Certified Approximation Algorithms for the Fermat Point and n-Ellipses

Kolja Junginger
Faculty of Informatics, Università della Svizzera italiana, Lugano, Switzerland

Ioannis Mantas
Faculty of Informatics, Università della Svizzera italiana, Lugano, Switzerland

Evanthia Papadopoulou
Faculty of Informatics, Università della Svizzera italiana, Lugano, Switzerland

Martin Suderland
Faculty of Informatics, Università della Svizzera italiana, Lugano, Switzerland

Chee Yap
Courant Institute, New York University, NY, USA

Abstract

Given a set A of n points in $\mathbb{R}^d$ with weight function $w : A \to \mathbb{R}_{>0}$, the Fermat distance function is $\varphi(\boldsymbol{x}) := \sum_{\boldsymbol{a}\in A} w(\boldsymbol{a})\|\boldsymbol{x}-\boldsymbol{a}\|$. A classic problem in facility location dating back to 1643, is to find the *Fermat point* $\boldsymbol{x}^*$, the point that minimizes the function φ. We consider the problem of computing a point $\widetilde{\boldsymbol{x}}^*$ that is an ε-approximation of $\boldsymbol{x}^*$ in the sense that $\|\widetilde{\boldsymbol{x}}^* - \boldsymbol{x}^*\| < \varepsilon$. The algorithmic literature has so far used a different notion based on ε-approximation of the value $\varphi(\boldsymbol{x}^*)$. We devise a certified subdivision algorithm for computing $\widetilde{\boldsymbol{x}}^*$, enhanced by Newton operator techniques. We also revisit the classic Weiszfeld-Kuhn iteration scheme for $\boldsymbol{x}^*$, turning it into an ε-approximate Fermat point algorithm. Our second problem is the certified construction of ε-isotopic approximations of n-*ellipses*. These are the level sets $\varphi^{-1}(r)$ for $r > \varphi(\boldsymbol{x}^*)$ and $d = 2$. Finally, all our planar ($d = 2$) algorithms are implemented in order to experimentally evaluate them, using both synthetic as well as real world datasets. These experiments show the practicality of our techniques.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases Fermat point, n-ellipse, subdivision, approximation, certified, algorithms

Digital Object Identifier 10.4230/LIPIcs.ESA.2021.54

Funding The work of C. Yap was supported by an USI Visiting Professor Fellowship (Fall 2018); additional support came from NSF Grants # CCF-1564132 and # CCF-2008768. The other authors were partially supported by the SNF project 200021E_154387.

1 Introduction

A classic problem in Facility Location, see e.g., [21, 43], is the placement of a facility to serve a given set of demand points or customers so that the total transportation costs are minimized. The total cost at any point is interpreted as the sum of the distances to the demand points. The point that minimizes this sum is called the *Fermat Point*; see Figure 1. This is an old geometric problem that has inspired scientists over the last three centuries.

A *weighted foci set* is a non-empty finite set of (demand) points $A = \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n\}$ in $\mathbb{R}^d$ associated with a positive weight function $w : A \to \mathbb{R}_{>0}$. Each $\boldsymbol{a} \in A$ is called a *focus* with weight $w(\boldsymbol{a})$. Let $W := \sum_{\boldsymbol{a}\in A} w(\boldsymbol{a})$. The *Fermat distance function* of A is given by

$$\varphi(\boldsymbol{x}) := \sum_{\boldsymbol{a}\in A} w(\boldsymbol{a})\|\boldsymbol{x}-\boldsymbol{a}\|,$$

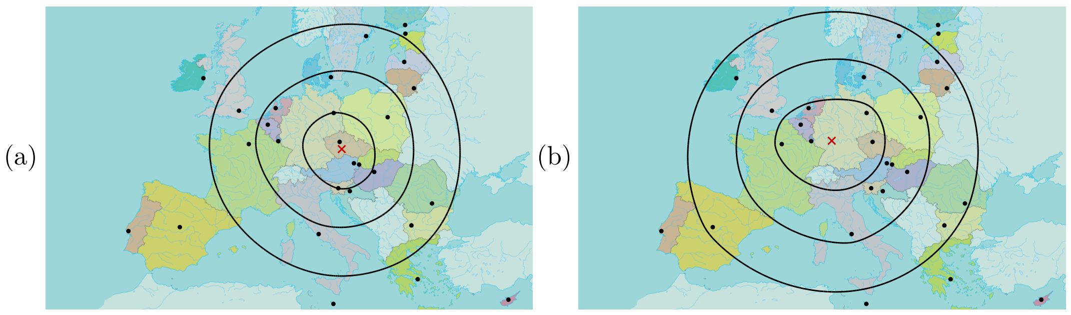


Figure 1 The Fermat point of the 28 EU-capitals (pre-Brexit), highlighted with ($\times$), along with three 28-ellipses of different radii. (a) The foci (capitals) are unweighted. (b) Each focus has the weight of the country's population. The source of the map is `https://www.consilium.europa.eu`.

where $\|\cdot\|$ is the Euclidean norm in $\mathbb{R}^d$. The global minimum value of φ is called the *Fermat radius* of A, denoted by $r^* = r^*(A)$. Any point $\boldsymbol{x} \in \mathbb{R}^d$ that achieves this minimum, $\varphi(\boldsymbol{x}) = r^*$, is called a *Fermat point*, denoted by $\boldsymbol{x}^* = \boldsymbol{x}^*(A)$. The Fermat point is not unique if and only if A is collinear and n is even. We can check if A is collinear in $O(n)$ time, and in that case, the median, which is a Fermat point, can be found in $O(n \log n)$ time. So henceforth, we assume that A is not collinear. In that case φ is a strictly convex function [35, 37], and $\boldsymbol{x}^*$ is unique.

We also consider the closely related problem of computing *n-ellipses* of A. For any $r > r^*(A)$, the level set of the Fermat distance function is $\varphi^{-1}(r) := \{\boldsymbol{x} \in \mathbb{R}^d : \varphi(\boldsymbol{x}) = r\}$. If $n = 1$, the level set is a sphere; and if $n = 2$ and $d = 2$, it is the classic ellipse. When A has n points, we call $\varphi^{-1}(r)$ an *n-ellipsoid*, or an *n-ellipse* if $d = 2$; hence the term *foci set*. From an application perspective, an n-ellipse of radius r can be viewed as a curve that bounds the candidate area for facility location [46], such that the total transportation cost to the demand points is at most r, as in Figure 1.

The question of approximating the Fermat point is of great interest as its coordinates are roots of polynomials of degree exponential in n [3]. For any $\varepsilon > 0$, an *ε-approximation* $\widetilde{\boldsymbol{x}}^*$ to the Fermat point $\boldsymbol{x}^*$ can be interpreted in 3 ways:

(A) Approximate Fermat Point: $\|\widetilde{\boldsymbol{x}}^* - \boldsymbol{x}^*\| \leq \varepsilon$;
(B) Absolute Approximate Fermat Radius: $\varphi(\widetilde{\boldsymbol{x}}^*) \leq \varphi(\boldsymbol{x}^*) + \varepsilon$;
(C) Relative Approximate Fermat Radius: $\varphi(\widetilde{\boldsymbol{x}}^*) \leq (1 + \varepsilon)\varphi(\boldsymbol{x}^*)$.

Thus, we have three ε-approximation problems: (A), (B) and (C). Essentially (B) and (C) are approximations of the Fermat radius, while (A) is a direct approximation of the Fermat point. In this paper we consider approximations in the sense of (A); to the best of our knowledge, only approximations in the sense of (B) and (C), have been considered before, see e.g., [8, 16]. Below, we show that (B) and (C) are easily *reduced* to (A) while the converse reductions are non-obvious (reductions in the sense of complexity theory).

In this work we introduce certified algorithms for approximating the Fermat point and n-ellipses, combining a subdivision approach with interval methods (cf. [33, 48]). The approach can be formalized in the framework of *soft predicates* [56]. Our certified algorithms are fairly easy to implement, and are shown to have good performance experimentally.

Related Work. The problem we study has a long history, with numerous extensions and variations. Out of the 15 names found in the literature, see [23], we call it *the Fermat point problem.* Other common names are the *Fermat-Weber problem* and the *Geometric median*

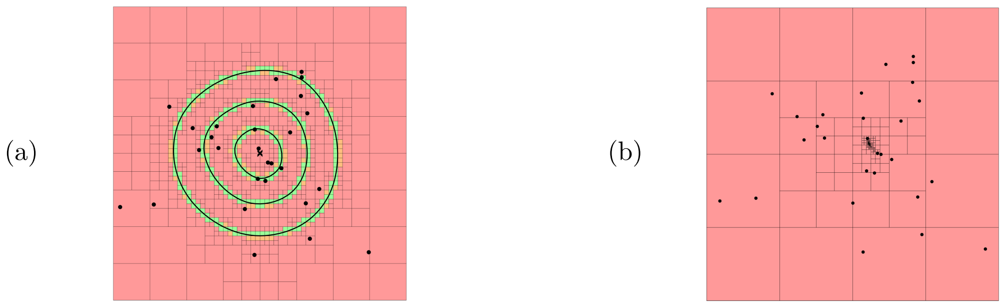


Figure 2 The resulting box subdivision of Figure 1(a) for (a) the n-ellipses and (b) the Fermat point.

problem. Apart from the Facility Location application introduced by Weber [57], the problem is motivated by applications in diverse fields such as statistics and data mining where it is known as the *1-Median problem*, and is an instance of the k-median clustering technique [27].

