundary of $\mathcal{S}$, which occurs if and only if $d_{\min} = 0$. As a result, P belongs to $\mathcal{S}$ if and only if `Crossing_Number` is odd or $d_{\min} = 0$.

Remark. The crossing number computed replacing the condition $x_I > x_P$ by $x_I \geq x_P$ in Algorithm 2 will not necessarily be odd if P belongs to the boundary of $\mathcal{S}$. For instance, if $\mathcal{S}$ is the rectangle $\mathcal{S} = \{(x, y) \ : \ x_1 \leq x \leq x_2, \ y_1 \leq y \leq y_2\}$ then if the condition $x_I > x_P$ is replaced by $x_I \geq x_P$ in Algorithm 2, if we take $P = ((x_1 + x_2)/2, y_1)$ then variable `Crossing_Number` will be even while if we take $P = (x_2, (y_1 + y_2)/2)$ this variable will be odd. However, both points belong to the boundary of $\mathcal{S}$.

Let us now comment on Algorithm 4 that computes the cumulative distribution function of D using Algorithms 1 and 2.

We first explain the different steps of Algorithm 4 when there is at least an edge of $\mathcal{S}$ that has a nonempty intersection with both the relative interior of $\mathcal{D}(P, d)$ and the complement of $\mathcal{D}(P, d)$. In other words, we exclude for the moment the cases $\mathcal{D}(P, d) \subset \mathcal{S}$, $\mathcal{S} \subset \mathcal{D}(P, d)$, and $\mathcal{D}(P, d) \cap \mathcal{S} = \emptyset$.

In this case, at the end of Algorithm 4, ℓ stores line integral (21) with $\mathbb{D} = \mathcal{S} \cap \mathcal{D}(P, d)$, i.e., the area of $\mathcal{S} \cap \mathcal{D}(P, d)$.

In the first **For** loop of Algorithm 4, starting from $\ell = 0$, we update ℓ travelling along the edges of $\mathcal{S}$ always leaving the relative interior of $\mathcal{S}$ to the left. In the end of this loop, ℓ is the sum of line integrals (22) computed for all the line segments belonging to the boundary of $\mathcal{S} \cap \mathcal{D}(P, d)$. More precisely, at iteration i of this loop, we consider edge $\overline{S_iS_{i+1}}$. For this edge, 6 cases can happen:

(i) S_i belongs to $\mathcal{D}(P, d)$ and S_{i+1} belongs to the relative interior of $\mathcal{D}(P, d)$. In this case, the whole segment $\overline{S_iS_{i+1}}$ belongs to the boundary of $\mathcal{S} \cap \mathcal{D}(P, d)$

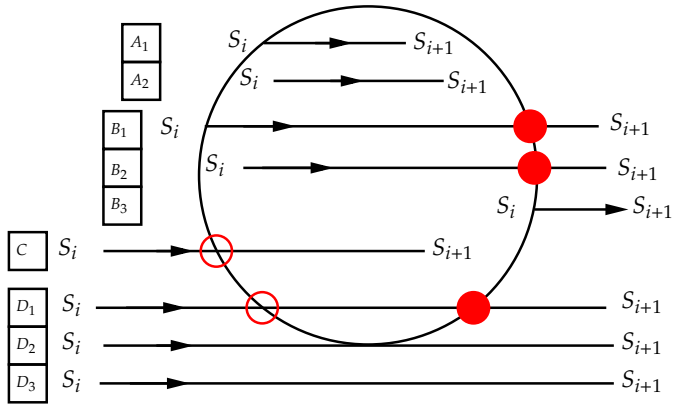

Fig. 14. Cases where S_{i+1} is not on the boundary of $\mathcal{D}(P,d)$.

and $\ell \leftarrow \ell + \frac{1}{2}\mathcal{I}_{\overline{S_iS_{i+1}}}$. This corresponds to subcases A_1 (where S_i is on the boundary of $\mathcal{D}(P,d)$) and A_2 (where S_i belongs to the relative interior of $\mathcal{D}(P,d)$) in Figure 14.

(ii) S_i belongs to $\mathcal{D}(P,d)$ and S_{i+1} does not belong to $\mathcal{D}(P,d)$. In this situation, either S_i belongs to the boundary of $\mathcal{D}(P,d)$ (subcases B_1 and B_3 in Figure 14) or S_i belongs to the relative interior of $\mathcal{D}(P,d)$ (subcase B_2 in Figure 14). If $\overline{S_iS_{i+1}}$ and $\mathcal{C}(P,d)$ have an intersection point I_i that is different from S_i then $\overline{S_iI_i}$ belongs to the boundary of $\mathcal{S} \cap \mathcal{D}(P,d)$ and $\ell \leftarrow \ell + \frac{1}{2}\mathcal{I}_{\overline{S_iI_i}}$.

(iii) S_i belongs to $\mathcal{D}(P,d)$ and S_{i+1} is on the boundary of $\mathcal{D}(P,d)$. As in (i), the whole segment $\overline{S_iS_{i+1}}$ belongs to the boundary of $\mathcal{S} \cap \mathcal{D}(P,d)$ and $\ell \leftarrow \ell + \frac{1}{2}\mathcal{I}_{\overline{S_iS_{i+1}}}$.

(iv) S_i does not belong to $\mathcal{D}(P,d)$ and S_{i+1} belongs to the relative interior of $\mathcal{D}(P,d)$ (case C in Figure 14). In this case, $\overline{S_iS_{i+1}}$ and $\mathcal{C}(P,d)$ have a single intersection point I_i, $\overline{I_iS_{i+1}}$ belongs to the boundary of $\mathcal{S} \cap \mathcal{D}(P,d)$, and $\ell \leftarrow \ell + \frac{1}{2}\mathcal{I}_{\overline{I_iS_{i+1}}}$.

(v) Both S_i and S_{i+1} are outside $\mathcal{D}(P,d)$. There are three subcases: $\overline{S_iS_{i+1}}$ and $\mathcal{C}(P,d)$ have two intersection points I_{i1} and I_{i2} (case D_1 in Figure 14); $\overline{S_iS_{i+1}}$ and $\mathcal{C}(P,d)$ have a single intersection point (case D_1 in Figure 14); or $\overline{S_iS_{i+1}}$ and $\mathcal{C}(P,d)$ have an empty intersection (case D_3 in Figure 14). In case D_1, $\overline{I_{i1}I_{i2}}$ belongs to the boundary of $\mathcal{S} \cap \mathcal{D}(P,d)$ and $\ell \leftarrow \ell + \frac{1}{2}\mathcal{I}_{\overline{I_{i1}I_{i2}}}$.

(vi) S_i does not belong to $\mathcal{D}(P,d)$ and S_{i+1} is on the boundary of $\mathcal{D}(P,d)$. If $\overline{S_iS_{i+1}}$ and $\mathcal{C}(P,d)$ have two intersection points I_i and S_{i+1} then $\overline{I_iS_{i+1}}$ belongs to the boundary of $\mathcal{S} \cap \mathcal{D}(P,d)$ and $\ell \leftarrow \ell + \frac{1}{2}\mathcal{I}_{\overline{I_iS_{i+1}}}$.

We also have to determine the arcs that belong to the boundary of $\mathcal{S} \cap \mathcal{D}(P,d)$. A simple way to do this would be as follows:

(a) store all the intersections between the edges of the polygone and the boundary of $\mathcal{D}(P,d)$.
(b) Sort these intersection points (x_i, y_i) in ascending order of their angles $\texttt{Angle}(x_i, y_i)$.
(c) To know if a given arc belongs to $\mathcal{S} \cap \mathcal{D}(P,d)$, take the middle M of this arc and compute the crossing number and $d_{\min}$ for $\mathcal{S}$ and M using Algorithm 2. The corresponding arc belongs to $\mathcal{S} \cap \mathcal{D}(P,d)$ if and only if the crossing number is odd or $d_{\min} = 0$

The complexity of this algorithm is $O(n^2)$ where n is the number of edges. Algorithm 4 which has complexity $O(n \ln n)$ selects the appropriate arcs in a more efficient manner. In this algorithm, the extremities of these arcs are stored, without repetitions, in the list `Intersections` which is updated along the iterations of the first **For** loop of Algorithm 4: `Intersections`(i) will be the i-th "relevant" (see below) intersection point found. To know the arcs that belong to $\mathcal{S} \cap \mathcal{D}(P,d)$, a second list `Arcs` is used: the i-th element of list `Arcs` is 1 if and only if the arc from the boundary of $\mathcal{D}(P,d)$ obtained starting at `Intersections`(i) and ending at the next element from list `Intersections` found travelling counter clockwise on the boundary of $\mathcal{D}(P,d)$ belongs to $\mathcal{S} \cap \mathcal{D}(P,d)$. To produce this information, when an intersection between $\mathcal{S}$ and $\mathcal{C}(P,d)$ is found we need to know the type of this intersection, knowing that there are three types of intersections:

T_1: the intersection is not "relevant", i.e., there is no arc from $\mathcal{S} \cap \mathcal{C}(P,d)$ starting or ending at this point;
T_2: there is an arc from $\mathcal{S} \cap \mathcal{C}(P,d)$ starting at this point (in this case the corresponding entry of `Arcs` is one);
T_3: there is an arc from $\mathcal{S} \cap \mathcal{C}(P,d)$ ending at this point (in this case the corresponding entry of `Arcs` is zero).

Now let us go back to the 6 cases (i)-(vi) discussed above and considered in the first **For** loop of Algorithm 4. It remains to explain how to determine in each of these cases the intersection type when an intersection is found.

First, since vertices belonging to the boundary of $\mathcal{D}(P,d)$ are starting vertices of an edge and ending vertices of another edge, to avoid counting them twice, we do not consider the intersection points that are starting vertices of an edge. With this convention, in case (i), i.e., subcases A_1 and A_2 in Figure 14, we do not need to store intersection points, even if S_i belongs to $\mathcal{D}(P,d)$.

In case (ii), corresponding to subcases B_1, B_2, and B_3 in Figure 14, if $\overline{S_iS_{i+1}}$ and $\mathcal{C}(P,d)$ have an intersection point that is different from S_i then this intersection point is stored in list `Intersections` and it is of type T_2: the corresponding entry in `Arcs` is one (these type T_2 intersections are represented by red balls in Figure 14).

In case (iv), corresponding to case C in Figure 14, there is a single intersection point between $\overline{S_iS_{i+1}}$ and $\mathcal{C}(P,d)$ and it is of type T_3: the corresponding entry in `Arcs` is zero (these type T_3 intersections are represented by red circles in Figure 14).

Case (v) corresponds to cases D_1, D_2, and D_3 in Figure 14. In subcase D_1, i.e., when $\overline{S_iS_{i+1}}$ and $C(P,d)$ have two intersections, the first one encountered when travelling from S_i to S_{i+1} is of type T_3 while the second one is of type T_2. In subcase D_2, $\overline{S_iS_{i+1}}$ and $C(P,d)$ have a single intersection which is of type T_1.

Let us now consider cases (iii) and (vi), the cases where S_{i+1} is on the boundary of $\mathcal{D}(P,d)$. We want to determine the intersection type for S_{i+1}. This is done using an auxiliary algorithm, Algorithm 3, that takes as entries P and d (the center and radius of $C(P,d)$) and three successive vertices S_i, S_{i+1}, and S_{i+2} of $\mathcal{S}$, knowing that S_{i+1} is on the boundary of $\mathcal{D}(P,d)$. The output variable `Arc` of this algorithm is one (resp. zero) if and only if S_{i+1} is of type T_2 or T_3 (resp. type T_1). What matters to determine the intersection type for S_{i+1} is whether $\overline{S_iS_{i+1}}$ is contained in some half-space (to be specified below) that does not contain P or not. An additional input variable of Algorithm 3 described below, variable `In`, takes the value zero in the former case and the value one in the latter case. To explain this algorithm, it is convenient to introduce two half spaces $\mathcal{H}_{\texttt{Right}}$ and $\mathcal{H}_P$ and a line L_1. These half spaces and lines depend on the entries of Algorithm 3, i.e., P and d (the center and radius of $C(P,d)$) and three successive vertices S_i, S_{i+1}, and S_{i+2} of $\mathcal{S}$. Line L_1 is the line that contains line segment $\overline{S_iS_{i+1}}$. The open half space $\mathcal{H}_{\texttt{Right}}$ is the set of points that are to the right of line L_1 when travelling on this line in the direction $S_i \rightarrow S_{i+1}$. Denoting by L_2 the line that is tangent to the circle $C(P,d)$ at S_{i+1} (recall that S_{i+1} belongs to $C(P,d)$), the closed half space $\mathcal{H}_P$ is the set of points that are on the side of line L_2 that does not contain P, including L_2. The definitions of these sets follow.