For $d = 2, n = 3$, the problem was first stated by P. Fermat (1607 - 1665) and was solved by E. Torricelli (1608 - 1647) and Krarup and Vajda [30] using a geometric construction. For $n = 4$, solutions were given by Fagnano [20] and Cieslik [14]. The first general method, for arbitrary n, is an iterative scheme proposed by Weiszfeld [58] in 1937. It was later corrected and improved by Kuhn [32] and Ostresh [43]; see Beck and Sabach [4] for a review. The method is essentially a gradient descent iterative algorithm. It behaves quite well in practice and has only linear convergence, with guaranteed convergence from any starting point.

A plethora of approximation algorithms for the Fermat point, in the senses of (B) and (C), can be found in the literature using various methods. There are algorithms based on semidefinite programming [45], interior point methods [16, 60], sampling [2, 16], geometric data structures [8] and coresets [26], among others [13, 22]. Moreover, special configurations of foci have been considered [7, 15], a continuous version of the problem [21], and also a generalized Fermat point of planar convex objects [1, 12, 18].

The literature on n-ellipses is smaller but equally old: Nagy [38] proved that n-ellipses are convex curves, calling them *egg curves*, and dating them back to Tschirnhaus in 1695 [55, p. 183]. Further, he characterized the singular points of the n-ellipses as being either foci or the Fermat point. Another early work is by Sturm in 1884 [53]. Sekino [51] showed that the Fermat distance function φ is C^∞ on $\mathbb{R}^2 \setminus A$. So, the n-ellipse is a piecewise smooth curve, as it may pass through several foci. Nie et al. [42] showed that the polynomial equation defining the n-ellipses has algebraic degree exponential in n.

Our Contributions. In this paper, we design, implement and experimentally evaluate algorithms for approximating the Fermat point of a given set of foci in $\mathbb{R}^d$. We also compute an ε-approximate n-ellipse; a problem not considered in computational literature before. These are the first certified algorithms [36, 54] for these problems. Our contributions are summarized as follows:

- We design two certified algorithm for the approximate Fermat point: one based on subdivision, the other based on Weiszfeld iteration [58].
- Our notion of ε-approximate Fermat point appears to be new; in contrast, several recent works focus on ε-approximation of the Fermat radius. The approximate Fermat radius can be reduced to approximate Fermat point; the converse reduction is unclear.
- Based on the *PV construction* [47, 33], we design an algorithm to compute a regular isotopic ε-approximation of an n-ellipse. We also augment the algorithm to compute simultaneous contour plots of the distance function φ, resulting in a useful visualization tool (see Figure 1).

- We implement and experimentally evaluate the performance of all our algorithms on different datasets in the plane, as a function of n and ε.

Various details of the interval primitives and proofs can be found in the full arXiv version.

2 Preliminaries

Vector variables are written in bold font: thus $\mathbf{0}$ is the origin of $\mathbb{R}^d$ and $\boldsymbol{x} = (x_1, \dots, x_d)$. For a differentiable function $f : \mathbb{R}^d \to \mathbb{R}$, let $\partial_i f$ denote partial differentiation with respect to x_i. The *gradient* $\nabla f : \mathbb{R}^d \to \mathbb{R}^d$ of f is given by the vector $\nabla f(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \dots, f_d(\boldsymbol{x}))^T$ where $f_i = \partial_i f$. In general, the operator ∇ is partial, i.e., $\nabla f(\boldsymbol{x}_0)$ might not be defined at a point $\boldsymbol{x}_0$. A point $\boldsymbol{x}_0$ is a *critical point* of f if $\nabla f(\boldsymbol{x}) = \mathbf{0}$ or $\nabla f(\boldsymbol{x})$ is undefined.

We consider analytic properties of a scalar function $f : \mathbb{R}^d \to \mathbb{R}$, mainly from the viewpoint of convex analysis [35, 39]. In our case, f is the Fermat distance function for some weighted set A. From an abstract perspective, the Fermat point problem (resp., n-ellipsoid problem) amounts to computing the critical points of the gradient of f (resp., computing the level sets of f). The Fermat point is the only critical point of ∇f in $\mathbb{R}^d \setminus A$, assuming A is non-collinear.

Most of the basic properties regarding the Fermat point are well-known and may be found in our references such as [32, 35, 39, 43, 58]. To emphasize the foci set A, we explicitly write φ_A instead of φ. A focus $\boldsymbol{a} \in A$ is the Fermat point of A if and only if $\|\nabla \varphi_{A \setminus \boldsymbol{a}}(\boldsymbol{a})\| \le w(a)$. Testing if the Fermat point $\boldsymbol{x}^*$ is in A can be done in $O(n^2 d)$ time. If $\boldsymbol{x}^*$ is not one of the foci, then $\nabla f(\boldsymbol{x}^*) = \mathbf{0}$, and the problem can be reduced to general finding real zeros of a square system of polynomial equations (e.g., [59]). However, the thrust of this paper is to develop direct methods that exploit the special properties of the Fermat problem.

We formally define the two main problems which we consider:

- Approximate Fermat Point: Given a weighted point set A in $\mathbb{R}^d$ and $\varepsilon > 0$, compute a point $\widetilde{\boldsymbol{x}}^*$ within ε distance to the Fermat point $\boldsymbol{x}^*$ of A.
- Approximate Isotopic n-Ellipses: Given $\varepsilon > 0$, a weighted point set A in $\mathbb{R}^2$ of size n and a radius $r > r^*(A)$, compute a closed polygonal curve E that is ε-*isotopic* to $\varphi^{-1}(r)$, i.e., there exists an ambient isotopy[1] $\gamma : \mathbb{R}^2 \times [0,1] \to \mathbb{R}^2$ with $\gamma(E, 1) = \varphi^{-1}(r)$ and for any point $\boldsymbol{a} \in \varphi^{-1}(r)$, the parametric curve $\gamma(\boldsymbol{a}, \cdot)$ has at most length ε. This implies a bound of ε on the Hausdorff distance between E and $\varphi^{-1}(r)$.

Approximation notions. We compare the three different notions of ε-approximation for the Fermat point. We reduce the approximation problem of notion (C) to (B), and (B) to (A). An ε-approximation $\widetilde{\boldsymbol{x}}^*$ of $\boldsymbol{x}^*$ in the sense $\|\widetilde{\boldsymbol{x}}^* - \boldsymbol{x}^*\| \le \varepsilon$ is also a $(W\varepsilon)$-approximation in the sense $\varphi(\widetilde{\boldsymbol{x}}^*) \le \varphi(\boldsymbol{x}^*) + W\varepsilon$, which follows directly from the triangle inequality

$$\varphi(\widetilde{\boldsymbol{x}}^*) = \sum_{\boldsymbol{a} \in A} w(\boldsymbol{a}) \|\widetilde{\boldsymbol{x}}^* - \boldsymbol{a}\| \le \sum_{\boldsymbol{a} \in A} w(\boldsymbol{a}) (\|\widetilde{\boldsymbol{x}}^* - \boldsymbol{x}^*\| + \|\boldsymbol{x}^* - \boldsymbol{a}\|) = W\varepsilon + \varphi(\boldsymbol{x}^*).$$

An ε-approximation $\widetilde{\boldsymbol{x}}^*$ of $\boldsymbol{x}^*$ in the sense $\varphi(\widetilde{\boldsymbol{x}}^*) \le \varphi(\boldsymbol{x}^*) + \varepsilon$ is also a $\frac{2\varepsilon}{\varphi(g)}$-approximation in the sense $\varphi(\widetilde{\boldsymbol{x}}^*) \le (1 + \frac{2\varepsilon}{\varphi(g)})\varphi(\boldsymbol{x}^*)$. The center of gravity g is a 2-approximation of the Fermat radius r^* (see [16]), i.e. $\varphi(\boldsymbol{x}^*) \ge \frac{1}{2}\varphi(g)$. Hence

$$\varphi(\widetilde{\boldsymbol{x}}^*) \le \varphi(\boldsymbol{x}^*) + \varepsilon = \left(1 + \frac{\varepsilon}{\varphi(\boldsymbol{x}^*)}\right) \varphi(\boldsymbol{x}^*) \le \left(1 + \frac{2\varepsilon}{\varphi(g)}\right) \varphi(\boldsymbol{x}^*)$$

[1] That is, a continuous map $\gamma : \mathbb{R}^2 \times [0,1] \to \mathbb{R}^2$ such that $\gamma_0 = \gamma(\cdot, 0)$ is the identity map, and, for all $t \in [0,1]$, $\gamma_t = \gamma(\cdot, t)$ is a homeomorphism on $\mathbb{R}^2$.

On the other hand, it is not clear how to derive an ε-approximation of type (A) if an approximation algorithm for type (B) and (C) is at hand, as the following 2 examples show.
Example 1: For any $\varepsilon > 0$ choose $c \leq \frac{\varepsilon}{2\sqrt{2}-2}$ and consider the weighted foci $\boldsymbol{a}_1 = (1,0)$, $\boldsymbol{a}_2 = (0,1)$, $\boldsymbol{a}_3 = (-1,0)$, $\boldsymbol{a}_4 = (0,-1)$ with $w(\boldsymbol{a}_1) = w(\boldsymbol{a}_3) = 1$ and $w(\boldsymbol{a}_2) = w(\boldsymbol{a}_4) = c$ for which the Fermat point is $\boldsymbol{x}^* = (0,0)$ for symmetry reasons, see Figure 3(a). Point $p = (1,0)$ is an ε-approximation of $\boldsymbol{x}^*$ in the sense (B) and (C), but it has a distance of 1 to $\boldsymbol{x}^*$.
Example 2: For any $\varepsilon > 0$ we choose $h > 0$ small enough such that: $2\sqrt{4+h^2} + 2h \leq 4\sqrt{1+h^2} + \varepsilon$. Consider the foci $a_1 = (0,-h)$, $a_2 = (0,h)$, $a_3 = (2,-h)$, $a_4 = (2,h)$ with unit weights. The Fermat point is $\boldsymbol{x}^* = (1,0)$ for symmetry reasons, see Figure 3(b). Point $p = (2,0)$ is an ε-approximation of $\boldsymbol{x}^*$ in the sense (B) and (C), but it has a distance of 1 to $\boldsymbol{x}^*$.