For L_1 and $\mathcal{H}_{\texttt{Right}}$, we obtain:

$$\begin{cases} \text{if } x_{S_{i+1}} = x_{S_i} \text{ and } y_{S_{i+1}} > y_{S_i} \text{ then} \\ \quad \begin{cases} L_1 = \{(x,y) \ : \ x = x_{S_i}\}, \\ \mathcal{H}_{\texttt{Right}} = \{(x,y) \ : \ x > x_{S_i}\}. \end{cases} \\ \text{If } x_{S_{i+1}} = x_{S_i} \text{ and } y_{S_{i+1}} < y_{S_i} \text{ then} \\ \quad \begin{cases} L_1 = \{(x,y) \ : \ x = x_{S_i}\}, \\ \mathcal{H}_{\texttt{Right}} = \{(x,y) \ : \ x < x_{S_i}\}. \end{cases} \\ \text{If } x_{S_{i+1}} > x_{S_i} \text{ then} \\ \quad \begin{cases} L_1 = \{(x,y) \ : \ y = y_{S_i} + \frac{y_{S_{i+1}} - y_{S_i}}{x_{S_{i+1}} - x_{S_i}}(x - x_{S_i})\}, \\ \mathcal{H}_{\texttt{Right}} = \{(x,y) \ : \ y < y_{S_i} + \frac{y_{S_{i+1}} - y_{S_i}}{x_{S_{i+1}} - x_{S_i}}(x - x_{S_i})\}. \end{cases} \\ \text{If } x_{S_{i+1}} < x_{S_i} \text{ then} \\ \quad \begin{cases} L_1 = \{(x,y) \ : \ y = y_{S_i} + \frac{y_{S_{i+1}} - y_{S_i}}{x_{S_{i+1}} - x_{S_i}} x_{S_i}(x - x_{S_i})\}, \\ \mathcal{H}_{\texttt{Right}} = \{(x,y) \ : \ y > y_{S_i} + \frac{y_{S_{i+1}} - y_{S_i}}{x_{S_{i+1}} - x_{S_i}}(x - x_{S_i})\}. \end{cases} \end{cases} \tag{29}$$

Next observe that $M = (x,y) \in \mathcal{H}_P$ if and only if $\langle \overrightarrow{S_{i+1}M}, \overrightarrow{S_{i+1}P} \rangle \leq 0$ and therefore

$$\mathcal{H}_P = \{(x,y) \ : \ (x - x_{S_{i+1}})(x_P - x_{S_{i+1}}) + (y - y_{S_{i+1}})(y_P - y_{S_{i+1}}) \leq 0\}. \tag{30}$$

Let us first consider the case when input variable `In` of Algorithm 3 is one, i.e., the case when S_i does not belong to $\mathcal{H}_P$. In this case, the edge $\overline{S_{i+1}S_{i+2}}$ can belong to three different regions, denoted by $\mathcal{R}_1$, $\mathcal{R}_2$, and $\mathcal{R}_3$ in Figure 15 and respectively

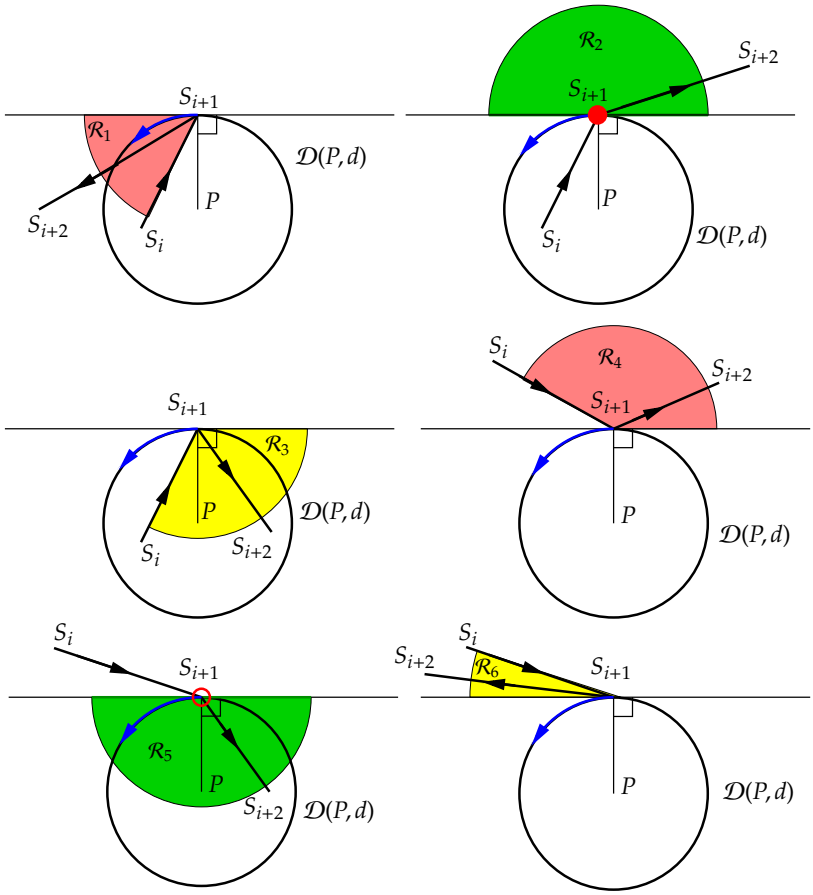

Fig. 15. The six cases where an endpoint S_{i+1} of an edge is on the boundary of $\mathcal{D}(P,d)$.

represented in pink at the top left, in green at the top right, and in yellow in the middle left figures of Figure 15. In this Figure 15, type T_2 intersections are represented by red balls while type T_3 intersections are represented by red circles. Regions $\mathcal{R}_1, \mathcal{R}_2$, and $\mathcal{R}_3$ are given by (see Figure 15):

$$\mathcal{R}_1 = \overline{\mathcal{H}_P} \cap \overline{\mathcal{H}_{\texttt{Right}} \cup L_1}, \quad \mathcal{R}_2 = \mathcal{H}_P, \text{ and } \mathcal{R}_3 = \overline{\mathcal{H}_P} \cap \mathcal{H}_{\texttt{Right}}.$$

If S_{i+2} belongs to $\mathcal{R}_1$ or $\mathcal{R}_3$, then S_{i+1} is a type T_1 intersection while if S_{i+2} belongs to $\mathcal{R}_2$ S_{i+1} is a type T_2 intersection.

We now consider the case where input variable `In` of Algorithm 3 is zero, i.e., the case where S_{i+1} belongs to $\mathcal{H}_P$. In this case, S_{i+2} can belong to three different regions, denoted by $\mathcal{R}_4$, $\mathcal{R}_5$, and $\mathcal{R}_6$ in Figure 15 and respectively represented in pink in the middle right, in green in the bottom left, and in yellow in the bottom right figures of Figure 15.

Regions $\mathcal{R}_4, \mathcal{R}_5$, and $\mathcal{R}_6$ are given by (see Figure 15):

$$\mathcal{R}_4 = \mathcal{H}_P \cap \overline{\mathcal{H}_{\texttt{Right}} \cup L_1}, \quad \mathcal{R}_5 = \overline{\mathcal{H}_P}, \text{ and } \mathcal{R}_6 = \mathcal{H}_P \cap \mathcal{H}_{\texttt{Right}}.$$

If S_{i+2} belongs to $\mathcal{R}_4$ or $\mathcal{R}_6$ then S_{i+1} is a type T_1 intersection while if S_{i+2} belongs to $\mathcal{R}_5$ S_{i+2} is a type T_3 intersection.

Summarizing our observations, if S_{i+1} belongs to the boundary of $\mathcal{D}(P,d)$, this intersection is stored as a "relevant" intersection (it is not a type T_1 intersection) if and only if `In`=1 and $S_{i+2} \in \mathcal{R}_2$ (in this case, it is a type T_2 intersection) or `In`=0 and $S_{i+2} \in \mathcal{R}_5$ (in this case, it is a type T_3 intersection).

Algorithm 3: Given three successive vertices S_i, S_{i+1}, and S_{i+2} of a simple polygone $\mathcal{S}$ and a circle of center P and radius $d > 0$ with S_{i+1} belonging to this circle, the algorithm determines if S_{i+1} is or is not a starting or ending point of an arc from the boundary of $\mathcal{D}(P,d) \cap \mathcal{S}$.

Inputs: $P, d, S_i, S_{i+1}, S_{i+2}$, `In`.

Initialization: `Arc`=0.

If `In` and $(x_{S_{i+2}} - x_{S_{i+1}})(x_P - x_{S_{i+1}}) + (y_{S_{i+2}} - y_{S_{i+1}})(y_P - y_{S_{i+1}}) \leq 0$ **then** `Arc` = 1.
Else if $\overline{\texttt{In}}$ and $(x_{S_{i+2}} - x_{S_{i+1}})(x_P - x_{S_{i+1}}) + (y_{S_{i+2}} - y_{S_{i+1}})(y_P - y_{S_{i+1}}) > 0$ **then** `Arc` = 1.
End if

Output: `Arc`.

In the end of the first **For** loop of Algorithm 4, the "relevant" intersections points (x_i, y_i) of $\mathcal{S}$ and $C(P,d)$ are stored in list `Intersections`. We then sort these intersections in ascending order of their angles `Angle`(x_i, y_i) where we recall that `Angle` is defined in (24). The values in list `Arcs` are sorted correspondingly. For `Nb_Intersections` intersection points, this defines `Nb_Intersections` arcs on the circle. At i-th iteration of the second **For** loop of Algorithm 4, the i-th arc is considered. If this arc belongs to $\mathcal{S} \cap \mathcal{D}(P,d)$, i.e., if `Arcs`$(i) = 1$, the corresponding line integral (25) is computed. The sum of these line integrals makes up the last part of line integral (21) for $\mathbb{D} = \mathcal{D}(P,d) \cap \mathcal{S}$.

It remains to check that the algorithm correctly computes $F_D(d)$ when variable `Nb_Intersections` in the end of Algorithm 4 is null. This can occur in three different manners reported in Figure 16: (i) $\mathcal{D}(P,d) \cap \mathcal{S} = \emptyset$, (ii) the polygone $\mathcal{S}$ is contained in $\mathcal{D}(P,d)$, and (iii) the disk $\mathcal{D}(P,d)$ is contained in $\mathcal{S}$. Case (ii) corresponds to $\ell = \mathcal{L}$ and in this case $F_D(d) = 1$. If $\ell \neq \mathcal{L}$, case (i) occurs when P is outside $\mathcal{S}$ and case (iii) when P belongs to the relative interior of $\mathcal{S}$. To know if case (i) or case (iii) occurs, we use the *crossing number* computed by Algorithm 2. If the crossing number is odd then P is inside $\mathcal{S}$ and $F_D(d) = \pi d^2/\mathcal{L}$. Otherwise, the crossing number is even, P is outside $\mathcal{S}$ (case (i)) and $F_D(d) = 0$.

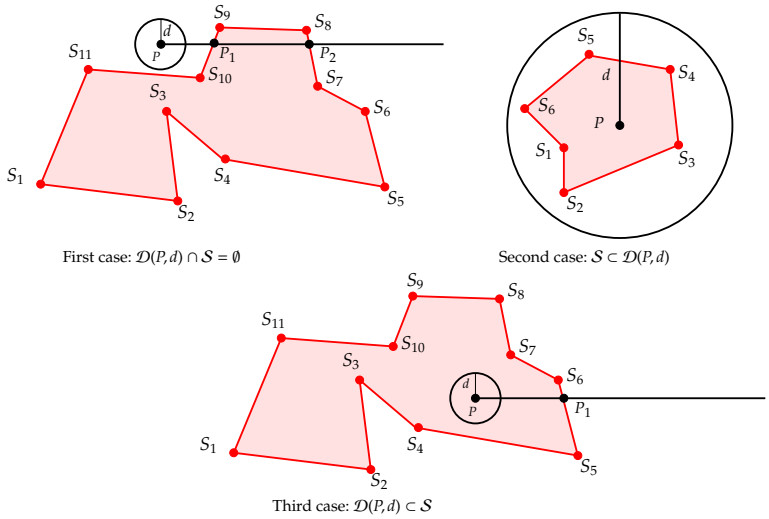

Fig. 16. Cases where `Nb_Intersections`=0.

Algorithm 4: Computation of the value $F_D(d)$ of the cumulative distribution function of D at d when X is uniformly distributed in a polygone contained in a plane with P in that plane.

Inputs: P, the vertices $S_1, \ldots, S_n$, of polygone $\mathcal{S}$, `Crossing_Number`, $\mathcal{L}$, d.

Initialization: $\ell = 0$ //Will store line integral (21) taking $\mathbb{D} = \mathcal{S} \cap \mathcal{D}(P,d)$,
//i.e., will compute the area of $\mathbb{D} = \mathcal{S} \cap \mathcal{D}(P,d)$.
`Intersections=Null`. //List of the intersections found for $\mathcal{S}$ and $\mathcal{C}(P,d)$.
`Nb_Intersections`=0. //Number of intersections found for $\mathcal{S}$ and $\mathcal{C}(P,d)$.
`Arcs=Null`. //Stores the arcs that are on the boundary of $\mathcal{S} \cap \mathcal{D}(P,d)$.

For $i = 1, \ldots, n$,
//Check if S_i belongs to $\mathcal{D}(P,d)$ or not:
If $\|\overrightarrow{S_iP}\|_2 \leq d$ **then**
If $\|\overrightarrow{S_{i+1}P}\|_2 < d$ **then** //Cases A_1 and A_2 in Figure 14
$\ell \leftarrow \ell + \frac{1}{2}\mathcal{I}_{\overline{S_iS_{i+1}}}$ where for a line segment $\overline{AB}$, $\mathcal{I}_{\overline{AB}}$ is given by (22).
Else If $\|\overrightarrow{S_{i+1}P}\|_2 > d$ //Cases B_1, B_2, and B_3 in Figure 14
Call Algorithm 1 to compute the intersections between the circle of center P and radius d with the line segment $\overline{S_iS_{i+1}}$.
If there is an intersection point different from S_i **then**
Let I_i be this intersection point.

$\ell \leftarrow \ell + \frac{1}{2}\mathcal{I}_{\overline{S_i I_i}}$ where for a line segment $\overline{AB}$, $\mathcal{I}_{\overline{AB}}$ is given by (22).
`Nb_Intersections` $\leftarrow$ `Nb_Intersections` + 1.
`Intersections[Nb_Intersections]` = I_i.
`Arcs[Nb_Intersections]` = 1.
End If
Else
$\ell \leftarrow \ell + \frac{1}{2}\mathcal{I}_{\overline{S_i S_{i+1}}}$ where for a line segment $\overline{AB}$, $\mathcal{I}_{\overline{AB}}$ is given by (22).
Call Algorithm 3 with input variables $P, d, S_i, S_{i+1}, S_{i+2}$ and with variable `In` set to 1.
If the variable `Arc` returned by this algorithm is 1 **then**
`Nb_Intersections` $\leftarrow$ `Nb_Intersections` + 1.
`Intersections[Nb_Intersections]` = S_{i+1}.
`Arcs[Nb_Intersections]` = 1.
End If
End If
Else
If $\|\overrightarrow{S_{i+1}P}\|_2 < d$ **then** //Case C in Figure 14
Call Algorithm 1 to compute the intersection I_i between the circle of center P and radius d with the line segment $\overline{S_i S_{i+1}}$ (note that the intersection is a single point).
$\ell \leftarrow \ell + \frac{1}{2}\mathcal{I}_{\overline{I_i S_{i+1}}}$ where for a line segment $\overline{AB}$, $\mathcal{I}_{\overline{AB}}$ is given by (22).
`Nb_Intersections` $\leftarrow$ `Nb_Intersections` + 1.
`Intersections[Nb_Intersections]` = I_i.
`Arcs[Nb_Intersections]` = 0.
Else If $\|\overrightarrow{S_{i+1}P}\|_2 > d$ **then** //Cases D_1, D_2, and D_3 in Figure 14
Call Algorithm 1 to compute the intersections between the circle of center P and radius d with the line segment $\overline{S_i S_{i+1}}$.
If there are two intersection points **then**
Let I_{i1} and I_{i2} be these intersection points where I_{i1} and I_{i2} satisfy $\frac{x_{I_{i1}} - x_{S_i}}{x_{S_{i+1}} - x_{S_i}} \leq \frac{x_{I_{i2}} - x_{S_i}}{x_{S_{i+1}} - x_{S_i}}$ if $x_{S_{i+1}} \neq x_{S_i}$ and $\frac{y_{I_{i1}} - y_{S_i}}{y_{S_{i+1}} - y_{S_i}} \leq \frac{y_{I_{i2}} - y_{S_i}}{y_{S_{i+1}} - y_{S_i}}$ if $x_{S_{i+1}} = x_{S_i}$.
$\ell \leftarrow \ell + \frac{1}{2}\mathcal{I}_{\overline{I_{i1} I_{i2}}}$ where for a line segment $\overline{AB}$, $\mathcal{I}_{\overline{AB}}$ is given by (22).
`Nb_Intersections` $\leftarrow$ `Nb_Intersections` + 2.
`Intersections[Nb_Intersections` $-$ `1]` = I_{i1}.
`Intersections[Nb_Intersections]` = I_{i2}.
`Arcs[Nb_Intersections` $-$ `1]` = 0.
`Arcs[Nb_Intersections]` = 1.
End If
Else
Call Algorithm 1 to compute the intersections between the circle of center P and radius d with the line segment $\overline{S_i S_{i+1}}$.
If there is one intersection **then**