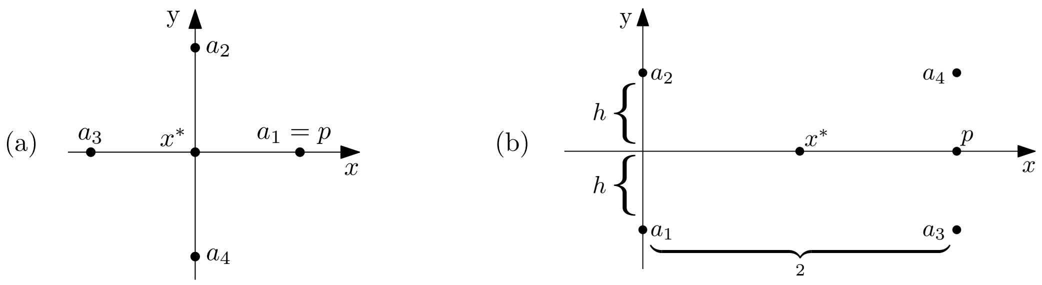


Figure 3 (a) Example that a *good* approximation of the Fermat point in sense (B) does not imply a *good* approximation in sense (A). (b) Analogous example for sense (C).

Subdivision Paradigm. The subdivision algorithms presented in this paper take as input an initial box $B_0 \subset \mathbb{R}^d$ and recursively split it. We organize the boxes in a *generalized quadtree* data structure [50]. A box can be specified by d intervals as $B = I_1 \times I_2 \times \cdots \times I_d$. Let m_B denote the *center* of B, r_B the *radius* of B (distance between m_B and a corner), and $\omega(B)$ the *width* of B (the maximum length of its defining intervals). The term $c \cdot B$ denotes the box with center m_B and radius $c \cdot r_B$. The function $\textsc{split}_1$ takes a box B and returns 2^d congruent subboxes *(children)*, one for each orthant. We use $\textsc{split}_2$ to indicate that we do two successive levels of $\textsc{split}_1$ operations (i.e., $1 + 2^d$ $\textsc{split}_1$ operations, resulting in $(2^d)^2 = 4^d$ leaves).

Soft Predicates. Let $\square\mathbb{R}^d$ denote the set of closed d-dimensional boxes (i.e., Cartesian products of intervals) in $\mathbb{R}^d$. Let P be a logical *predicate* on boxes, i.e., $P : \square\mathbb{R}^d \to \{\texttt{true}, \texttt{false}\}$. For example, the *Fermat point predicate* is given by $P_{\text{fp}}(B) = \texttt{true}$ if and only if $\boldsymbol{x}^* \in B$. Logical predicates are hard to implement, and thus, we may focus on *tests*, which are viewed as one-sided predicates. Formally, a test T looks like a predicate: $T : \square\mathbb{R}^d \to \{\texttt{success}, \texttt{failure}\}$ and it is always associated to some predicate P: call T a *test for predicate* P if $T(B) = \texttt{success}$ implies $P(B) = \texttt{true}$. However, we conclude nothing if $T(B) = \texttt{failure}$. Denote this relation by "$T \Rightarrow P$".

Soft predicates [56] are an intermediate concept between a test and a predicate. Typically, they arise from a partial scalar function $f : \mathbb{R}^d \to \mathbb{R} \cup \{\uparrow\}$ where $f(\boldsymbol{x}) =\uparrow$ means $f(\boldsymbol{x})$ is not defined. We then define a partial *geometric predicate* P_f on boxes B as follows:

$$P_f(B) = \begin{cases} \uparrow & \text{if } \uparrow \in f(B), \\ +1 & \text{if } f(B) > 0, \\ -1 & \text{if } f(B) < 0, \\ 0 & \text{else.} \end{cases}$$

We can now derive various logical predicates P from P_f, by identifying the values in the set $\{-1, 0, +1, \uparrow\}$ with `true` or `false`. For instance, we call P an *exclusion predicate* if we associate the 0- and $\uparrow$-value with `false` and the other values with `true`. For the *inclusion* predicate, we associate the 0-value with `true`, others with `false`. For example, a test for the Fermat point predicate P_{fp} is an inclusion predicate based on the partial function $f(\boldsymbol{x}) = \sum_i (\partial_i f(\boldsymbol{x}))^2$; the function is partial because $f(\boldsymbol{x}) =\uparrow$ when $\boldsymbol{x}$ is a focus point. Although our box predicates $P(B)$ are defined for full-dimensional boxes B, we can extend them to any point $\boldsymbol{x}$ as follows: $P(\boldsymbol{x})$ has the logical value associated with the $sign(f(\boldsymbol{x})) \in \{\uparrow, +1, -1, 0\}$.

▶ **Definition 1.** *Let T be a test for a predicate P. We call T a* soft predicate *(or* soft version *of P) if it is convergent in this sense: if $(B_i : i = 0, 1, \ldots)$ is a monotone sequence of boxes $B_{i+1} \subseteq B_i$ that converges to a point $\boldsymbol{a}$, then $P(\boldsymbol{a}) \equiv T(B_i)$ for i large enough.*

Here, "$P(\boldsymbol{a}) \equiv T(B_i)$" means $P(\boldsymbol{a}) =$ `true` if and only if $T(B_i) =$ `success`. A soft version of $P(B)$ is usually denoted $\square P(B)$. We note that soft versions of exclusion predicates are generally easier to construct than inclusion predicates. The former can be achieved by numerical approximation, while the latter requires some deeper principle such as the Brouwer fixed point theorem [9].

Interval arithmetic. We construct soft predicates using functions of the form $F : \square\mathbb{R}^d \to \square(\mathbb{R} \cup \{-\infty, \infty\})$ that approximate the scalar function $f : D \to \mathbb{R}$ with $D \subset \mathbb{R}^d$.

▶ **Definition 2.** *Call F a* soft version *of f if it is*

i) conservative*, i.e., for all $B \in \square\mathbb{R}^d$, $F(B)$ contains $f(B) := \{f(p) : p \in B \cap D\}$, and*

ii) convergent*, i.e., if for monotone sequence $(B_i : i \geq 0)$ that converges to a point $\boldsymbol{a} \in D$, $\lim_{i\to\infty} \omega(F(B_i)) = 0$ holds.*

We shall denote F by $\square f$ when F is a soft version of f. There are many ways to achieve $\square f$. For example, if f has an arithmetic expression E, we can simply evaluate E using interval arithmetic. More sophisticated methods may be needed for performance. The next lemma shows how $\square f$ leads to soft exclusion predicates based on f.

▶ **Lemma 3.** *If P is an exclusion predicate based on f, then the test $\square P(B) : 0 \notin \square f(B)$ is a soft version of P.*

Below, we need a multivariate generalization, to the case where $\boldsymbol{f} : \mathbb{R}^d \to \mathbb{R}^m$, and the exclusion predicate $P(B)$ is $\boldsymbol{0} \notin \boldsymbol{f}(B)$. If $\square\boldsymbol{f} : \square\mathbb{R}^d \to \square\mathbb{R}^m$ is a soft version of $\boldsymbol{f}$, then a soft version of $P(B)$ is the given by the test $T(B) : \boldsymbol{0} \notin \square\boldsymbol{f}(B)$. If $\boldsymbol{f} = (f_1, \ldots, f_m)$, then this reduces to $0 \notin \square f_i(B)$ for some $i = 1, \ldots, m$.

3 Approximate Fermat points

We now present three approximation algorithms for the Fermat point $\boldsymbol{x}^*$. For simplicity, we assume in our algorithms that the Fermat point is not a focus, i.e. $\boldsymbol{x}^* \notin A$. This assumption can be easily checked in $O(n^2 d)$ preprocessing time, or with a more elegant approach, in $O(nd)$ time during the execution of our subdivision algorithms.

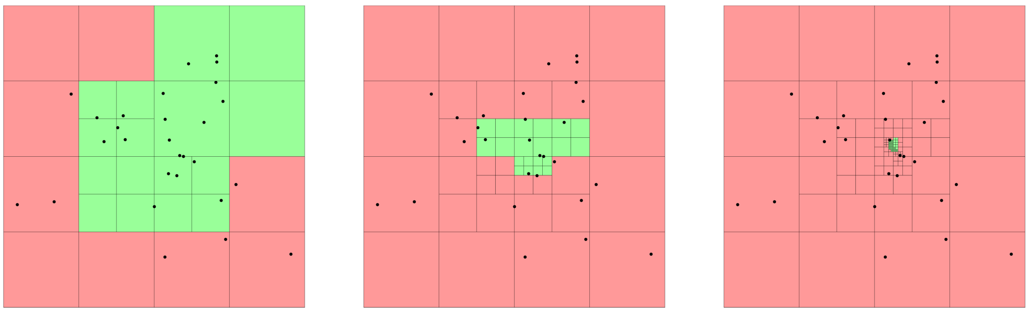


Figure 4 Different steps during the the execution of Algorithm 1. The dark red boxes cannot contain the Fermat point, whereas the light green boxes may contain it.

3.1 Using the Subdivision Paradigm

The subdivision paradigm requires an initial box B_0 to start subdividing. If B_0 is not given, it is easy to find a box that contains $\boldsymbol{x}^*$, since $\boldsymbol{x}^*$ lies in the convex hull of A [32]. We use a function Initial-Box(A) which, in $O(nd)$ time, computes an axis-aligned bounding box with corners having the minimum and maximum x, y coordinates.

We define an exclusion and inclusion predicate based on the gradient function $\nabla\varphi$.

▶ **Definition 4.** *Given a box B, the* gradient exclusion predicate *$C_0^\nabla(B)$ is defined by the condition $\mathbf{0} \notin \nabla\varphi(B)$. The* gradient inclusion predicate *$C_1^\nabla(B)$ is just the complement of $C_0^\nabla(B)$, that is $\mathbf{0} \in \nabla\varphi(B)$.*

Under our assumptions that $\boldsymbol{x}^* \notin A$, we have that $C_1^\nabla(B)$ holds if and only if $\boldsymbol{x}^* \in B$. We obtain a soft version of the exclusion predicate $C_0^\nabla(B)$ by replacing $\nabla\varphi$ in its definition with any soft version $\square\nabla\varphi$, see Lemma 3. But it is not so easy to get a soft version of $C_1^\nabla(B)$; we shall return to this when we treat the Newton operator below.

In Algorithm 1, using the exclusion predicate we discard boxes that are guaranteed not to contain $\boldsymbol{x}^*$ (red in Figure 4) and we split boxes that might contain $\boldsymbol{x}^*$ (green in Figure 4). While subdividing, we test whether we can already approximate $\boldsymbol{x}^*$ well enough by putting a bounding box around all the boxes that are not excluded yet, using the following predicate.

▶ **Definition 5.** *Given a set of boxes Q that contains the Fermat point, the* stopping predicate *$C^\varepsilon(Q)$ returns true, if and only if the minimum axis-aligned bounding box containing all boxes in Q has a radius at most ε.*

If C^ε returns true, then we can stop. Since the radius of the minimum bounding box is at most ε, the center of the box is an ε-approximate Fermat point $\widetilde{\boldsymbol{x}}^*$.

Algorithm 1 Subdivision for the approximate Fermat point (*SUB*).

```
Input  : Foci set $A$, constant $\varepsilon > 0$
Output : Point $\widetilde{\boldsymbol{x}}^*$
1 $B_0 \leftarrow$ Initial-Box($A$);   $Q \leftarrow$ Queue();   $Q$.push($B_0$);
2 while not $C^\varepsilon(Q)$ do
3     $B \leftarrow Q$.pop();
4     if not $\square C_0^\nabla(B)$ then
5         $Q$.push(split$_1$($B$));
6 return $\widetilde{\boldsymbol{x}}^* \leftarrow$ Center of the bounding box of $Q$;
```

Regarding the runtime of Algorithm 1, evaluating $\nabla\varphi$ and its soft version takes linear time in n. The subdivision approach induces an exponential dependency on d, as splitting a box creates 2^d many children. Further, a $\textsc{split}_1$ operation decreases the boxwidth by a factor of 2, therefore, Algorithm 1 cannot converge faster than linear in ε.

3.2 Enhancing the Subdivision Paradigm

In this section, we augment Algorithm 1 with a speed up based on a *Newton operator*, which will ensure eventual quadratic convergence.

The Newton operator. Newton-type algorithms have been considered in the past, usually independently of other methods, and thus suffer from lack of global convergence. Moreover, from a numerical viewpoint, such methods face the *precision-control problem.* Our algorithm integrates subdivision with the Newton operator (an old idea that goes back to Dekker [17] in the 1960's), thus ensuring global convergence.

We want to find the Fermat point, i.e., the root of $\boldsymbol{f} = \nabla\varphi$. Newton-type operators are well-studied in the interval literature, and they have the form $N = N_{\boldsymbol{f}} : \square\mathbb{R}^d \to \square\mathbb{R}^d$. There are three well-known versions of $N_{\boldsymbol{f}}$: the simplest version, from Moore [36] and Nickel [40], is

$$N(B) = m_B - J_{\boldsymbol{f}}^{-1}(B) \cdot \boldsymbol{f}(m_B),$$

where $J_{\boldsymbol{f}}$ is the Jacobian matrix of $\boldsymbol{f}$. Since $\boldsymbol{f} = \nabla\varphi$, this matrix is actually the Hessian of φ. The second version by Krawzcyk [31, 52] is:

$$N(B) = \; m_B - K \cdot f(m_B) + (I - K \cdot \boldsymbol{f}(B)) \cdot (B - m_B),$$

where K is any non-singular $d \times d$ matrix, usually chosen to be an approximation of $J_{\boldsymbol{f}}^{-1}(m_B)$. The third version, from Hansen and Sengupta [24, 25], can be viewed as a sophisticated implementation of the Moore-Nickel operator using an iteration reminiscent of the Gauss-Seidel algorithm, combined with preconditioning. Later we report on our implementation of the first two Newton operators. In general, the Newton operator $N(B)$ does not return a box even if B is a box; so we define $\square N(B)$ to be a box that contains $N(B)$. For simplicity, we assume that $\square N(B)$ is the smallest box containing $N(B)$ with the same aspect ratio as B.

The following three properties of Newton box operators are consequences of Brouwer's Fixed Point Theorem [9, 41, 52, 59]:

1. (Inclusion Property) If $N(B) \subseteq B$ then $\boldsymbol{x}^* \in N(B)$.
2. (Exclusion Property) If $N(B) \cap B = \emptyset$ then $\boldsymbol{x}^* \notin B$.
3. (Narrowing Operator) If $\boldsymbol{x}^* \in B$ then $\boldsymbol{x}^* \in N(B)$.

Based on these properties, we can define two tests and an operator:

▶ **Definition 6.** *Newton tests for gradient exclusion/inclusion predicates (below we explain why we use $2B$ instead of B):*

- ***Newton exclusion test:***
  $T_0^N(B) =$ `success` *iff* $N(2B) \cap B = \emptyset$. *Thus* $T_0^N(B) \Rightarrow C_0^{\nabla}(B)$.
- ***Newton inclusion test:***
  $T_1^N(B) =$ `success` *iff* $N(2B) \subseteq 2B$. *Thus* $T_1^N(B) \Rightarrow C_1^{\nabla}(2B)$.
- ***Newton narrowing operator:***
  $N_\cap(B)$ *returns* $B \cap N(2B)$.

Note that the Newton tests $T_0^N(B)$ and $T_1^N(B)$ are defined using the exact Newton operator $N(B)$. If we replace it by a soft version $\square N(B)$ in these definitions, they remain as inclusion/exclusion tests for $C_1^\nabla(B)$ and $C_0^\nabla(B)$; we denote them by $\square C_1^\nabla(B)$ and $\square C_0^\nabla(B)$.

To compute $\square N(B)$, we use standard interval arithmetic to evaluate the Newton operators. We already noted that if $N(B) \subseteq B$, then $\boldsymbol{x}^* \in N(B)$. But if $\boldsymbol{x}^*$ is on the boundary of B, then $\square N(B) \subseteq B$ might not hold, and this issue persists even after splitting B. We circumvent this problem by using $2B$ instead of B in the definition of $T_1^N(B)$.

We enhance Algorithm 1 by the soft inclusion predicate $\square T_1^N(B)$, as sketched in Algorithm 2. If $\square T_1^N(B)$ succeeds, we conclude that $\boldsymbol{x}^*$ is contained in $\square N(2B)$. In that case, we can discard all other boxes and initialize a new queue Q on $\square N(2B)$. In subsequent calls to $\square T_1^N(B')$ for $B' \in Q$, we conclude that $\boldsymbol{x}^* \in 2B'$. But to ensure that $w(2B') < w(B)$ (to avoid an infinite loop), we initialize the queue Q with the 4^d boxes of $\text{SPLIT}_2(\square N(2B))$.

Algorithm 2 Enhanced subdivision for the approximate Fermat point (*ESUB*).

```
    As in Algorithm 1 but replace line 5 with the following:
5.1 if □T_1^N(B) then
5.2     Q ← Queue();                          // initialize a new queue
5.3     Q.push(split_2(□N(2B));                  // 2 split operations
5.4 else
5.5     Q.push(split_1(B));
```

With respect to the runtime of Algorithm 2, we observe that once the soft Newton inclusion predicate succeeds, then it will also do so for an initial box of the new queue. This, essentially, divides the algorithm into two phases. The first phase can be basically seen as Algorithm 1. In the second phase, the Newton test guarantees quadratic convergence in ε. Getting into the second phase depends on the configuration of the foci set but not on ε, hence, our approach is of particular interest for small values of ε.

The termination of both subdivision algorithms follows from the soft gradient exclusion predicate being convergent. The algorithms terminate once the predicate $C^\varepsilon(Q)$ succeeds, yielding an ε-approximate Fermat point, so we summarize as follows.