Call Algorithm 3 with input variables $P, d, S_i, S_{i+1}, S_{i+2}$
and with variable `In` set to 0.
If the variable `Arc` returned by this algorithm is 1 **then**
`Nb_Intersections` $\leftarrow$ `Nb_Intersections` + 1.
`Intersections`[`Nb_Intersections`] = S_{i+1}.
`Arcs`[`Nb_Intersections`] = 0.
End If
Else If there are two intersections I_i and S_{i+1} **then**
$\ell \leftarrow \ell + \frac{1}{2}\mathcal{I}_{\overline{I_i S_{i+1}}}$ where for a line segment $\overline{AB}$,
$\mathcal{I}_{\overline{AB}}$ is given by (22).
`Nb_Intersections` $\leftarrow$ `Nb_Intersections` + 1.
`Intersections`[`Nb_Intersections`] = I_i.
`Arcs`[`Nb_Intersections`] = 0.
Call Algorithm 3 with input variables $P, d, S_i, S_{i+1}, S_{i+2}$
and with variable `In` set to 1.
If the variable `Arc` returned by this algorithm is 1 **then**
`Nb_Intersections` $\leftarrow$ `Nb_Intersections` + 1.
`Intersections`[`Nb_Intersections`] = S_{i+1}.
`Arcs`[`Nb_Intersections`] = 1.
End If
End If
End If
End If
End For

If `Nb_Intersections`=0 **then**
If $\ell = \mathcal{L}$ **then**
$F_D(d) = 1$
Else if variable `Crossing_Number` is odd **then**
//$\mathcal{D}(P, d)$ is inside the polygone
$F_D(d) = \pi d^2/\mathcal{L}$
Else
//$\mathcal{D}(P, d)$ has no intersection with the polygone
$F_D(d) = 0$
End If
Else
Sort the elements (x_i, y_i) of list `Intersections`
by ascending order of their angles `Angle`(x_i, y_i) and sort the elements of list
`Arcs` correspondingly.
Let again `Intersections` and `Arcs` be the corresponding sorted lists.
For $i = 1, \ldots,$ `Nb_Intersections`
If `Arcs`[i]=1 **then**

$\ell \leftarrow \ell + \frac{1}{2}\mathcal{I}_{\widehat{AB}_{d,P}}$ where $A = \texttt{Intersections}[i]$,

$$B = \begin{cases} \texttt{Intersections}[i+1] & \text{if } i < \texttt{Nb\_Intersections}, \\ \texttt{Intersections}[1] & \text{if } i = \texttt{Nb\_Intersections}, \end{cases}$$

and where $\mathcal{I}_{\widehat{AB}_{d,P}}$ is obtained substituting R_0 by d in (25).

End If

End For

$F_D(d) = \ell/\mathcal{L}$.

End If

Output: $F_D(d)$.

After calling Algorithm 2, if the crossing number is odd, we know that P belongs to the relative interior of $\mathcal{S}$ and for $0 \leq d \leq d_{\min}$, we have $f_D(d) = \frac{2\pi d}{\mathcal{L}}$. For $d \geq d_{\max}$ or $d \leq 0$, the density is null. If the crossing number is even, $f_D(d)$ is null for $0 \leq d \leq d_{\min}$. For $d_{\min} \leq d \leq d_{\max}$, Algorithm 5 provides approximations of the density at points $d_i, i = 1, \ldots, N-1$.

Algorithm 5: Computation of the approximate density of D (distance from P to a random variable uniformly distributed in a polygone) in the range $[d_{\min}, d_{\max}]$.

Inputs: The vertices $S_1, \ldots, S_n$ of a polygone contained in a plane, a point P in this plane, and the number N of discretization points.

Initialization: Call Algorithm 2 to compute $d_{\min}, d_{\max}$, the crossing number, and the area $\mathcal{L}$ of $\mathcal{S}$.

$\texttt{F\_Old} = 0$.

For $i = 1, \ldots, N-1$,

Compute $d_i = d_{\min} + \frac{(d_{\max} - d_{\min})i}{N}$.

Call Algorithm 4 with input variables the crossing number, $\mathcal{L}$, $d_{\min}$, $d_{\max}$, and $d = d_i$ to compute $F_D(d_i)$.

Compute $\tilde{f_D}(d_i) = \frac{N\left[F_D(d_i) - \texttt{F\_Old}\right]}{d_{\max} - d_{\min}}$ and set $\texttt{F\_Old} = F_D(d_i)$.

End For

Outputs: $\tilde{f_D}(d_i), i = 1, \ldots, N-1$.

Finally, we consider the case where the polygone is contained in a plane $\mathcal{P}$ and P is not contained in that plane. In this situation, referring to arguments from Section 3, we can use the previous results reparametrizing the problem and replacing P and d respectively by P_0, the projection of P onto $\mathcal{P}$, and $R(d) = \sqrt{d^2 - \|\overrightarrow{PP_0}\|_2^2}$. Indeed,

since $\mathcal{S} \subset \mathcal{P}$, we have

$$\mathcal{S} \cap \mathcal{B}(P,d) = \mathcal{S} \cap \mathcal{P} \cap \mathcal{B}(P,d) = \mathcal{S} \cap \mathcal{D}(P_0, R(d))$$

where $\mathcal{D}(P_0, R(d))$ is the disk of center P_0 and radius $R(d)$ contained in the plane $\mathcal{P}$ (see Figure 8). Since S_1, S_2, and S_3 are consecutive extremal points of $\mathcal{S}$, the vectors $\overrightarrow{S_2S_1}$ and $\overrightarrow{S_2S_3}$ are linearly independent. Using Gram-Schmidt orthonormalization process, we obtain two points S_1' and S_3' of the plane $\mathcal{P}$ such that the vectors $\overrightarrow{S_2S_1'}$ and $\overrightarrow{S_2S_3'}$ are orthonormal and for any point Q in plane $\mathcal{P}$, the vector $\overrightarrow{S_2Q}$ can be uniquely written as a linear combination of these vectors. Vectors $\overrightarrow{S_2S_1'}$ and $\overrightarrow{S_2S_3'}$ are given by

$$\overrightarrow{S_2S_1'} = \frac{\overrightarrow{S_2S_1}}{\|\overrightarrow{S_2S_1}\|_2} \text{ and } \overrightarrow{S_2S_3'} = \frac{\overrightarrow{S_2S_3} - \langle \overrightarrow{S_2S_3}, \overrightarrow{S_2S_1'} \rangle \overrightarrow{S_2S_1'}}{\|\overrightarrow{S_2S_3} - \langle \overrightarrow{S_2S_3}, \overrightarrow{S_2S_1'} \rangle \overrightarrow{S_2S_1'}\|_2}.$$

It follows that if A is the $(3,2)$ matrix $[\overrightarrow{S_2S_1'}, \overrightarrow{S_2S_3'}]$ whose first column is $\overrightarrow{S_2S_1'}$ and whose second column is $\overrightarrow{S_2S_3'}$, then the matrix $A^\top A$ is invertible and the projection $P_0 = \pi_{\mathcal{P}}[P]$ of P on $\mathcal{P}$ can be expressed as

$$\overrightarrow{S_2P_0} = A(A^\top A)^{-1} A^\top \overrightarrow{S_2P}. \tag{31}$$

Before calling Algorithms 2 and 4, we need to reparametrize the problem: we write $\overrightarrow{S_2P_0} = x_{P_0}\overrightarrow{S_2S_1'} + y_{P_0}\overrightarrow{S_2S_3'} = A(x_{P_0}; y_{P_0})$ and $\overrightarrow{S_2S_i} = A(x_i; y_i)$ for $i = 1, \ldots, n$. In particular, we have $(x_1, y_1) = (\|\overrightarrow{S_2S_1}\|_2, 0)$ and $(x_2, y_2) = (0,0)$. Since A has rank 2, eventually after re-ordering the lines of A, we can assume that A is of the form $A = [A_0; a_0]$ where A_0 is a $(2,2)$ invertible matrix with $A_0(1,1) \neq 0$. Using Gaussian elimination, the system $\overrightarrow{S_2P_0} = A(x_{P_0}; y_{P_0})$ can be written $\begin{bmatrix} U_0 \\ 0 \;\; 0 \end{bmatrix} \begin{bmatrix} x_{P_0} \\ y_{P_0} \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}$ for some two-dimensional vector b and an invertible upper triangular matrix $U_0 = \begin{bmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix}$. Another by-product of Gaussian elimination is the lower triangular matrix $L_0 = \begin{bmatrix} 1 & 0 \\ L_{21} & 1 \end{bmatrix}$ such that $A = L_0U_0$ is the LU decomposition of A_0. We obtain

$$x_{P_0} = \frac{\overrightarrow{S_2P_0}(1)}{U_{11}}\Big[1 + \frac{U_{12}L_{21}}{U_{22}}\Big] - \frac{U_{12}}{U_{11}}\overrightarrow{S_2P_0}(2),\ y_{P_0} = \frac{\overrightarrow{S_2P_0}(2) - L_{21}\overrightarrow{S_2P_0}(1)}{U_{22}}, \tag{32}$$

and for $i \geq 3$,

$$x_i = \frac{\overrightarrow{S_2S_i}(1)}{U_{11}}\Big[1 + \frac{U_{12}L_{21}}{U_{22}}\Big] - \frac{U_{12}}{U_{11}}\overrightarrow{S_2S_i}(2),\ y_i = \frac{\overrightarrow{S_2S_i}(2) - L_{21}\overrightarrow{S_2S_i}(1)}{U_{22}}. \tag{33}$$

Algorithms 2, 3, and 4 can now be used with P replaced by (x_{P_0}, y_{P_0}) and where the coordinates of the extremal points of the polygone are $(x_i, y_i), i = 1, \ldots, n$. First, Algorithm 2 is called to compute the area $\mathcal{L}$ of $\mathcal{S}$, the crossing number for P_0 and $\mathcal{S}$,

and the minimal and maximal distances from P_0 to the boundary of $\mathcal{S}$, respectively denoted by $d_{\min}$ and $d_{\max}$. Recalling the definition (31) of P_0, we introduce

$$d_m = \sqrt{d_{\min}^2 + \|\overrightarrow{PP_0}\|_2^2} \text{ and } d_M = \sqrt{d_{\max}^2 + \|\overrightarrow{PP_0}\|_2^2}. \tag{34}$$

With this notation, for $d \geq d_M$ or $d \leq 0$, the density is null and if the crossing number is odd, i.e., if P_0 belongs to the relative interior of $\mathcal{S}$, then for $0 \leq d \leq d_m$, we have $f_D(d) = \frac{2\pi d}{\mathcal{L}}$. Otherwise, if the crossing number is even, $f_D(d)$ is null for $0 \leq d \leq d_m$.

For $d_m \leq d \leq d_M$, Algorithm 6 provides approximations $\tilde{f_D}(d_i)$ of the value of the density at points $d_i, i = 1, \ldots, N-1$.

Algorithm 6: Computation of the approximate density of D (distance from P to a random variable uniformly distributed in a polyhedron) in the range $[d_m, d_M]$.

Inputs: The vertices $S_1, \ldots, S_n$ of a polyhedron contained in a plane, the point P, and the number N of discretization points.

Initialization: Call Algorithm 2 with P replaced by (x_{P_0}, y_{P_0}) (see equation (32)) and where the coordinates of the extremal points of the polyhedron are $(x_i, y_i), i = 1, \ldots, n$, given by (33). This will compute the area $\mathcal{L}$ of $\mathcal{S}$, the crossing number for P_0 and $\mathcal{S}$, and the minimal and maximal distances from P_0 to the boundary of $\mathcal{S}$, respectively denoted by $d_{\min}$ and $d_{\max}$.
`F_Old`= 0.
Compute d_m and d_M given by (34).
For $i = 1, \ldots, N-1$,
  Compute $d_i = d_m + \frac{(d_M - d_m)i}{N}$.
  Call Algorithm 4 with input variables the crossing number, $\mathcal{L}$, $d_{\min}$, $d_{\max}$, and $d = \sqrt{d_i^2 - \|\overrightarrow{PP_0}\|_2^2}$ to compute $F_D(d_i)$.
  Compute $\tilde{f_D}(d_i) = \frac{N\left[F_D(d_i) - \texttt{F\_Old}\right]}{d_{\max} - d_{\min}}$ and set `F_Old` $= F_D(d_i)$.
End For

Outuds: $\tilde{f_D}(d_i), i = 1, \ldots, N-1$.

5.3. *Simple polygone: an algorithm based on a triangulation of the polygone*

Let S be a simple polygone contained in a plane given by its extremal points $\{S_1, S_2, \ldots, S_n\}$ where the boundary of S is $\bigcup_{i=1}^{n} \overline{S_i S_{i+1}}$ where $S_i \neq S_j$ for $i \neq j$ and $1 \leq i, j \leq n$.

The computation, at a given value R, of the CDF of the distance between a given point P and S requires computing the area of the intersection of S and of the ball of center P and radius R. Without loss generality, we will assume that P is in the plane containing S (if this is not the case, we can project P onto this plane and modify the value of the radius). The area of the intersection can be obtained computing a triangulation of the polyhedron (with complexity $n\log(n)$), then computing the area of intersection of disk $\mathcal{D}(P,R)$ with the $n-2$ triangles of the triangulation, and summing these $n-2$ areas. Therefore, we need to devise an algorithm to compute the area of the intersection of a disk and a triangle and we proceed as follows.