▶ **Theorem 7.** *Both Algorithms 1 and 2 terminate and return an ε-approximate Fermat point.*

3.3 Certifying the Weiszfeld method

Weiszfeld's iterative method [32, 43, 58] describes a sequence $\boldsymbol{p}_i$ $(i = 0, 1, \dots)$ of points that converges to the Fermat point $\boldsymbol{x}^*$, starting from any initial $\boldsymbol{p}_0$. The recurrence relation is $\boldsymbol{p}_{i+1} = T(\boldsymbol{p}_i)$, where $T(\boldsymbol{x})$ is defined by

$$T(\boldsymbol{x}) = \frac{\sum_{\boldsymbol{a}\in A, \boldsymbol{a}\neq\boldsymbol{x}} w(\boldsymbol{a}) \frac{\boldsymbol{a}}{\|\boldsymbol{x}-\boldsymbol{a}\|}}{\sum_{\boldsymbol{a}\in A, \boldsymbol{a}\neq\boldsymbol{x}} w(\boldsymbol{a}) \frac{1}{\|\boldsymbol{x}-\boldsymbol{a}\|}}.$$

Note that when $\boldsymbol{x}$ is a focus, then $T(\boldsymbol{x})$ depends just on all other foci.

This simple iterative method is widely used, and although it converges, it does not solve our ε-approximation problem as we do not know when to stop. To see that this is a real issue, consider the example in Figure 5.

We augment the Weiszfeld iteration by adding Newton tests during the computation, turning it into an ε-approximation algorithm. While at the i-th iteration, we define a small box B with point $\boldsymbol{p}_i$ as center, and map it to the box $\square N(B)$ using the Newton operator; see Figure 6. If $\square N(B) \subseteq B$, then the Fermat point $\boldsymbol{x}^*$ lies in $\square N(B)$. On the contrary, if $\square N(B) \not\subseteq B$ we move on to the next point $\boldsymbol{p}_{i+1}$ and adjust the box size as follows.

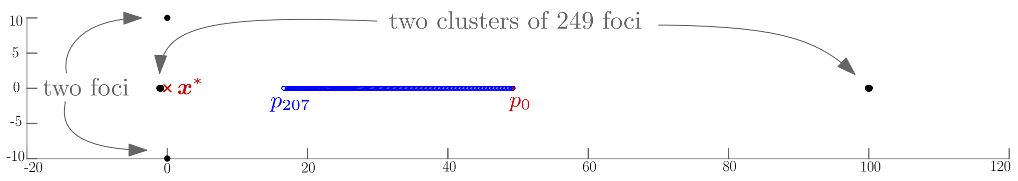


Figure 5 An example with 500 foci, showing that Weiszfeld's scheme does not solve the ε-approximation problem. The scheme stopped when $\left\|\boldsymbol{p}_{i-1} - \boldsymbol{p}_i\right\| \leq 1/10$, after 207 steps (blue points). The distance $\|\boldsymbol{x}^* - \boldsymbol{p}_{207}\|$ can be arbitrarily big ($\|\boldsymbol{x}^* - \boldsymbol{p}_{207}\| > 15$ in this case).

If $\frac{B}{10} \cap \square N(\frac{B}{10}) = \emptyset$, then the box $\frac{B}{10}$ does not contain $\boldsymbol{x}^*$ and we therefore expand B by a factor of 10. If $\frac{B}{10} \cap \square N(\frac{B}{10}) \neq \emptyset$, then there might be a focus in box $\frac{B}{10}$, which hinders $\square N(B) \subseteq B$ to succeed. In that case we shrink B by a factor of 10. If a focus is not in $\frac{B}{10}$, shrinking B does not effect the algorithm negatively, as B can expand again.

Using these tests we augment the point sequence scheme, sketched in Algorithm 3, with the property that if the Newton test evaluates to true, then we are guaranteed an ε-approximation of $\boldsymbol{x}^*$. As a starting point, we choose the *center of mass* $\boldsymbol{p}_0$ of A, i.e., $\boldsymbol{p}_0 = \frac{1}{W}\sum_{\boldsymbol{a}\in A} w(\boldsymbol{a})\,\boldsymbol{a}$.

With respect to the runtime, the point sequence $T(\boldsymbol{x})$ converges linearly in ε towards $\boldsymbol{x}^*$ [29] but in order for Algorithm 3 to terminate the test $\square N(B) \subseteq B$ must succeed. Similar to other Newton operators, $\square N(B) \subseteq B$ succeeds for boxes in a neighborhood surrounding $\boldsymbol{x}^*$. This neighborhood depends only on the configuration of A but not on ε. Further, evaluating $T(\boldsymbol{x})$ and $\square N(B)$ can be done in $O(nd^2)$ time. We conclude as follows.

▶ **Theorem 8.** *Algorithm 3 terminates and returns an ε-approximate Fermat point.*

Algorithm 3 Certified Weiszfeld for the approximate Fermat point (CW).

```
Input: Foci set $A$, constant $\varepsilon > 0$        Output: Point $\widetilde{\boldsymbol{x}}^*$
 1  $\boldsymbol{p} \leftarrow \boldsymbol{p}_0$;   $l \leftarrow \varepsilon$;
 2  while TRUE do
 3      $B \leftarrow$ Box $B(m_B = p,\ \omega(B) = l)$;
 4      if $\square N(B) \subseteq B$ then                       // Figure 6(a)
 5          return $\widetilde{\boldsymbol{x}}^* \leftarrow \boldsymbol{p}$;
 6      else if $\square N\left(\frac{B}{10}\right) \cap \frac{B}{10} = \emptyset$ then      // Figure 6(b)
 7          $l \leftarrow \min\{10 \cdot l, \varepsilon\}$;
 8      else                                           // Figure 6(c)
 9          $l \leftarrow \frac{1}{10} \cdot l$;
10      $\boldsymbol{p} \leftarrow T(\boldsymbol{p})$;
```

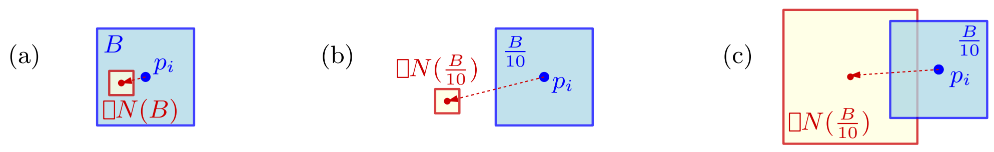


Figure 6 The case analysis of Algorithm 3. (a) $\square N(B) \subseteq B$, (b) $\square N(\frac{B}{10}) \cap \frac{B}{10} = \emptyset$, and (c) $\square N(\frac{B}{10}) \cap \frac{B}{10} \neq \emptyset$.

4 Approximating n-ellipses

In this section, we describe an algorithm to construct approximate n-ellipses, based on the subdivision paradigm. Throughout this work we maintain the subdivision *smooth*, i.e., the width of any two adjacent boxes, which are leaves of the quadtree, may differ at most by a factor of 2. Maintaining smoothness is easy to implement and has amortized $O(1)$ cost per operation [6]. Without maintaining smoothness, the amortized cost can be $\Omega(\log n)$ [6].

The Plantinga and Vegter (PV) construction [47, 33, 34] approximates the zero set of a function $F : \mathbb{R}^d \to \mathbb{R}$ where $d \in \{2, 3\}$. Assuming that $S = F^{-1}(0)$ is regular, i.e., the gradient ∇F is non-zero at every point of S, this approximation is isotopic to S. Our goal is to use this construction to approximate the n-ellipse defined by $F(p) := \varphi(p) - r$ with $r > r^*$. For simplicity, we assume all boxes are square; for the construction to succeed, we only need an aspect ratio $\leq \sqrt{2}$ (see [33]). We use the notation $\langle \cdot, \cdot \rangle$ for the scalar product. The following are the key predicates and tests in the PV construction of the n-ellipse $F^{-1}(0)$.

▶ **Definition 9.** *Fix $F(p) = \varphi(p) - r$. Let B be a square box.*

1. *The fundamental box predicate is the* inclusion predicate $C_1^F(B) : 0 \in F(B)$, *and its complement, the* exclusion predicate $C_0^F(B) : 0 \notin F(B)$.
2. *The* (corner) inclusion test $T_{cor}(B) =$ `success` *iff F, when evaluated at the corners of B, admits both negative and positive values. Clearly, $T_{cor}(B)$ is a test for $C_1^F(B)$. (There is a standard PV trick whereby any 0-value can be arbitrarily made positive.)*
3. *The* normal variation predicate $C_{nv}(B)$ *is defined by the condition* $\langle \nabla F(B), \nabla F(B) \rangle > 0$.

We obtain the soft versions $\square C_0^F(B)$ and $\square C_{\mathrm{nv}}(B)$ by the usual device of replacing $F(B)$ in the definition of the predicates by a soft version $\square F(B)$. But for the inclusion predicate $C_1^F(B)$ we have no soft version. Instead, the corner test $T_{\mathrm{cor}}(B)$ is a test for $C_1^F(B)$. To supplement the corner test, we need the normal variation predicate $C_{\mathrm{nv}}(B)$. This predicate is equivalent to the condition that the angle between the gradient of any two points in B is at most 90°. It implies that the n-ellipse is monotone in either x- or y-direction within the box. In Figure 7, boxes are: **red** if they pass the $\square C_0^F(B)$ test, **green** if they pass both $\square C_{\mathrm{nv}}$ and T_{cor}, **orange** if the pass only $\square C_{\mathrm{nv}}$, and gray otherwise. Note that orange boxes may, or may not, contain parts of the approximate n-ellipse.