Consider a triangle with vertices A, B, C and a disk of center P and radius R. We want to compute the area of the intesection of this triangle and this disk.

We first compute:

- the n_1 intersections of $\overline{AB}$ and $\mathcal{C}(P,R)$ denoted by A_1 when $n_1 = 1$ and A_1, A_2 when $n_1 = 2$ where $\|\overrightarrow{AA_1}\|_2 < \|\overrightarrow{AA_2}\|_2$ (see Figure 17);
- the n_2 intersections of $\overline{BC}$ and $\mathcal{C}(P,R)$ denoted by B_1 when $n_2 = 1$ and B_1, B_2 when $n_2 = 2$ where $\|\overrightarrow{BB_1}\|_2 < \|\overrightarrow{BB_2}\|_2$ (see Figure 17);
- the n_3 intersections of $\overline{AC}$ and $\mathcal{C}(P,R)$ denoted by C_1 when $n_3 = 1$ and C_1, C_2 when $n_3 = 2$ where $\|\overrightarrow{CC_1}\|_2 < \|\overrightarrow{CC_2}\|_2$ (see Figure 17).

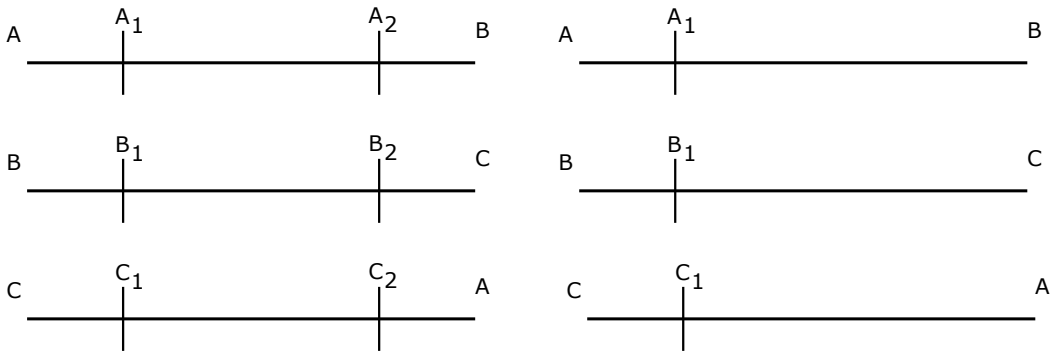

Fig. 17. Intersections between the sides of the triangle and the cercle $\mathcal{C}(P,R)$.

Each $n_i, i = 1, 2, 3$, can take 3 values (0, 1, or 2) and therefore the triple (n_1, n_2, n_3) can take 27 possible different values. We associate to base 3 number $n_3n_2n_1$ its decimal equivalent, denoted by `Code` in what follows, and given by $\texttt{Code} = 9n_3 + 3n_2 + n_1$. To identify the shape of the intersection between the triangle and the disk, we branch according to the value of variable `Code` and for a given value of `Code`, we consider all possible shapes for the intersection of the triangle and the disk. Obviously, for all values of `Code` corresponding to different permutations of the same triple (n_1, n_2, n_3) the different possible shapes for the intersection are the same. We now discuss for

every possible value of variable Code how to compute the area of the intersection.

- **Code=0.**

In this case, there is no intersection between the circle $C(P, d)$ and the triangle. There are three possible shapes for the intersection represented in Figure 18: (a) the triangle is contained in the disk, (b) the disk is outside the triangle, and (c) the disk is inside the triangle.

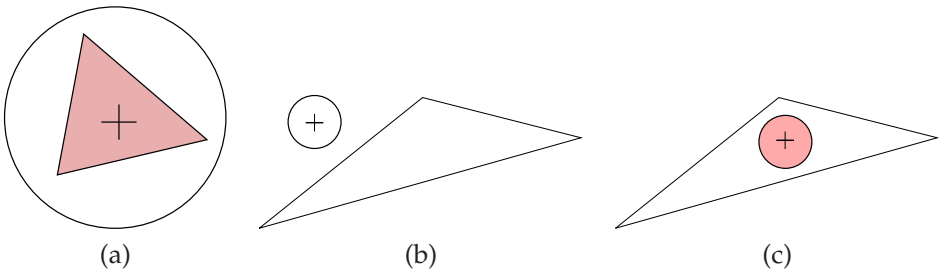

Fig. 18. Possible shapes for the intersection when Code = 0.

To know which of subcases (a), (b), or (c) we are in, we compute

$$d_A = ||\overrightarrow{PA}||_2, d_B = ||\overrightarrow{PB}||_2, d_C = ||\overrightarrow{PC}||_2. \tag{35}$$

In what follows, we denote by

$$T(A, B, C) = \frac{1}{2}\left|x_A(y_B - y_C) + x_B(y_C - y_A) + x_C(y_A - y_B)\right| \tag{36}$$

the area of triangle ABC. If $d_A < R, d_B < R$, and $d_C < R$ then the area is $T(A, B, C)$. Otherwise either P is inside the triangle and the area is πR^2 or it is outside and the area is null. To know if P is inside or outside the triangle, we compute the crossing number for P and the triangle and the minimal distance $d_{\min}$ from P to the border of the triangle. Knowing that the crossing number is odd if and only if P belongs to the relative interior of the triangle, P is inside the triangle if and only if $d_{\min} = 0$ or the crossing number is odd.

• **Code** $= 1, 3, 9$, corresponding to $(n_1, n_2, n_3) \in \{(1, 0, 0), (0, 1, 0), (0, 0, 1)\}$, i.e., one side of the triangle has a single intersection with the circle and the remaining two have no intersection.

There are two possible shapes for the intersection represented in Figure 19-(a),(b).

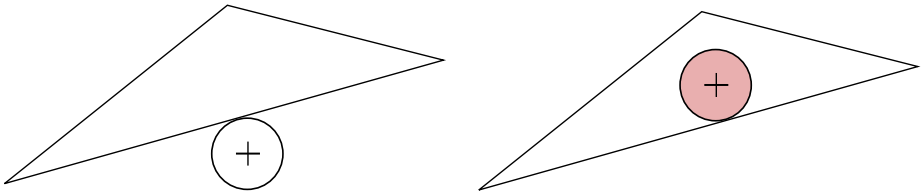

Fig. 19. Possible shapes for the intersection when Code=1, 3, 9.

In case (a), P is outside the triangle and the area is null while in case (b), P is inside the triangle and the area is πR^2. We have already seen how to differentiate these two cases on the basis of $d_{\min}$ and of the crossing number.

- **Code=2, 6, 18**, obtained when (n_1, n_2, n_3) is $(2, 0, 0), (0, 2, 0), (0, 0, 2)$, respectively.

There are two possible shapes for the intersection represented in Figure 20-(a), (b).

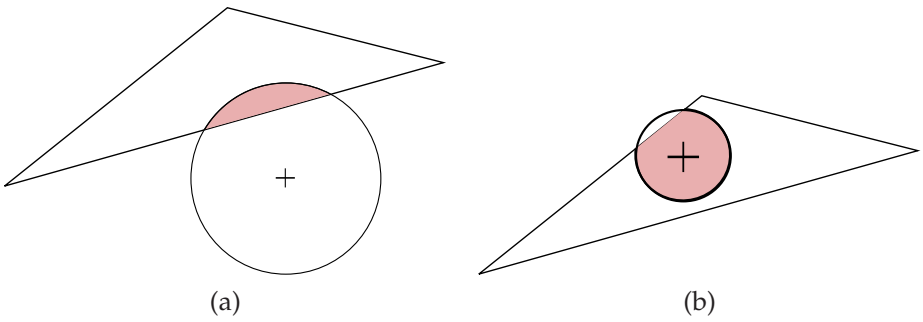

Fig. 20. Possible shapes for the intersection when Code=2, 6, 18.

In each case, the intersection is a lens, of area $\leq 0.5\pi R^2$ in case (a) and $\geq 0.5\pi R^2$ in case (b). Let us recall how to compute analytically these areas.

Consider a chord $\overline{AB}$ of circle $\mathcal{C}(P, R)$. It defines two lenses represented in Figure 21-(a), (b).

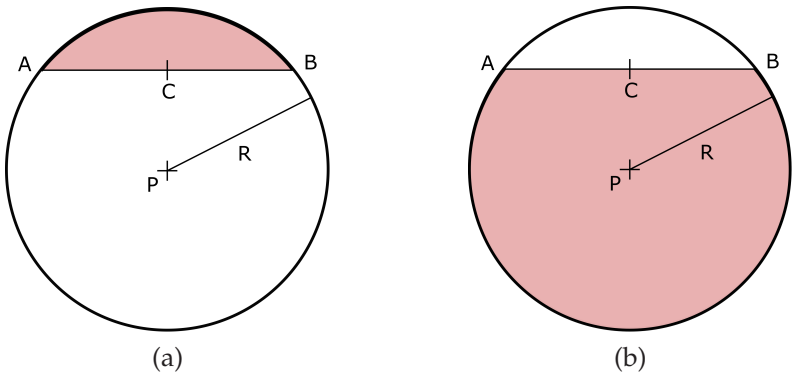

Fig. 21. Two lenses given by chord $\overline{AB}$ in a disk of center P and radius R.

Let us recall the formula for the area $L(P, R, A, B)$ of the lens in case (a) (in case (b), it is given by $\pi R^2 - L(P, R, A, B)$). In case (a), let $C = \frac{A+B}{2}$ and let $c\theta$ be the cosine of acute angle $\angle CPB$ given by

$$c\theta = \frac{\langle \overrightarrow{PC}, \overrightarrow{PB} \rangle}{R||\overrightarrow{PC}||_2}.$$

Then

$$L(P, R, A, B) = R^2 \arccos(c\theta) - R^2 c\theta \sqrt{1 - c\theta^2}.$$

With this notation, when `Code` $= 18$, i.e., when $(n_3, n_2, n_1) = (2, 0, 0)$, in case (a), the area of the intersection is given by $L(P, R, C_1, C_2)$. The cases where `Code` $= 2, 6$ are dealt with by appropriate permutation of the intersection points.

- **Code=4, 10, 12,** obtained when (n_3, n_2, n_1) is $(0, 1, 1)$, $(1, 0, 1)$, $(1, 1, 0)$.

The possible shapes for the intersection between the triangle and disk are given in Figure 22. We discuss how to compute the intersection area when $(n_3, n_2, n_1) = (1, 1, 0)$, i.e., `Code = 12`. The cases `Code=4,10`, are similar, obtained by appropriate permutation of A, B, C and the intersection points.

Let us first discuss on how to distinguish between cases (e) and (f). In theses cases, the intersection is the union of a triangle and a lens. In case (e) the lens has area lower than or equal to $0.5\pi R^2$ while in case (f) the lens has area greater than $0.5\pi R^2$. In case (f), P and C are in two different half-spaces separated by line (C_1B_1). More precisely, let us introduce the function

`[out]=Are_In_Same_Half_Space(C,P,A,B)`

with inputs four points C, P, A, B in $\mathbb{R}^2$ which returns 0 if C and P are in two different half-spaces separated by line (AB) and 1 otherwise. Clearly, `out` is 1 if and only if

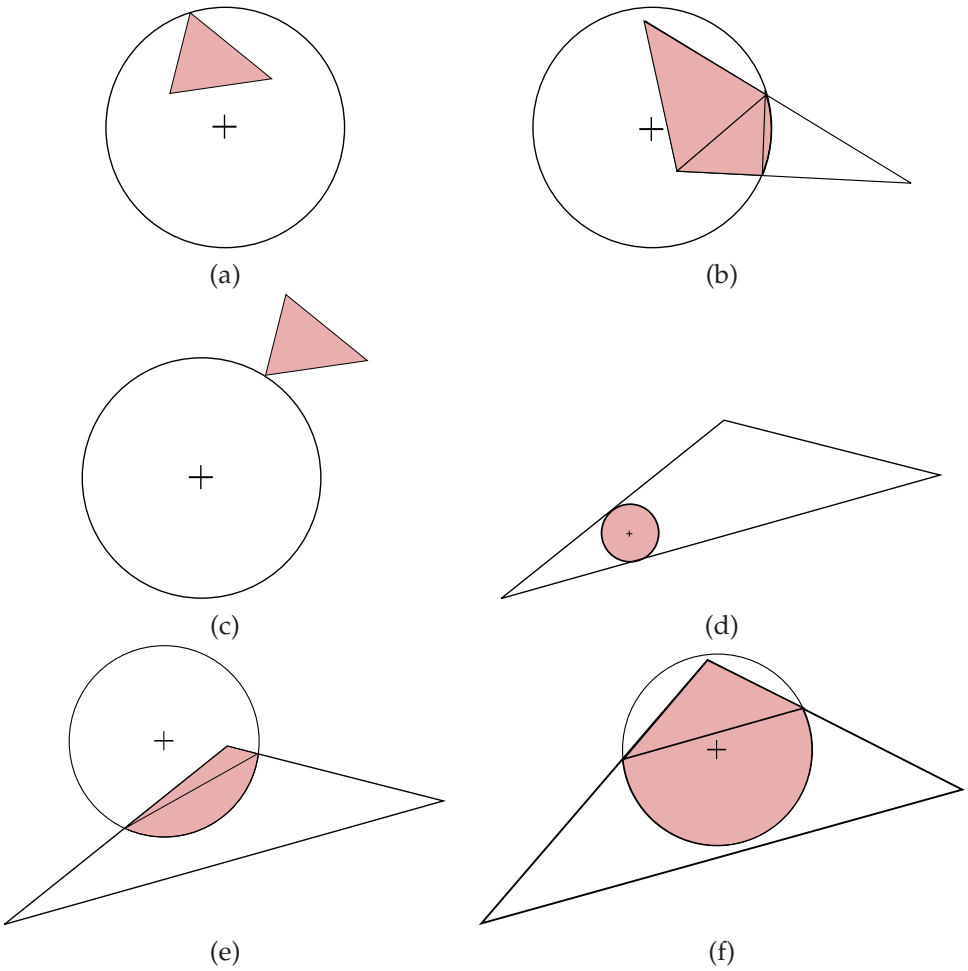

Fig. 22. Possible shapes for the intersection when `Code=4, 10, or 12`.