An n-ellipse is not regular if it passes through some focus [51]; in that case a direct PV construction is not possible. We develop a variation, sketched in Algorithm 4, where we simultaneously subdivide boxes and construct pieces of the n-ellipse *on the fly*, instead of doing that in the end. Further, boxes in which the n-ellipse may not be regular are treated differently. During the subdivision part of the algorithm, we classify boxes in three categories:

1. Boxes which satisfy $\square C_0^F(B)$ (**red**): These do not contain any piece of the n-ellipse, so they do not need to be further considered and are discarded.
2. Boxes which satisfy $\square C_{\mathrm{nv}}$ and have width smaller than $\varepsilon/2$ (**green** or **orange**): We immediately draw edges in each of these boxes, in contrast to the normal PV construction. Note that at a later stage of the algorithm it might happen that we split one of B's neighboring boxes. In that case we need to take into account the sign of F at the new vertex on B's boundary. If necessary, the edges in box B then need to be updated.
3. The remaining boxes (**gray**): Such boxes occur near foci and need more careful attention, as we cannot apply the standard PV construction. Instead, given a set of gray boxes we first distinguish them in connected components, using a DFS algorithm. Then, for each connected component of gray boxes K_i, we check if a set of conditions is satisfied:

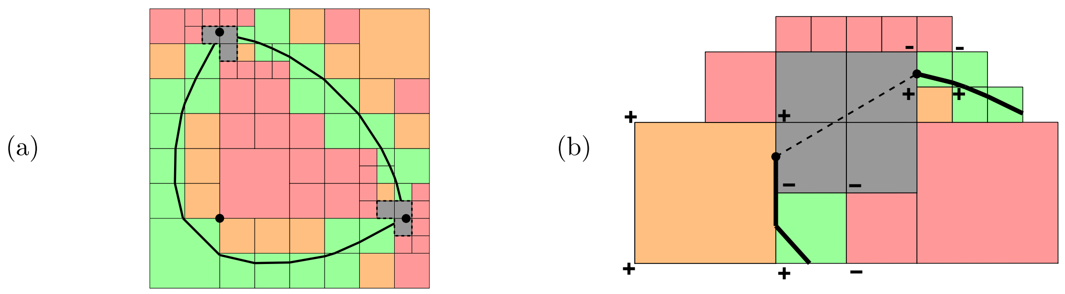


Figure 7 (a) A 3-ellipse passing through two foci. Components of gray boxes (temporarily) surround the foci. (b) If a gray component satisfies (B1) - (B3) the two ingoing edges are connected with an edge (shown dashed).

(B1) K_i contains exactly one focus.
(B2) There are exactly two PV-edges leading to K_i.
(B3) The distance between any two corners of the boxes in K_i is at most $\varepsilon/2$.

If K_i satisfies all (B1) - (B3), then we connect the 2 PV-edges leading to K_i by a line segment and discard boxes of K_i, see Figure 7(b). Otherwise, the children of the boxes of K_i are put back in Q for further classification.

Algorithm 4 Approximating an n-ellipse.

```
Input: Foci set $A$, radius $r$, constant $\varepsilon$, box $B_0$        Output: Curve $E$
$Q \leftarrow$ QUEUE();   $Q$.PUSH($B_0$);
while $Q \neq \emptyset$ do
    $Q_{new} \leftarrow$ QUEUE();
    while $Q \neq \emptyset$ do
        $B \leftarrow Q$.POP();
        if not □$C_0^F(B)$ then                                // exclude red
            if □$C_{nv}(B)$ and $\omega(B) < \varepsilon/2$ then        // green or orange
                $E_{\cap B} \leftarrow$ ONLINE-PV($B$);
            else                                               // gray
                $Q_{new}$.PUSH(SPLIT$_4$($B$));
    $Q \leftarrow$ CONNECTED-COMPONENTS-ANALYSIS($Q_{new}$)
return $E$;
```

By controlling the size of the boxes containing parts of the output curve, and by the modification the PV construction we prove the following.

▶ **Theorem 10.** *Algorithm 4 returns an isotopic ε-approximation of the n-ellipse $F^{-1}(0)$.*

Interpolating edges. The PV construction creates edges within a box B, which start and end at midpoints of box edges. One can derive a nicer-looking approximation by using linear interpolation on the box edges by taking into account the value of F at B's corners.

Contour Plotting. As an application, we can use the above technique in order to produce a topologically correct, ε-approximate and visually nice *n-elliptic contour plot.* To do so, we first adapt our algorithm in order to simultaneously plot several n-ellipses inside a bounding

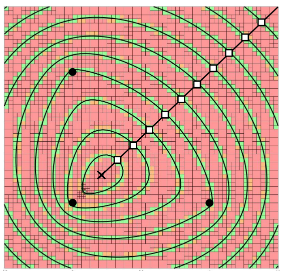
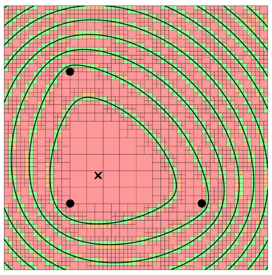
(a)  (b) 

Figure 8 Two different 3-elliptic contour plots with 10 contour lines, having the same set of foci. (a) Using radii of *equidistant points.* (b) Using *equidistant radii.*

box, corresponding to the same foci but with different radii. Each n-ellipse is a *contour line*, and we describe how to plot them *visually nice*, i.e., the contour lines are roughly equally distributed in space. See Figure 8 for two different approaches and their visualization effect.

5 Experiments

We implemented our algorithms for $\mathbb{R}^2$ and conducted a series of experiments. Our current software is written in MATLAB (version R2018b), taking advantage of its graphics ability. The numerical accuracy is therefore IEEE numerical precision. The platform used was MacOS Big Sur v11.2.3, with 2.5 GHz Quad-Core Intel Core i7 and 16 GB 1600MHz DDR3.

Following, we report on our experiments, discussing some notable points one by one. We evaluated our algorithms on both synthetic and real-world datasets. For all algorithms approximating the Fermat point we chose a time limit of 600 seconds. Moreover, for most experiments we executed 10 different instances for completeness. In the illustrated charts, the curves pass through the mean of the 10 running times, and additionally we also marked the minimum and maximum running times. All axes in the charts are of logarithmic scale.

Datasets. We mainly experimented with two different types of synthetic datasets, namely Unif-1 and Unif-2. In Unif-1 the n foci are sampled uniformly from a disk of radius 1. In Unif-2 again the n foci are sampled uniformly from a disk of radius 1 and then $n/2$ foci are translated by a vector $(10, 10)$, see Figure 9(a) and Figure 9(b). Despite their similarity, the two datasets present strong differences. As we later see, Unif-2 is significantly more difficult to solve in comparison to Unif-1, and further Unif-1 resembles nicely real-world datasets. The foci of Unif-2 lie almost all on a common line, which implies that there are many points for which the gradient is close to 0. This makes it difficult to find the actual Fermat point, for which the gradient is exactly 0. We experimented with more types of synthetics datasets, such as points in convex position, vertices of a regular n-gon, clusters of points, but we do not report on these results, as they are similar to Unif-1 or Unif-2.

Newton operators. Adding a Newton operator to the subdivision process drastically improves the running time. We compared Algorithm 1 with two versions of Algorithm 2, where we once use the Newton operator based on Moore and Nickel and also the operator by Krawzcyk. The results for various values of n and ε on both Unif-1 and Unif-2 are summarized in Figure 10. Note that Algorithm 2 initially needs to perform simple splitting operations until at some point the Newton test succeeds the first time. After that the algorithm converges quadratically in ε, which explains why the running time of both versions almost do not increase for decreasing ε. Even though the operator by Krawzcyk returns a

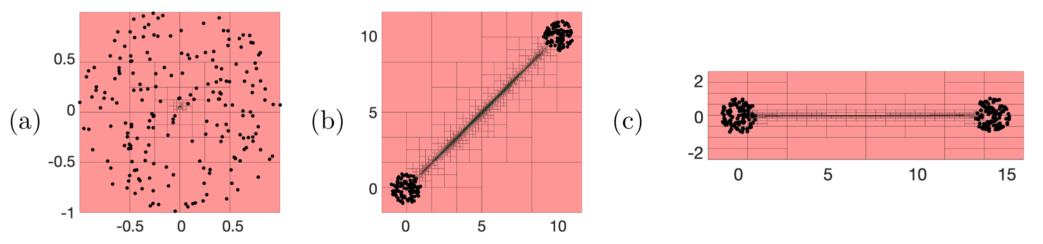


Figure 9 A box subdivision for $n = 200$ foci: (a) Unif-1, (b) Unif-2 and (c) Unif-2 after PCA.

smaller box $\square N(B)$, i.e. it is more precise, than Moore and Nickel, it performs slower for Unif-1 as evaluating the operator takes more time. We conclude that using a Newton operator speeds up the computations, and we use the one of by Moore and Nickel in Algorithm 2.

Principal component analysis. Foci sets like Unif-2 are challenging as all foci are close to a common line. In this case, the subdivision algorithms can be slow because there are many boxes for which the gradient $\nabla\varphi$ is close to 0. Our approach to tackle this problem is to use subdivision with rectangular boxes. In a preprocessing step we do a *principal component analysis* (PCA) of the foci as heuristic. Then, we rotate the coordinate system such that the x-direction is the first principal component. In the box subdivision we use rectangular boxes with long x-width, see Figure 9(c). Observe in the following table, that for well distributed foci sets like Unif-1, using the PCA adds only a small overhead to the total running time.