($x_A = x_B$ and $(x_A - x_P)(x_A - x_C) \geq 0$) or ($x_A \neq x_B$ and $(y_P - D_{A,B}(x_P))(y_C - D_{A,B}(x_C)) \geq 0$) where

$$D_{A,B}(x) = y_A + \frac{y_B - y_A}{x_B - x_A}(x - x_A).$$

With this notation, the intersection area in cases (a)-(f) is computed with the following pseudo-code where `area` will store the intersection area:

`area=0.`
if $d_A < R$ and $d_B < R$ and $d_C = R$ then `area` $= T(A, B, C)$,
else if $d_A < R$ and $d_B < R$, and $d_C > R$ then `area` $= L(P, R, C_1, B_1) + T(A, B, B_1) + T(A, B_1, C_1)$,

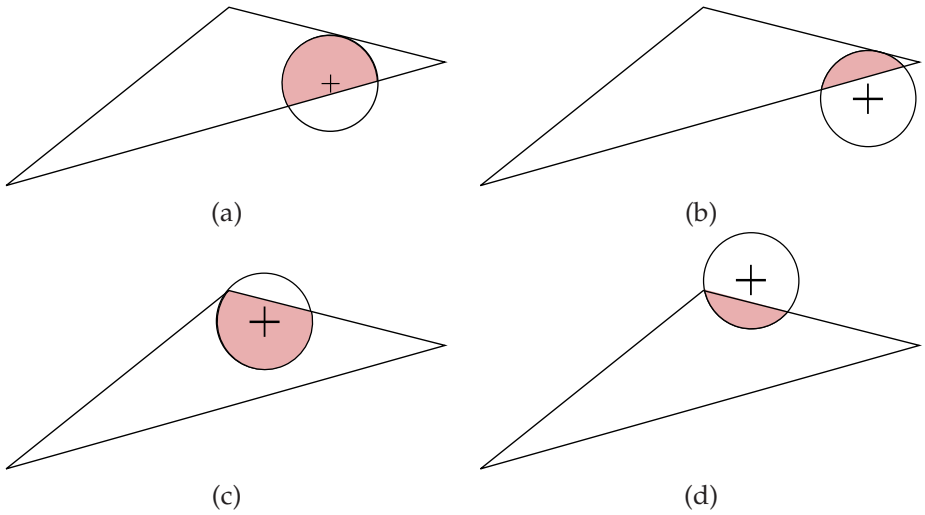

Fig. 23. Possible shapes for the intersection when Code=5, 7, 11, 15, 19, or 21.

else if $d_A > R$ and $d_B > R$ and $d_C > R$ then area = πR^2,
else if $d_A > R$ and $d_B > R$ and $d_C < R$ then
[out]=Are_In_Same_Half_Space(C, P, B_1, C_1),
if out=1 then area= $T(C, B_1, C_1) + L(P, R, B_1, C_1)$,
else area= $T(C, B_1, C_1) + \pi R^2 - L(P, R, B_1, C_1)$,
end if
end if

• **Code=5,7,11,15,19,21,** corresponding to $(n_3, n_2, n_1) \in \{(0,1,2), (0,2,1), (1,0,2), (1,2,0), (2,0,1), (2,1,0)\}$.

The possible shapes for the intersection are represented in Figure 23. It follows that when Code=5 or 11, i.e., when (n_3, n_2, n_1) is $(0,1,2)$ or $(1,0,2)$, the area of the intersection is computed with the following pseudo-code (stored in variable area):

if the crossing number for P and the triangle is odd or $d_{\min} = 0$ then area= $\pi R^2 - L(P, R, A_1, A_2)$
else area=$L(P, R, A_1, A_2)$.
end if

The pseudo-codes when Code=7,15,19,21, are obtained by appropriate permutations of the intersection points.

• **Code=8, 20, 24** obtained when $(n_3, n_2, n_1) = (0,2,2), (2,0,2), (2,2,0)$.

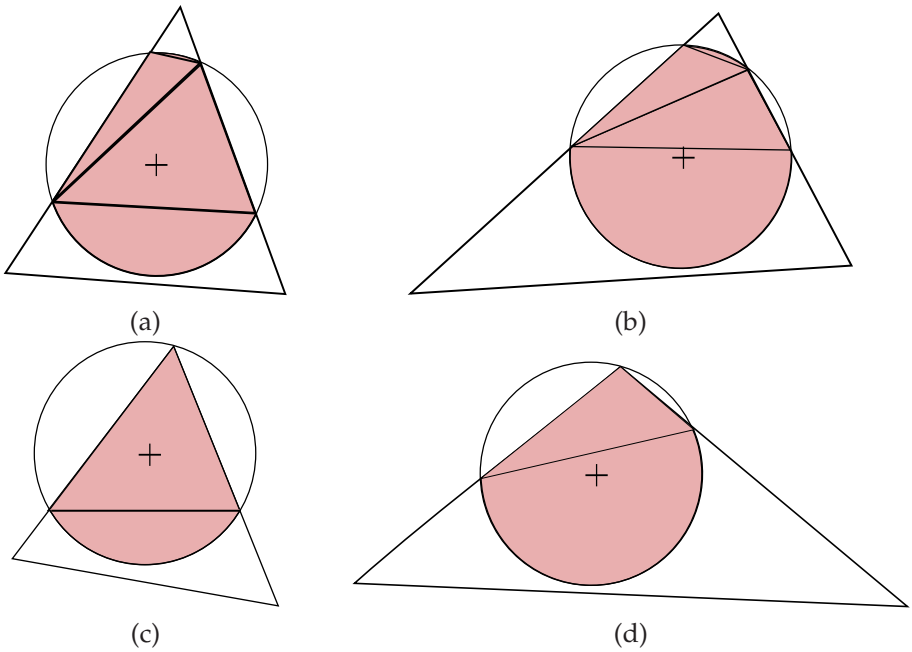

Fig. 24. Possible shapes for the intersection when Code=8, 20, or 24

The possible shapes for the intersection are represented in Figure 24. It follows that when Code=8, i.e., when $(n_3, n_2, n_1) = (0, 2, 2)$ the pseudo-code for computing the area of the intersection (stored in variable area) is the following:

area=0.
[out]=Are_In_Same_Half_Space(B, P, A_1, B_2)
if $d_B > R$
  if out=1
    area= $L(P, R, A_1, B_2) + T(B_1, A_1, B_2) + T(A_2, A_1, B_1) + L(P, R, A_2, B_1)$,
  else
    area= $\pi R^2 - L(P, R, A_1, B_2) + T(B_1, A_1, B_2) + T(A_2, A_1, B_1) + L(P, R, A_2, B_1)$,
  end if
else
  if out=1
    area= $L(P, R, A_1, B_2) + T(B, A_1, B_2)$,
  else
    area= $\pi R^2 - L(P, R, A_1, B_2) + T(B, A_1, B_2)$.
  end if
end if

The pseudo-codes for Code= 20 and 24 are obtained by appropriate permutations of A, B, C, and the intersection points.

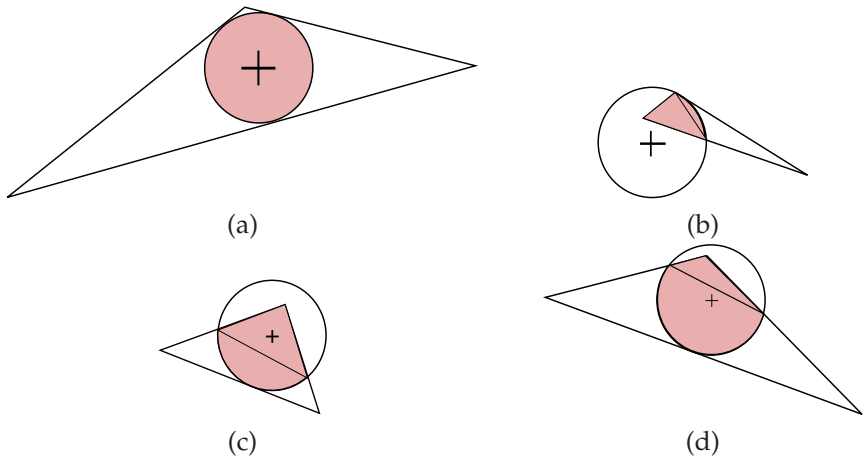

Fig. 25. Possible shapes for the intersection when Code=13.

• **Code=13** obtained when $(n_3, n_2, n_1) = (1, 1, 1)$.

The possible shapes for the intersection are represented in Figure 25. Therefore the pseudo-code for computing the intersection area is the following (where the area of the intersection is stored in variable area):

if $d_A > R$ and $d_B > R$ and $d_C > R$ then area $= \pi R^2$,
else if $d_A = R$ and $d_B > R$ and $d_C < R$ then area $= L(P, R, A, B_1) + T(A, C, B_1)$,
else if $d_A = R$ and $d_C > R$ and $d_B < R$ then area $= L(P, R, A, B_1) + T(A, B, B_1)$,
else if $d_B = R$ and $d_A > R$ and $d_C < R$ then area $= L(P, R, B, C_1) + T(C, B, C_1)$,
else if $d_B = R$ and $d_C > R$ and $d_A < R$ then area $= L(P, R, B, C_1) + T(A, B, C_1)$,
else if $d_C = R$ and $d_B > R$ and $d_A < R$ then area $= L(P, R, C, A_1) + T(C, A, A_1)$,
else if $d_C = R$ and $d_B < R$ and $d_A > R$ then area $= L(P, R, C, A_1) + T(C, B, A_1)$,
else if $d_A < R$ and $d_B > R$ and $d_C > R$ then
  [out]=Are_In_Same_Half_Space(P, A, A_1, C_1),
  if out=1 then area= $L(P, R, A_1, C_1) + T(A, A_1, C_1)$,
  else area= $\pi R^2 - L(P, R, A_1, C_1) + T(A, A_1, C_1)$
  end if
else if $d_B < R$ and $d_A > R$ and $d_C > R$ then
  [out]=Are_In_Same_Half_Space(P, B, A_1, B_1),
  if out=1 then area= $L(P, R, A_1, B_1) + T(B, A_1, B_1)$,
  else area= $\pi R^2 - L(P, R, A_1, B_1) + T(B, A_1, B_1)$
  end if
else if $d_C < R$ and $d_A > R$ and $d_B > R$ then
  [out]=Are_In_Same_Half_Space(P, C, C_1, B_1),

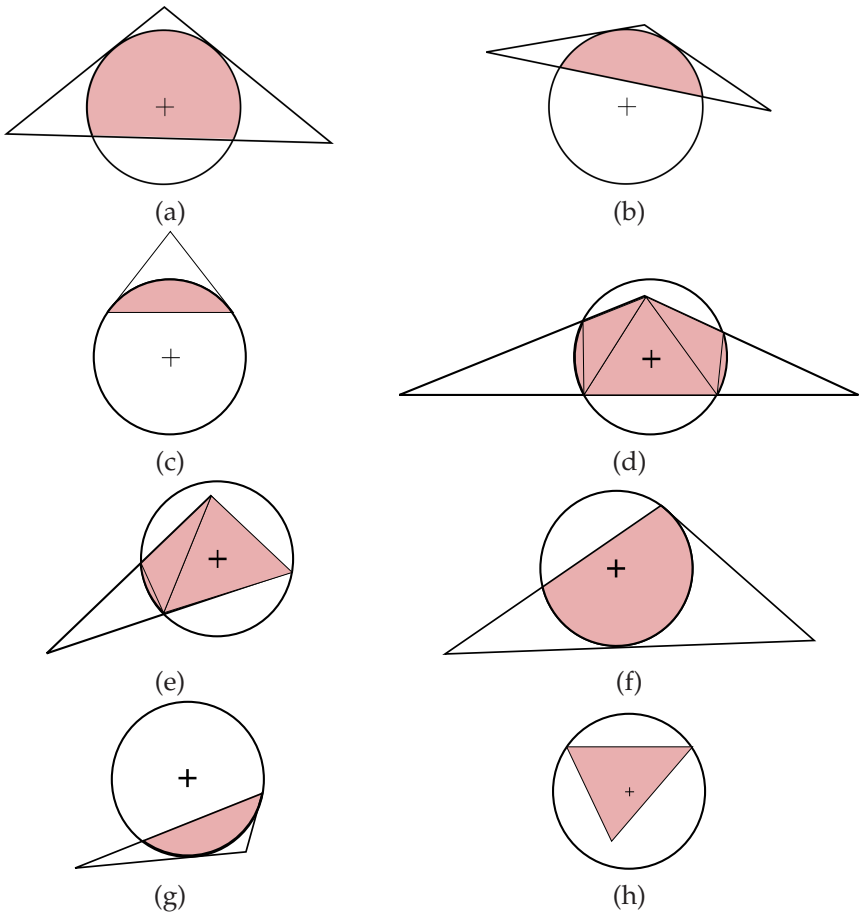

Fig. 26. Possible shapes for the intersection when Code=14, 16 or 22.

if out=1 then area= $L(P, R, C_1, B_1) + T(C, C_1, B_1)$,
else area= $\pi R^2 - L(P, R, C_1, B_1) + T(C, C_1, B_1)$,
end if
end if

• **Code=14, 16, 22** obtained when $(n_3, n_2, n_1) = (1, 1, 2), (1, 2, 1), (2, 1, 1)$. The possible shapes for the intersection are represented in Figure 26. It follows that the pseudo-code for computing the area of the intersection (stored in variable area) when Code= 14 is the following:

area= 0.
if $d_A > R$ and $d_B > R$ and $d_C > R$ then
[out]=Are_In_Same_Half_Space(P, C, A_1, A_2),

if out=1 then area= $\pi R^2 - L(P, R, A_1, A_2)$,
else area= $L(P, R, A_1, A_2)$,
end if
else if $d_A > R$ and $d_C < R$ and $d_B > R$ then
area= $L(P, R, A_1, C_1) + L(P, R, B_1, A_2) + T(C, C_1, A_1) + T(C, A_1, A_2) + T(C, B_1, A_2)$,
else if $d_B = R$ and $d_A = R$ and $d_C < R$ then area= $T(A, B, C)$,
else if $d_B = R$ and $d_A = R$ and $d_C > R$ then area= $L(P, R, A, B)$,
else if $d_B = R$
if $d_C < R$ then area= $L(P, R, A_1, C_1) + T(B, C, A_1) + T(C, A_1, C_1)$,
else
[out]=Are_In_Same_Half_Space(P, C, B, A),
if out= 1 then area= $\pi R^2 - L(P, R, A_1, B)$,
else area= $L(P, R, A_1, B)$,
end if
end if
else if $d_A = R$
if $d_C < R$ then area= $L(P, R, B_1, A_2) + T(A, C, A_2) + T(C, B_1, A_2)$,
else
[out]=Are_In_Same_Half_Space(P, C, B, A),
if out= 1 then area= $\pi R^2 - L(P, R, A, A_2)$,
else area= $L(P, R, A, A_2)$,
end if
end if
end if

The pseudo-codes when Code= 16 and 22 are obtained by appropriate permutation of A, B, C, and the intersection points.

• **Code=17, 23, 25** obtained when $(n_3, n_2, n_1) = (1, 2, 2), (2, 1, 2), (2, 2, 1)$. The possible shapes for the intersection are represented in Figure 27. It follows that when Code= 17 the pseudo-code for computing the area of the intersection is the following:

if $d_A > R$ and $d_B > R$ and $d_C > R$ then
[out]=Are_In_Same_Half_Space(P, B, B_2, A_1),
if out= 1 then area= $L(P, R, B_1, A_2) + L(P, R, A_1, B_2) + T(B_1, A_2, B_2) + T(A_1, A_2, B_2)$,
else area= $L(P, R, B_1, A_2) + \pi R^2 - L(P, R, A_1, B_2) + T(B_1, A_2, B_2) + T(A_1, A_2, B_2)$,
end if
else if $d_A = R$ and $d_B = R$ then area= $L(P, R, A, B_2) + T(A, B, B_2)$,
else if $d_C = R$ and $d_B = R$ then area= $L(P, R, C, A_1) + T(B, C, A_1)$,
else if $d_B = R$ then
[out]=Are_In_Same_Half_Space(P, B, B_2, A_1),

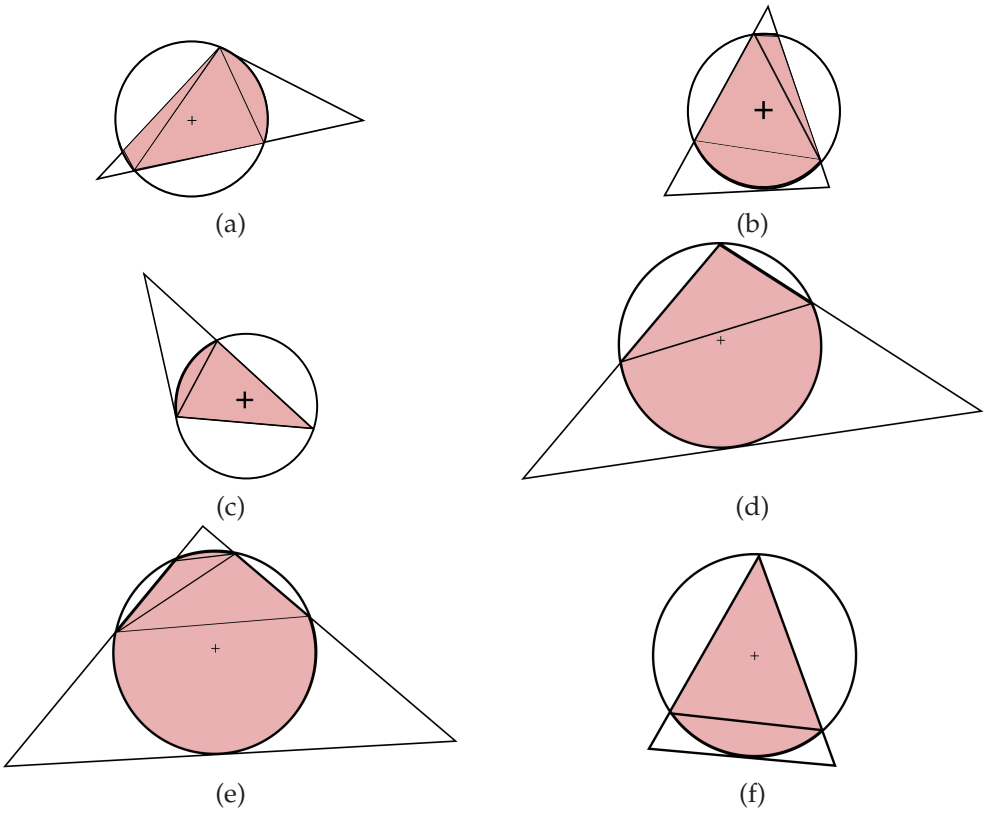

Fig. 27. Possible shapes for the intersection when Code=17, 23 or 25.

if out= 1 then area= $L(P, R, A_1, B_2) + T(A_1, B, B_2)$,
else area= $\pi R^2 - L(P, R, A_1, B_2) + T(A_1, B, B_2)$,
end if
else if $d_A = R$ then

$$\texttt{area} = L(P, R, A_1, B_2) + L(P, R, A_2, B_1) + T(A, B_2, B_1) + T(A, B_1, A_2),$$

else if $d_C = R$ then

$$\texttt{area} = L(P, R, A_1, C) + L(P, R, A_2, B_1) + T(C, A_2, A_1) + T(C, B_1, A_2),$$

end if.

The pseudo-codes when Code= 23 and 25 are obtained by appropriate permutation of A, B, C, and the intersection points.

• **Code=26** corresponding to $(n_3, n_2, n_1) = (2, 2, 2)$. The possible shapes for the intersection are represented in Figure 28. From this figure we obtain the following

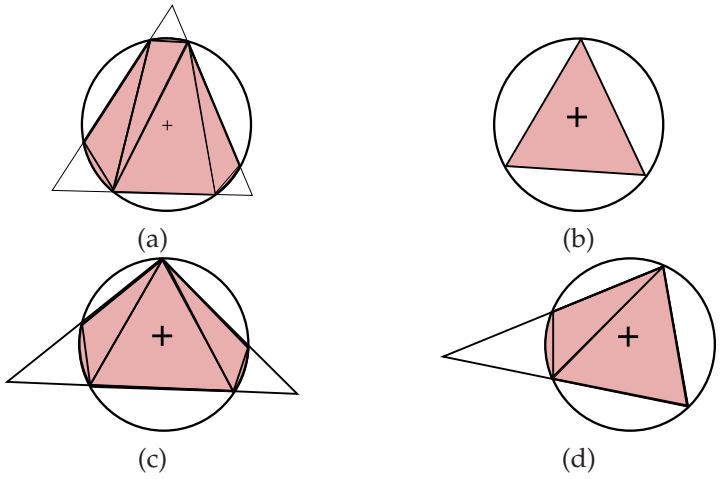

Fig. 28. Possible shapes for the intersection when Code=26.

pseudo-code to compute the intersection area:

if $d_A > R$ and $d_B > R$ and $d_C > R$ then

$$\texttt{area} = L(P,R,A_2,B_1) + L(P,R,B_2,C_1) + L(P,R,A_1,C_2) + T(A_2,B_1,A_1) + T(A_1,C_2,B_1) + T(C_2,B_1,B_2) + T(B_2,C_1,C_2),$$

else if $d_A = R$ and $d_B = R$ and $d_C = R$ then $\texttt{area} = T(A,B,C)$,

else if $d_A = R$ and $d_B = R$ and $d_C > R$ then

$$\texttt{area} = L(P,R,C_1,B_2) + T(A,B,C_1) + T(B,C_1,B_2),$$

else if $d_A = R$ and $d_C = R$ and $d_B > R$ then

$$\texttt{area} = L(P,R,A_2,B_1) + T(A,C,A_2) + T(C,A_2,B_1),$$

else if $d_B = R$ and $d_C = R$ and $d_A > R$ then

$$\texttt{area} = L(P,R,A_1,C_2) + T(C,B,A_1) + T(C,A_1,C_2),$$

else if $d_A = R$ and $d_B > R$ and $d_C > R$ then

$$\texttt{area} = L(P,R,A_2,B_1) + L(P,R,B_2,C_1) + T(A,A_2,B_1) + T(A,B_1,B_2) + T(A,C_1,B_2),$$

else if $d_B = R$ and $d_A > R$ and $d_C > R$ then

$$\texttt{area} = L(P,R,A_1,C_2) + L(P,R,C_1,B_2) + T(A,B,C_2) + T(B,C_1,C_2) + T(B,B_2,C_1),$$

else if $d_C = R$ and $d_A > R$ and $d_B > R$ then

$$\texttt{area} = L(P,R,A_1,C_2) + L(P,R,A_2,B_1) + T(C,C_2,A_1) + T(C,A_1,A_2) + T(C,B_1,A_2),$$

end if

6. The library: main functions and examples

The Matlab library, available at `https://github.com/vguigues/Areas_Library`, computes the density of the distance D to a random variable uniformly distributed in some sets and the area of intersection of disks and balls with those sets.

All necessary files to run the functions of the library are in folders `Areas_Library` and its subfolder `Examples` which contains files to run the main functions on examples. No external library is needed. We implemented all functions of these folders except function `polygon_triangulate` which computes a triangulation of a polygone. This function, which can be found in folder `Areas_Library`, is the Matlab version by John Burkardt of the original C version by Joseph ORourke [1998].

If the folder `Areas_Libary` is copied in folder `C:\Users\user_name`, before using the library, update the path in Matlab with commands:

```
addpath 'C:\Users\user_name\Areas_Libary'
addpath 'C:\Users\user_name\Areas_Libary\Examples'
```

The next section shows how to use the functions of the library from Sections 5.2 and 5.3 using the files of examples that can be found in folder `Areas_Libary\Examples`.

6.1. *Density of the distance to a random variable uniformly distributed in a polygone*

The function to compute the density of the distance from a point $P \in \mathbb{R}^2$ to a random variable uniformly distributed in a polygone S is:

```
[d,time,dmin,dmax]=density_polyhedron(S,P,Np,algo)
```

where input variables are:

- `algo`: a char indicating the algorithm used. It can take two values 'g' and 't1'. When algo='g', the algorithm from Section 5.2 based on Green's theorem is used. When algo='t1' the algorithm from Section 5.3 is used to compute the intersection areas of disks and the polygone.
- `Np`: the number of discretization points: the density is computed at `Np` equally spaced points $x_1, x_2, \ldots, x_{\texttt{Np}}$ from the support of the random variable distance.
- P: point P as explained above.
- $S = [S_1; S_2; S_3; \ldots; S_n; S_1]$: the polyhedron where n is the number of vertices and $S_1, S_2, S_3, \ldots, S_n$ are the successive vertices of the polyhedron. Observe that the first point S_1 is repeated. When algo='g', when travelling on the boundary of S from S_1 to S_2, then from S_2 to S_3 and so on until the last line segment $\overline{S_n S_1}$, one always has the relative interior of S to the left. When

algo='t1' this restriction does not apply: if algo='t1', when travelling on of S from S_1 to S_2, then from S_2 to S_3 and so on until the last line segment $\overline{S_nS_1}$, one can either have the relative interior of S to the left or to the right.

Output variables of function `density_polyhedron` are:

- `d`: a vector of size `Np` where $d(i)$ is the estimation of the density of the random variable at x_i.
- `time` is the time required to compute d.
- `]dmin,dmax[` is the support of the random variable meaning that `dmax` is the maximal distance between P and the boundary of the polygone. If P is inside the polygone then `dmin`$= 0$ and if P is outside the polygone then `dmin` is the minimal distance from P to the boundary of the polygone.

We illustrate the use of this function on several examples written in folder `'Areas_Libary\Examples'`.

We start with an example written in file `drectex.m` of folder `Examples` where S is a rectangle with side lengths L and αL with $0 < \alpha < 1$ and P is the center of the rectangle. For this example, the density of the distance from P to a random variable uniformly distributed in S is known in closed form and is given in Stewart and Zhang [2013]. Therefore, this example allows us to test the implementation of function `density_polyhedron` comparing output `d` of this function when algo='g', 't1' with the theoretical values given in Stewart and Zhang [2013].

The function corresponding to this example is

`[dG,dT1,d,ErrG,ErrT1]=drectex(Np,`α`,`L`)`.

The input variables are parameters `Np`, α, L, given above and the outputs are the following:

- `dG` and `dT1` are vectors of size `Np` and `dG`(i) (resp. `dT1`(i)) is the value of the density at x_i, $i = 1, \ldots,$`Np`, computed calling function `density_polyhedron` with variable algo='g' (resp. calling function `density_polyhedron` with variable algo='t1'). Recall that x_i, $i = 1, \ldots, x_{\texttt{Np}}$ are `Np` equally spaced points in `]dmin,dmax[`.
- `d` is a vector of size `Np`: `d`(i) is the exact value of the density at x_i computed using the analytic formulas given in Stewart and Zhang [2013].
- `ErrG` is the maximal error when algo='g', i.e., `ErrG`$= \max_{i=1,\ldots,\texttt{Np}} |\texttt{dG}(i) - \texttt{d}(i)|$.
- `ErrT1` is the maximal error when algo='t1', i.e., `ErrT1`$= \max_{i=1,\ldots,\texttt{Np}} |\texttt{dT1}(i) - \texttt{d}(i)|$.

On top of that, the function plots vectors `dG`, `dT1`, and `d`. For instance, running

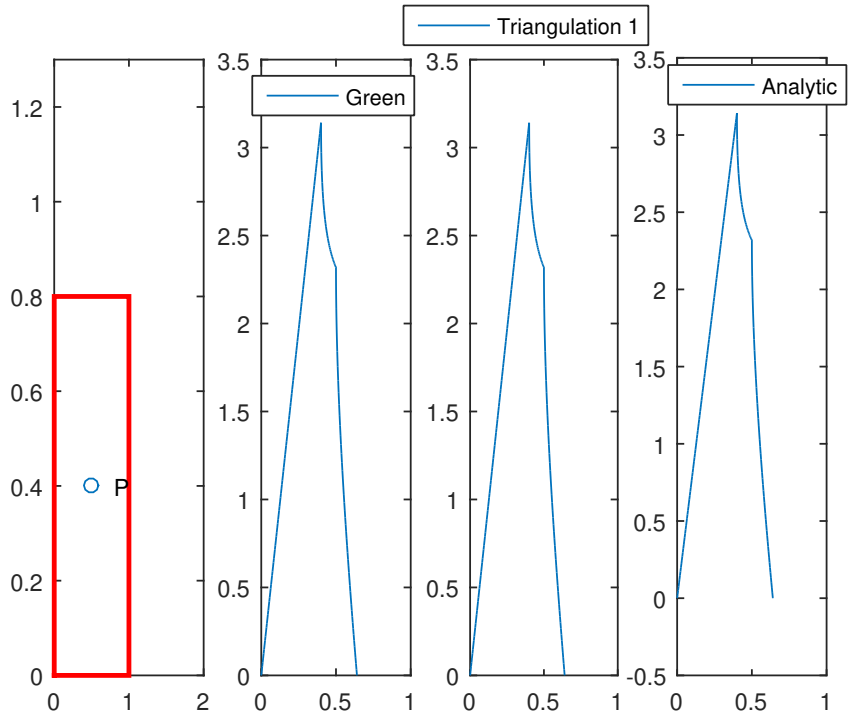

Fig. 29. Plots produced calling `[dG,dT1,d,ErrG,ErrT1]=drectex(Np,`α`,`L`)`. From left to right: rectangle S and P, plot of dG, plot of dT1, and plot of d.

```
[dG,dT1,d,ErrG,ErrT1]=drectex(1,0.8,1)
```

the plots of Figure 29 are displayed. On this Figure, from left to right, the first plot represents rectangle $S = [(0,0);(1,0);(1,0.8);(0,0.8);(0,0)]$ and $P = (0.5, 0.4)$, the second plot represents the density of D obtained using the algorithm from Section 5.2, the third plot is the density of D obtained using the algorithm from Section 5.3, while the last plot is the graph of the true density of D.