$\varepsilon = 10^{-3}$, $n =$	10	100	1000	10000
without PCA	0.12	0.31	2.33	23.4
with PCA	0.10	0.30	2.30	23.9

$n = 100$, $\varepsilon =$	10^{-1}	10^{-3}	10^{-5}	10^{-7}
without PCA	0.20	0.30	0.33	0.34
with PCA	0.18	0.30	0.33	0.35

On the contrary, for sets like Unif-2, adding the PCA decreases drastically the running time, as shown next. Hence, the PCA preprocessing is a useful addition to Algorithm 2, which we will use also in the following experiments.

$\varepsilon = 10^{-3}$, $n =$	10	100	1000	10000
without PCA	90.7	48.5	170	*timeout*
with PCA	0.15	0.40	3.21	32.7

$n = 100$, $\varepsilon =$	10^{-1}	10^{-3}	10^{-5}	10^{-7}
without PCA	37.1	49.2	49.2	49.5
with PCA	0.36	0.40	0.42	0.43

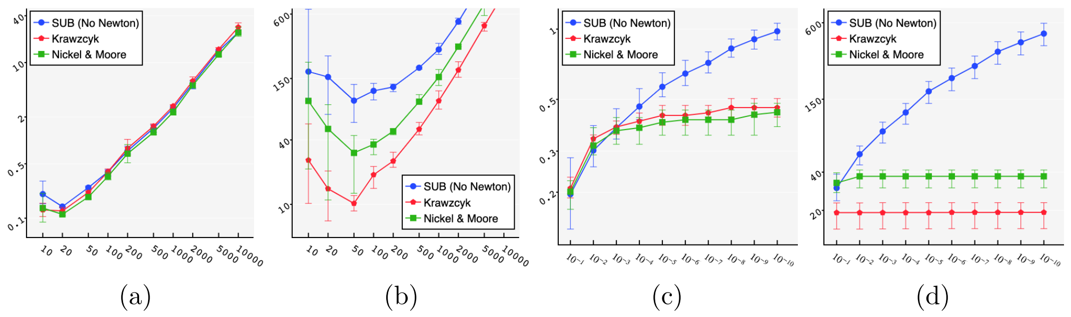
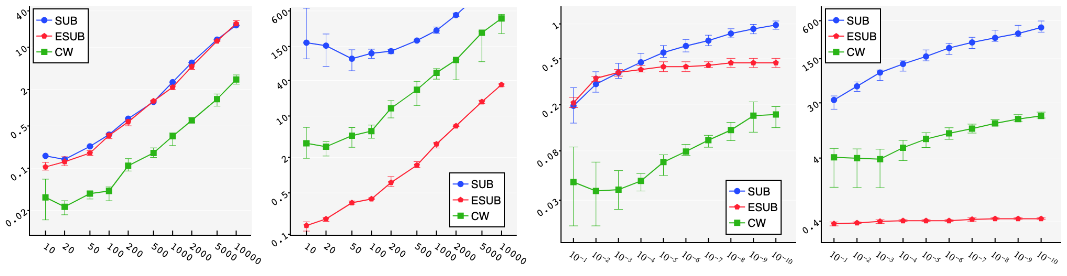


Figure 10 A comparison of Algorithm 1 (● SUB), Algorithm 2 with the Krawczyk Newton operator (⬟ Krawczyk), and Algorithm 2 with the Nickel and Moore Newton operator (🟩 Nickel & Moore). (a),(b) Time as a function of n, with $\varepsilon = 10^{-4}$. (c),(d) Time as a function of ε with $n = 100$. (a),(c) Unif-1 datasets. (b),(d) Unif-2 datasets.

Real Datasets. Inspired by the applications in facility location we chose to experiment with instances of the well-known *Traveling Salesman Person Library* [49] or TSPLIB. The foci correspond mostly to location of cities in different areas around the world. It appears that real-world instances show a similar behavior to UNIF-1 datasets; so we infer that UNIF-1 are realistic datasets for the evaluation of different algorithms. In our experiments, to verify that for each TSPLIB dataset we created an additional foci set, where we uniformly sampled the same number of foci in the axis-aligned bounding box. As ε we chose 10^{-6} times the width of the corresponding bounding box. These experiments are illustrated in Figure 12(a), and the similarity of the running time for the two datasets is obvious.

Summary on the Fermat point. We make an overall comparison of Algorithm 1, Algorithm 2 with the PCA, and Algorithm 3, illustrated in Figure 11. The running time of all methods shows a linear dependency on n, but there are big differences regarding the dependency on ε. Overall, Algorithm 3 performs well in all cases, but due to the linear convergence of Weiszfeld's point sequence, it cannot converge faster as ε decreases. In contrast, Algorithm 2 takes more time in the subdivision phase, but once the Newton tests succeeds, the algorithm terminates very quickly. So, it does not exhibit almost any changes in the running time for decreasing ε. This makes it favorable when a high precision approximate solution is required. It is also very fast in UNIF-2 instances and outperforms Algorithm 3. Summarizing, we suggest to use Algorithm 2 in small dimensional spaces and for small ε due to its eventual quadratic convergence in ε. On the other hand, the subdivision methods take exponential time in d, therefore, we suggest to use Algorithm 3 for higher dimensional spaces.

***n*-ellipses.** Finally, we evaluated the runtime of n-ellipses algorithm. In Figure 12(b) we evaluate the dependency on n. In order to keep the length of the curve almost constant we choose the radii $r = \frac{(10\sqrt{2}+2)n}{2}$. The bounding box used is $[-2, 12]^2$. In Figure 12(c) we analyze the dependency on the length of the n-ellipse. The bounding box is fixed and we experimented with different radii such that the lengths of the curve differ by a factor of $3/2$. The runtime shows a linear dependency on n, as expected, and it also shows a linear dependency on the length of curve. This can be justified, as covering an n-ellipse of length l with boxes of width ε takes $O(l/\varepsilon)$ many boxes.



(a) (b) (c) (d)

Figure 11 An overall comparison of Algorithm 1 (● SUB), Algorithm 2 with the PCA (⬟ ESUB), and Algorithm 3 (🟩 CW). (a),(b) Time as a function of n, with $\varepsilon = 10^{-4}$. (c),(d) Time as a function of ε with $n = 100$. (a),(c) UNIF-1 datasets. (b),(d) UNIF-2 datasets.

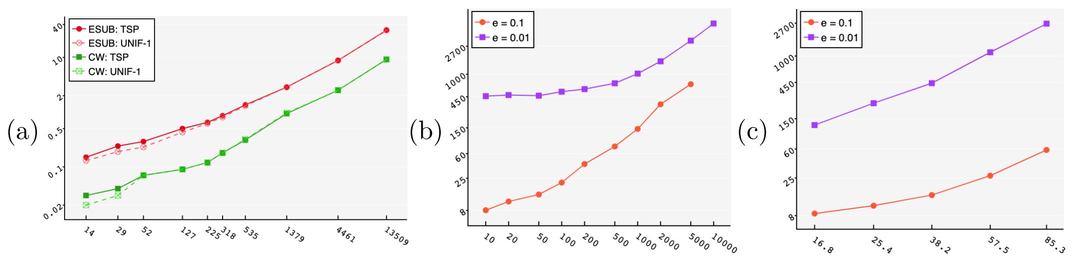


Figure 12 (a) A comparison of TSP data sets (filled shapes) with UNIF-1 (empty shapes, dashed curve) for both Algorithm 2 (● ESUB) and Algorithm 3 (🟩 CW). Fermat point with time as a function of n. (b),(c) n-ellipse on UNIF-2 with time as a function of (b) n and (c) the length of the n-ellipse. Two ε approximations with $\varepsilon = 0.1$ (●) and $\varepsilon = 0.01$ (🟪) have been computed.

6 Concluding Remarks

In this work, we mainly focused on finding ε-approximate Fermat points, in a strong sense $\|\widetilde{\boldsymbol{x}}^* - \boldsymbol{x}^*\| \leq \varepsilon$, which had not been considered before. This approximation can also be used to derive an ε-approximation of the Fermat radius. This was done using a simple-to-implement subdivision approach. All of our algorithms are certified in the sense of interval arithmetic. Moreover, we certified the famous point-sequence algorithm of Weiszfeld [58] to guarantee that it does find an ε-approximate Fermat point. We also designed an algorithm to construct ε-approximate n-ellipses. The simplicity and efficiency of our algorithms were evaluated experimentally for $d = 2$.

There are many directions for further research. One is to derive algorithmic complexity bounds. Our intuition regarding the time complexity of our algorithms was affirmed by the experimental runtime evaluation. Such bounds are rare for iterative numerical algorithms. There has been considerable success in the area of root isolation [10, 11] where the idea of “continuous amortization” should also apply here. Further, we expect the usage of the Hansen-Sengupta Newton operator to result in a speedup.

Regarding the construction of n-ellipses, it would be interesting to design an alternative algorithm based on curve-tracing. This could improve the runtime once a starting point on the n-ellipse is found.

Another direction is related to *Voronoi diagrams*. From one perspective, it is interesting to approximate the Voronoi diagram, where the sites are n-ellipses; so far only 2-ellipses have been studied [19]. From a different perspective, if the sites are sets of foci (each associated with a Fermat distance function) it is interesting to compute their Voronoi diagram, defined as the minimization diagram of the Fermat distance functions. This is a *min-sum* diagram in the context of cluster Voronoi diagrams, see e.g., [28, 44]. We believe that subdivision methods augmented with root boxes, similar to [5], would be applicable to these problems.