In this example, the densities are computed with the following Matlab code:

```
P1=[0,0]; P2=[L,0]; P3=[L,alpha×L]; P4=[0,alpha×L];
P=[L/2,alpha×L/2];
S=[P1;P2;P3;P4;P1];
[dG,timeg,dmin,dmax]=density_polyhedron(S,P,Nb,'g');
[dT1,timet1,dmin,dmax]=density_polyhedron(S,P,Nb,'t1');
ErrG=max(abs(d-dG)); ErrT1=max(abs(d-dT1)).
```

Np	ErrG	ErrT1
10 000	0.017	0.023
20 000	0.010	0.020
50 000	0.007	0.04
100 000	0.004	0.03

Table 1. Maximal error obtained with the algorithms from Sections 5.2 and 5.3 to compute the density of D (D being the distance from the center of a rectangle with side lengths 1 and 0.8 to a random variable with uniform distribution in this rectangle) at Np discretization points.

To check the implementations of the algorithm from Sections 5.2 and 5.3, we now report in Table 1 the values of ErrG and ErrT1 for several values of the number Np of discretization points, namely when Np varies in the set $\{10\,000, 20\,000, 50\,000, 100\,000\}$.

In all cases the maximal error is very small which shows that both algorithms correctly compute the Np areas of intersection of the disks and polygone of this example.[b] We also observe that the approximations are slightly better with the algorithm from Section 5.2 and, as expected, the maximal error decreases with Np for this algorithm. This is not the case for the other, probably due to roundoff errors.

We now compare the algorithms on other examples coded in Matlab file dpolyex.m. More precisely, we consider three polyhedra (a triangle, a rectangle, and an arbitrary polygone) and in each case a point P inside the polygone and a point P outside. For these 6 examples the Matlab codes are the following.

• $S = [(1,1);(10,1);(3,4);(1,1)]$ is a triangle and $P = [5;0]$ is outside this triangle. In Figure 30, S and P are represented in the left plot while the corresponding density of D is represented in the right plot. This density is obtained with the following Matlab code:

```
P1=[1,1]; P2=[10,1]; P3=[3,4]; S=[P1;P2;P3;P1]; P=[5,0];
[dT1,timeT1,dminT1,dmaxT1]=density_polyhedron(S,P,Np,algo);
```

where algo='g' or 't1'.

• $S = [(1,1);(10,1);(3,4);(1,1)]$ is a triangle and $P = [4;2]$ is inside this triangle. In Figure 31, S and P are represented in the left plot while the corresponding density of D is represented in the right plot. This density is obtained with the following Matlab code:

[b]To approximate the density at Np points, we need to compute the cumulative distribution function at Np points and therefore when Np= 100 000, the algorithms are called 100 000 times each to compute 100 000 areas.

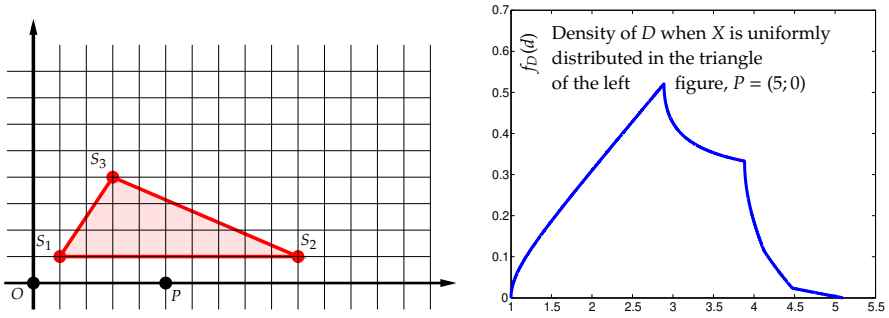

Fig. 30. Density of D when X is uniformly distributed in a triangle.

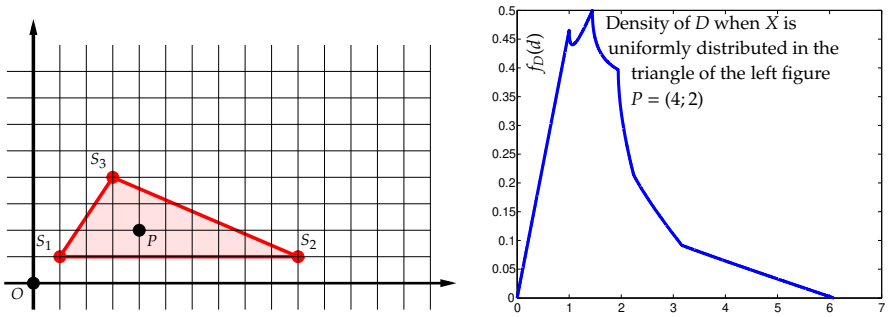

Fig. 31. Density of D when X is uniformly distributed in a triangle.

```
P1=[1,1]; P2=[10,1]; P3=[3,4]; S=[P1;P2;P3;P1];;P=[4,2];
[dT2,timeT2,dminT2,dmaxT2]=density_polyhedron(S,P,Np,algo);
```

where algo='g' or 't1'.

• $S = [(3,3);(12,3);(12,7);(3,7);(3,3)]$ is a rectangle and $P = [1;1]$ is outside this rectangle. In Figure 32, S and P are represented in the left plot while the corresponding density of D is represented in the right plot. This density is obtained with the following Matlab code:

```
P1=[3,3]; P2=[12,3]; P3=[12,7]; P4=[3,7]; S=[P1;P2;P3;P4;P1];
P=[1,1];
[dR1,timeR1,dminR1,dmaxR1]=density_polyhedron(S,P,Np,algo);
```

where algo='g' or 't1'.

• $S = [(3,3);(12,3);(12,7);(3,7);(3,3)]$ is a rectangle and $P = [6;5]$ is inside this

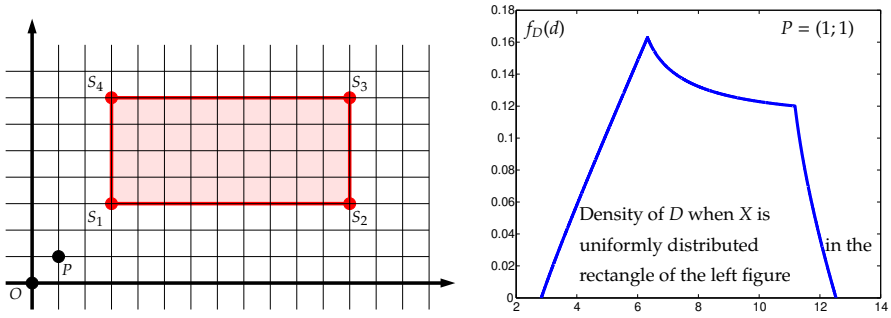

Fig. 32. Density of D when X is uniformly distributed in a rectangle.

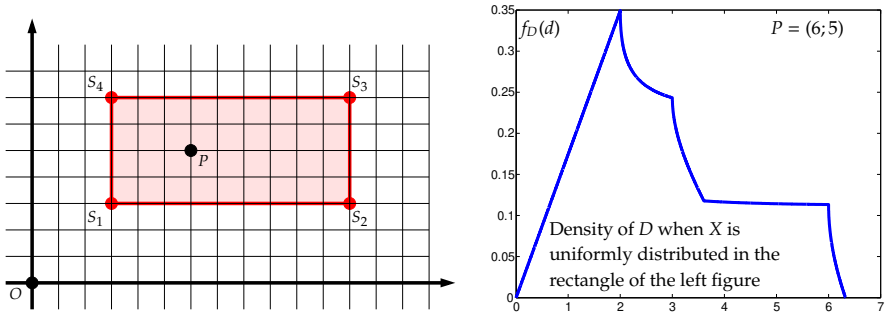

Fig. 33. Density of D when X is uniformly distributed in a rectangle.

rectangle. In Figure 33, S and P are represented in the left plot while the corresponding density of D is represented in the right plot. This density is obtained with the following Matlab code:

```
P1=[3,3]; P2=[12,3]; P3=[12,7]; P4=[3,7]; S=[P1;P2;P3;P4;P1];
P=[6,5];
[dR2,timeR2,dminR2,dmaxR2]=density_polyhedron(S,P,Np,algo);
```

where algo='g' or 't1'.

• $S = [(1,1);(3,1);(5,2);(7,1);(8,3);(6,3);(7,6);(4,5);(1,3);(2,2);(1,1)]$ is a polygone and $P = [4;0]$ is outside this polygone. In Figure 34, S and P are represented in the left plot while the corresponding density of D is represented in the right plot. This density is obtained with the following Matlab code:

```
P1=[1,1]; P2=[3,1]; P3=[5,2]; P4=[7,1]; P5=[8,3];
P6=[6,3]; P7=[7,6]; P8=[4,5]; P9=[1,3]; P10=[2,2];
S=[P1;P2;P3;P4;P5;P6;P7;P8;P9;P10;P1];
```

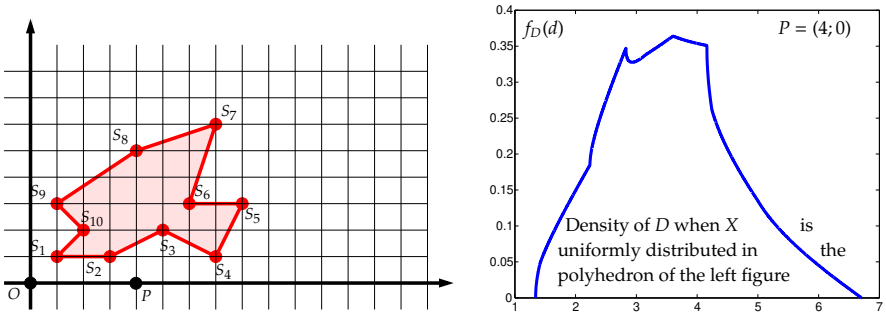

Fig. 34. Density of D when X is uniformly distributed in a polygone.

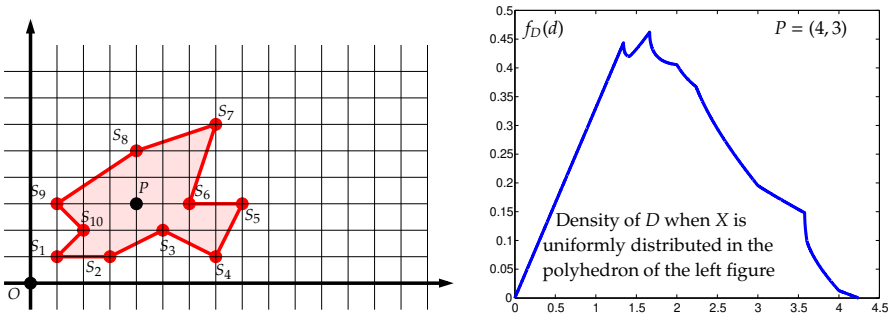

Fig. 35. Density of D when X is uniformly distributed in a polygone.

```
P=[4,0];
[dP1,timeP1,dminP1,dmaxP1]=density_polyhedron(S,P,Np,algo);
```

where algo='g' or 't1'.

• $S = [(1,1);(3,1);(5,2);(7,1);(8,3);(6,3);(7,6);(4,5);(1,3);(2,2);(1,1)]$ is a polygone and $P = [4;3]$ is inside this polygone. In Figure 35, S and P are represented in the left plot while the corresponding density of D is represented in the right plot. This density is obtained with the following Matlab code:

```
P1=[1,1]; P2=[3,1]; P3=[5,2]; P4=[7,1]; P5=[8,3];
P6=[6,3]; P7=[7,6]; P8=[4,5]; P9=[1,3]; P10=[2,2];
S=[P1;P2;P3;P4;P5;P6;P7;P8;P9;P10;P1];
P=[4,3];
[dP2,timeP2,dminP2,dmaxP2]=density_polyhedron(S,P,Np,algo);
```

where algo='g' or 't1'.

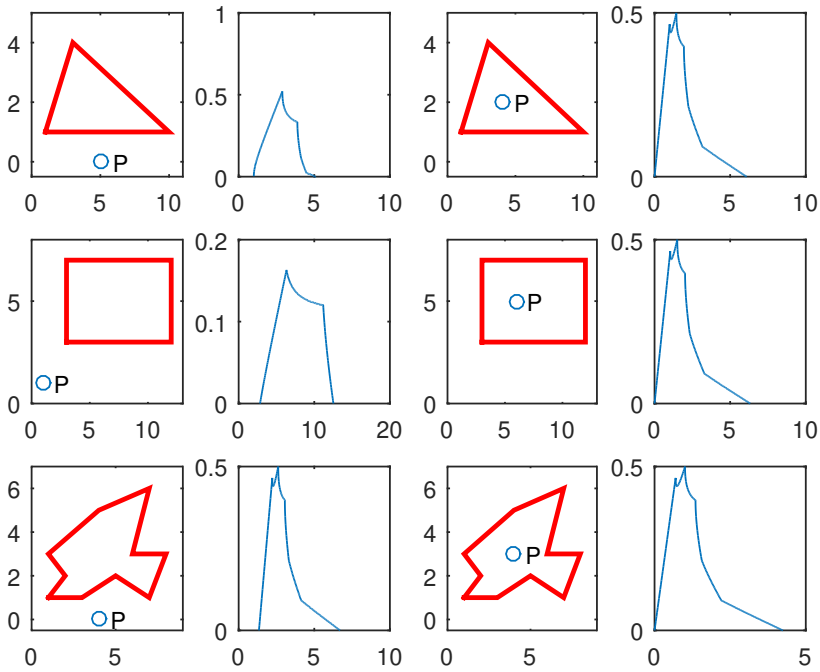

Fig. 36. Plots produced calling `[dT1,dT2,dR1,dR2,dP1,dP2]=dpolyex(10000,'g')`. In red, polygones S with the corresponding densities of D on their right (plots of dT1, dT2 on top, dR1, dR2 in the middle, and dP1, dP2 at the bottom).

Command

```
[dT1,dT2,dR1,dR2,dP1,dP2]=dpolyex(10000,'g')
```

will run the code above to compute `dT1,dT2,dR1,dR2,dP1,dP2` with algo='g', `Np`= 10 000, and will produce Figure 36 which represents polygones S above and the corresponding densities of D on their right.

Command

```
[dT1,dT2,dR1,dR2,dP1,dP2]=dpolyex(Np,'t1')
```

does the same with algo='t1'.

Let $f_G(x_i)$ (resp. $f_T(x_i)$) be the approximation of the density computed by the

algorithm from Section 5.2 based on Green's theorem (resp. the algorithm from Section 5.3, based on a triangulation of the polygone) at x_i. The maximal errors $\max_{i=1,\ldots,\texttt{Np}} |f_G(x_i) - f_T(x_i)|$ were 5.7×10^{-10} for S, P given in Figure 30, 4.1×10^{-8} for S, P given in Figure 31, 8.6×10^{-10} for S, P given in Figure 32, 4.5×10^{-10} for S, P given in Figure 33, 2.3×10^{-5} for S, P given in Figure 34, and 1.7×10^{-9} for S, P given in Figure 35.

The fact that these errors are very small is an indication that both algorithms were correctly implemented.

6.2. *Area of the intersection of a disk and a polygone*

The area of the intersection of polygone

$$S = [S_1; S_2; \ldots; S_n; S_1]$$

(in Matlab notation) and the disk of center $P \in \mathbb{R}^2$ and radius d is computed as follows with the library:

```
[Crossing_Number,AreaP,dmin,dmax]=polyhedron(S,P,n)
[area]=area_intersection_disk_polygone(S,P,d,n,Crossing_Number,AreaP,algo)
```

where output `area` of function `area_intersection_disk_polygone` is the area of the intersection and the outputs of the first function `polyhedron` are:

- `Crossing_Number`: the crossing number for S and P;
- `AreaP`: the area of polygone S;
- `dmin` (resp. `dmax`): the minimal (resp. maximal) distance from P to the border of the polygone.

When algo='g' (resp. 't') the area of the intersection is computed with the algorithm described in Section 5.2 (resp. the algorithm given in Section 5.3).

We test this function computing the areas of intersection of 350 disks and polygones as well as the mean and maximal time required to compute these areas. The polygones are generated using function

```
[Polygone]=generate_polygone(n,R0)
```

of the library where parameters n and R_0 are described below. This function generates randomly a polygone with $4n$ vertices as follows. We sample $4n$ points taking n points in each orthant with polar angles generated randomly and independently in this orthant and radial coordinates generated randomly and independently in the interval $[0,R_0]$ (we take $R_0 = 1000$ in our experiments). We then sort in ascending order the polar angles of these points. This list defines the successive vertices of a star-shaped (simple) polygone. An example of such a star-shaped polygone with $n = 3$ and $4n = 12$ vertices is given in Figure 37, together with a triangulation of this polygone.

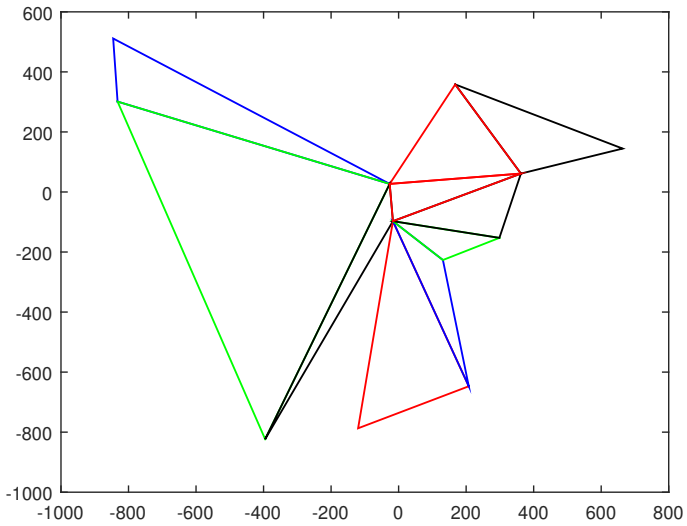

Fig. 37. Star-shaped polygone and a triangulation of this polygone.

The coordinates of the centers of the disks (resp. the radii) are obtained sampling independently from the uniform distribution on the interval $[-100, 100]$ (resp. $[50, 250]$).

For each value of n in the set $\{10, 25, 50, 80, 100, 150, 200\}$ we generate 50 star-shaped polyhedra and disks as explained above and for each polygone and disk, we compute the area of their intersection using both algorithms. The corresponding function of the library is

`[Errmax,ErrMoy,TimeGreen,TimeTr1]=random_areas(`M`)`

where M is the number of Monte-Carlo simulations ($M = 50$ in our experiments) and where the outputs are the following:

- `TimeGreen(k,j)` (resp. `TimeTr1(k,j)` is the time required to compute the intersection area for k-th instance and j-th value of n (for instance for $j = 1$ we have $n = 10$, for $j = 2$, we have $n = 25$) when `area_intersection_disk_polygone` is called with algo='g' (resp. algo='t1');
- `Errmax` and `ErrMoy` are vectors of size 7. `ErrMoy`(j) and `Errmax`(j) are defined respectively by $\frac{1}{50}\sum_{k=1}^{50} |\mathcal{A}_G(k,j) - \mathcal{A}_T(k,j)|$ and $\max_{k=1,\ldots,50} |\mathcal{A}_G(k,j) - \mathcal{A}_T(k,j)|$ where $\mathcal{A}_G(k,j)$ and $\mathcal{A}_T(k,j)$ are the areas of the intersection for k-th Monte-Carlo simulation and j-th value of n computed with respectively algo='g' and algo='t1'.

$4n$	Mean time algo='t1'	Mean Time algo='g'	Max time algo='t1'	Max time algo='g'	ErrMoy	Errmax
40	0.30	0.004	0.34	0.008	9.5×10^{-10}	10^{-8}
100	1.99	0.007	2.54	0.014	2.5×10^{-9}	4.1×10^{-8}
200	8.09	0.012	8.96	0.018	3.8×10^{-9}	3.6×10^{-8}
320	22.57	0.020	34.26	0.036	6.8×10^{-9}	3.4×10^{-8}
400	45.21	0.021	669.76	0.039	1.1×10^{-8}	1.7×10^{-7}
600	128.52	0.033	2 772.5	0.074	1.4×10^{-8}	1.2×10^{-7}
800	369.80	0.043	9 661.8	0.076	1.7×10^{-8}	9.7×10^{-8}

Table 2. Mean and maximal time (in seconds) required to compute the areas of 50 polyhedra with $4n$ vertices for algo='g' and algo='t1'. The last two columns report respectively the mean and maximal errors.

For each value of n, the mean and maximal time (over the 50 instances) required to compute these areas are reported in Table 2. We also report in this table the values of `Errmax` and `ErrMoy`.

We observe that errors are negligible which shows that both algorithms compute the same areas. Moreover, on all instances the algorithm from Section 5.2 computes all areas extremely quickly and much quicker than the algorithm of Section 5.3. For this latter algorithm, both the mean and maximal time required to compute the intersection areas significantly increase with the number of vertices of the polygone.

7. Application to PSHA and extensions

The results of Sections 3, 4, and 5 can be used to determine for the application presented in Section 2 the distribution of the distance between the epicenter in $\mathcal{S}$ and an arbitrary point P when $\mathcal{S}$ is a union of disks, a union of balls, or the boundary of a polyhedron in $\mathbb{R}^3$. For this application, the coordinates of P, of the centers of the disks and of two points on the boundaries of these disks, of the centers of the balls, and of the vertices $S_1, \ldots, S_n$ of the polyhedron are given providing for each point its latitude, its longitude, and its depth measured from the surface of the earth. To apply the computations of the previous sections, we need to choose a Cartesian coordinate system and use the corresponding Cartesian coordinates of these points. These coordinates are given as follows. We take for the positive x-axis the ray OA where O is the center of the earth and A is the point on the surface of the earth with longitude 0 and latitude 0. We take for the positive z-axis the ray OB where O is the center of the earth and B is the north pole. The positive y-axis is chosen correspondingly and corresponds to ray OC where C is the point on the surface of the earth with latitude 0 and longitude 90^o East. Let P be a point at depth d from the surface of the earth with latitude $\varphi \in [0, 90^o]$ (North or South) and longitude $\lambda \in [0, 180^o]$ (East or West). If the latitude is φ North (resp. φ South), we use the notation φN (resp. φS) while if the longitude is λ East (resp. λ West), we use the notation λE (resp. λW). Denoting by R the earth radius, the Cartesian

coordinates of P in the chosen Cartesian coordinate system are

$$\begin{aligned}
&\Big((R-d)\cos\varphi\cos\lambda, (R-d)\cos\varphi\sin\lambda, (R-d)\sin\varphi\Big) \text{ if } P=(R-d,\lambda E,\varphi N),\\
&\Big((R-d)\cos\varphi\cos\lambda, (R-d)\cos\varphi\sin\lambda, -(R-d)\sin\varphi\Big)\text{if } P=(R-d,\lambda E,\varphi S),\\
&\Big((R-d)\cos\varphi\cos\lambda, -(R-d)\cos\varphi\sin\lambda, (R-d)\sin\varphi\Big)\text{if } P=(R-d,\lambda W,\varphi N),\\
&\Big((R-d)\cos\varphi\cos\lambda, -(R-d)\cos\varphi\sin\lambda, -(R-d)\sin\varphi\Big)\text{if } P=(R-d,\lambda W,\varphi S).
\end{aligned}$$

For the polygone case considered in Sections 5.2 and 5.3, instead of using a triangulation of the polygone as in Section 5.3, we could decompose each polygone into several convex components as in [18,19] and then use the algorithm from Section 5.2 to compute the intersection between the ball and each convex component. However, it is not clear that convexity can be exploited for our problem since for convex polygones, we can have an arbitrary large number of possible intersection shapes and of intersections between the ball and the polygone. We would on top of that need to add the preprocessing time of computing the decomposition in convex polygones. The decomposition in convex components could be of interest if, similar to the case where the convex components are triangles as in Section 5.3, an algorithm more efficient in practice than the algorithm from section 5.2 can be developped. In theory, the proposed algorithms have optimal complexity anyway.

In the case where the ℓ_2-norm is replaced by either the ℓ_1-norm or the ℓ_∞-norm and when $\mathcal{S}$ is a union of disks contained in a plane with P in that plane, we can use the results of Section 5. Indeed, since the level curves of the ℓ_1-norm and the ℓ_∞-norm in the plane are squares, to compute the CDF of D at a given point in these cases we need to determine the area of the intersection of a square (a particular polygone) with disks. It also possible to extend Algorithm 5 to the case where the ℓ_2-norm is replaced by either the ℓ_1-norm or the ℓ_∞-norm and $\mathcal{S}$ is a union of simple polygones.

Another extension of interest is the case where $\mathcal{S}$ is an arbitrary polyhedron in $\mathbb{R}^3$. In this case, the CDF and density of the corresponding random variable D given by $D(\omega) = \|\overrightarrow{PX(\omega)}\|_2$ for any $\omega \in \Omega$ can be approximated using Monte Carlo methods. This is possible if we have at hand a black box able to decide if a given point in $\mathbb{R}^3$ belongs to polyhedron $\mathcal{S}$ or not.

Acknowledgments The author would like to thank Marlon Pirchiner who pointed out useful references for PSHA.

References

1. Open-source software for computing seismic hazard. *OPENQUAKE*, `http://www.globalquakemodel.org/openquake/`.
2. J. W. Baker. An Introduction to Probability Seismic Hazard Analysis (PSHA). `http://www.stanford.edu/~bakerjw/publications.html`, pages 1–72, 2008.
3. C.A. Cornell. Engineering seismic risk analysis. *Bull. Seism. Soc. Am.*, 58:1583–1606, 1968.

4. A. Frankel. Mapping seismic hazard in the Central and Eastern United States. *Seism. Res. Lett.*, 66:8–21, 1995.
5. L. J. Guibas, D. Salesin, and J. Stolfi. Epsilon geometry: building robust algorithms from imprecise computations. *In Proc. 5th Symposium on Computational Geometry*, pages 208–217, 1989.
6. B. Gutenberg and C.F. Richter. Frequency of earthquakes in California. *Bull. Seism. Soc. Am.*, 34:185–188, 1944.
7. I. Kostitsyna, K. Buchin, M. Löffler, and R. I. Silveira. Region-based Approximation Algorithms for Visibility between Imprecise Locations. *Proc. 30th Meeting on Algorithm Engineering & Experiments (ALENEX 2015)*, pages 94–103, 2015.
8. M. Löffler and M. van Kreveld. Largest and Smallest Convex Hulls for Imprecise Points. *Algorithmica*, 56:235–269, 2010.
9. R.K. McGuire. Fortran computer program for seismic risk analysis. *US Geological Survey Open-File Report, Series Number: 76-67*, 1976.
10. Y. Myers and L. Joskowicz. The linear parametric geometric uncertainty model: Points, lines and their relative positioning. *In Proc. 24th European Workshop on Computational Geometry*, pages 137–140, 2008.
11. Y. Myers and L. Joskowicz. Point distance problems with dependent uncertainties. *In Proc. 25th European Workshop on Computational Geometry*, pages 73–76, 2009.
12. M. Ordaz, F. Martinelli, A. Aguilar, J. Arboleda, C. Meletti, and V. D'Amico. Fortran program for computing seismic hazard. *CRISIS 2012 Ver. 1.0*, 2012.
13. J. O'Rourke. *Computational Geometry in C*. Cambridge University Press New York, NY, USA, 1998.
14. Y. Ostrovsky-Berman and L. Joskowicz. Uncertainty envelopes. *In Proc. 21st European Workshop on Computational Geometry*, pages 175–178, 2005.
15. A. J. Stewart. Robust point location in approximate poly. *In Proc. 3rd Canadian Conference on Computational Geometry*, pages 179–182, 1991.
16. R. Stewart and H. Zhang. A note concerning the distances of uniformly distributed points from the centre of a rectangle. *Bull. Aust. Math. Soc.*, 87:115–119, 2013.
17. G. Woo. Kernel Estimation Methods for Seismic Hazard Area Source Modeling. *Bull. Seism. Soc. Am.*, 86:353–362, 1996.
18. B. Chazelle. Convex Decompositions of Polyhedra. *STOC*, 70–79, 1981.
19. J.-M. Lien, N.M. Amato. Approximate convex decomposition of polyhedra. *SIGGRAPH Posters*, 2004.