References

1 A. Karim Abu-Affash and Matthew J. Katz. Improved bounds on the average distance to the Fermat-Weber center of a convex object. *Information Processing Letters*, 109(6):329–333, 2009.

2 Mihai Badoiu, Sariel Har-Peled, and Piotr Indyk. Approximate clustering via core-sets. In *Proc. Symposium on Theory of Computing*, pages 250–257. ACM, 2002. doi:10.1145/509907.509947.

3 Chanderjit Bajaj. The algebraic degree of geometric optimization problems. *Discrete & Computational Geometry*, 3(2):177–191, 1988.

4 Amir Beck and Shoham Sabach. Weiszfeld's method: Old and new results. *Journal of Optimization Theory and Applications*, 164(1):1–40, 2015.

5 Huck Bennett, Evanthia Papadopoulou, and Chee Yap. Planar minimization diagrams via subdivision with applications to anisotropic Voronoi diagrams. *Computer Graphics Forum*, 35(5):229–247, 2016. doi:10.1111/cgf.12979.

6 Huck Bennett and Chee Yap. Amortized analysis of smooth quadtrees in all dimensions. *Computational Geometry*, 63:20–39, 2017. doi:10.1016/j.comgeo.2017.02.001.

7 Bhaswar B. Bhattacharya. On the Fermat-Weber point of a polygonal chain and its generalizations. *Fundamenta Informaticae*, 107(4):331–343, 2011.

8 Prosenjit Bose, Anil Maheshwari, and Pat Morin. Fast approximations for sums of distances, clustering and the Fermat-Weber problem. *Computational Geometry*, 24(3):135–146, 2003.

9 Luitzen Egbertus Jan Brouwer. Über Abbildung von Mannigfaltigkeiten. *Mathematische Annalen*, 71(1):97–115, 1911.

10 Michael Burr, Felix Krahmer, and Chee Yap. Continuous amortization: A non-probabilistic adaptive analysis technique. *Electronic Colloquium on Computational Complexity*, TR09(136), 2009.

11 Michael A. Burr. Continuous amortization and extensions: With applications to bisection-based root isolation. *Journal of Symbolic Computation*, 77:78–126, 2016. doi:10.1016/j.jsc.2016.01.007.

12 Paz Carmi, Sariel Har-Peled, and Matthew J. Katz. On the Fermat-Weber center of a convex object. *Computational Geometry*, 32(3):188–195, 2005. doi:10.1016/j.comgeo.2005.01.002.

13 Hui Han Chin, Aleksander Madry, Gary L. Miller, and Richard Peng. Runtime guarantees for regression problems. In *Proc. Innovations in Theoretical Computer Science*, pages 269–282. ACM, 2013.

14 Dietmar Cieslik. *Steiner minimal trees*, volume 23. Springer Science & Business Media, 2013.

15 Ernest J. Cockayne and Zdzislaw A. Melzak. Euclidean constructibility in graph-minimization problems. *Mathematics Magazine*, 42(4):206–208, 1969.

16 Michael B. Cohen, Yin Tat Lee, Gary L. Miller, Jakub Pachocki, and Aaron Sidford. Geometric median in nearly linear time. In *Proc. Symposium on Theory of Computing*, pages 9–21. ACM, 2016. doi:10.1145/2897518.2897647.

17 Theodorus Jozef Dekker. Finding a zero by means of successive linear interpolation. In *Constructive Aspects of the Fundamental Theorem of Algebra*, pages 37–48. Wiley Interscience, 1967.

18 Adrian Dumitrescu, Minghui Jiang, and Csaba D. Tóth. New bounds on the average distance from the Fermat-Weber center of a planar convex body. *Discrete Optimization*, 8(3):417–427, 2011. doi:10.1016/j.disopt.2011.02.004.

19 Ioannis Z. Emiris, Elias P. Tsigaridas, and George M. Tzoumas. The predicates for the Voronoi diagram of ellipses. In *Proc. Symposium on Computational Geometry*, pages 227–236. ACM, 2006.

20 Giovanni Francesco Fagnano. Problemata quaedam ad methodum maximorum et minimorum spectantia. *Nova Acta Eruditorum*, pages 281–303, 1775.

21 Sándor P. Fekete, Joseph S.B. Mitchell, and Karin Beurer. On the continuous Fermat-Weber problem. *Operations Research*, 53(1):61–76, 2005.

22 Dan Feldman and Michael Langberg. A unified framework for approximating and clustering data. In *Proc. Symposium on Theory of Computing*, pages 569–578. ACM, 2011.

23 Horst Hamacher and Zvi Drezner. Facility location: applications and theory. *Science & Business Media: Springer*, 2002.

24 Eldon R. Hansen. A multidimensional interval newton method. *Reliable Computing*, 12(4):253–272, 2006. doi:10.1007/s11155-006-9000-y.

25 Eldon R. Hansen and Saumyendra Sengupta. Bounding solutions of systems of equations using interval analysis. *BIT*, 21:203–211, 1981.

26 Sariel Har-Peled and Akash Kushal. Smaller coresets for k-median and k-means clustering. *Discrete & Computational Geometry*, 37(1):3–19, 2007.

27 Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *Proc. 36th Annual ACM Symposium on Theory of computing*, pages 291–300. ACM, 2004.

28 Daniel P. Huttenlocher, Klara Kedem, and Micha Sharir. The upper envelope of Voronoi surfaces and its applications. *Discrete & Computational Geometry*, 9(3):267–291, 1993.

29 I. Norman Katz. Local convergence in Fermat's problem. *Mathematical Programming*, 6(1):89–104, 1974.

30 Jakob Krarup and Steven Vajda. On Torricelli's geometrical solution to a problem of Fermat. *Journal of Management Mathematics*, 8(3):215–224, 1997.

31 Rudolf Krawczyk. Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken. *Computing*, 4(3):187–201, 1969. `doi:10.1007/BF02234767`.

32 Harold W. Kuhn. A note on Fermat's problem. *Mathematical programming*, 4(1):98–107, 1973.

33 Long Lin and Chee Yap. Adaptive isotopic approximation of nonsingular curves: the parameterizability and nonlocal isotopy approach. *Discrete & Computational Geometry*, 45(4):760–795, 2011. `doi:10.1007/s00454-011-9345-9`.

34 Long Lin, Chee Yap, and Jihun Yu. Non-local isotopic approximation of nonsingular surfaces. *Computer-Aided Design*, 45(2):451–462, 2012.

35 Luis Fernando Mello and Lucas Ruiz dos Santos. On the location of the minimum point in the Euclidean distance sum problem. *São Paulo Journal of Mathematical Sciences*, 12:108–120, 2018.

36 Ramon E. Moore. *Interval Analysis*, volume 4. Prentice-Hall Englewood Cliffs, NJ, 1966.

37 Kent E Morrison. The fedex problem. *The College Mathematics Journal*, 41(3):222–232, 2010.

38 Gyula Sz Nagy. Tschirnhaus'sche Eiflächen und Eikurven. *Acta Mathematica Academiae Scientiarum Hungarica*, 1(1):36–45, 1950.

39 Nguyen Mau Nam. The Fermat-Torricelli problem in the light of convex analysis. *ArXiv e-prints*, 2013. `arXiv:1302.5244v3`.

40 Karl Nickel. Triplex-algol and applications. *Interner Bericht des Instituts für Informatik der Universität Karlsruhe*, 1969.

41 Karl Nickel. On the Newton method in interval analysis. Technical report, Wisconsin University-Madison Mathematics Research Center, 1971.

42 Jiawang Nie, Pablo A. Parrilo, and Bernd Sturmfels. Semidefinite representation of the k-ellipse. In *Algorithms in algebraic geometry*, pages 117–132. Springer, 2008.

43 Lawrence M. Ostresh Jr. Convergence and descent in the Fermat location problem. *Transportation Science*, 12(2):153–164, 1978.

44 Evanthia Papadopoulou. The Hausdorff Voronoi diagram of point clusters in the plane. *Algorithmica*, 40(2):63–82, 2004.

45 Pablo A. Parrilo and Bernd Sturmfels. Minimizing polynomial functions. *Algorithmic and quantitative real algebraic geometry, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 60:83–99, 2003.

46 Maja Petrović, Bojan Banjac, and Branko Malešević. The geometry of trifocal curves with applications in architecture, urban and spatial planning. *Spatium*, pages 28–33, 2014.

47 Simon Plantinga and Gert Vegter. Isotopic approximation of implicit curves and surfaces. In *Proc. of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pages 245–254. ACM, 2004.

48 Helmut Ratschek and Jon Rokne. *Computer methods for the range of functions.* Horwood, 1984.

49 Gerhard Reinelt. TSPLIB - A traveling salesman problem library. *ORSA Journal on Computing*, 3(4):376–384, 1991.

50 Hanan Samet. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, 1990.

51 Junpei Sekino. n-ellipses and the minimum distance sum problem. *The American mathematical monthly*, 106(3):193–202, 1999.

52 Sergey P Shary. Krawczyk operator revised. *Novosibirsk, Institute of Computational Technologies, Rússia*, 2004.

53 Rudolf Sturm. Über den Punkt kleinster Entfernungssumme von gegebenen Punkten. *Journal für die reine und angewandte Mathematik*, 97:49–61, 1884.

54 Warwick Tucker. *Validated Numerics: A short intro to rigorous computations.* Princeton Press, 2011.

55 Ehrenfried Walther von Tschirnhaus. *Medicina Mentis Et Corporis.* Fritsch, Lipsiae, 1695. URL: `http://mdz-nbn-resolving.de/urn:nbn:de:bvb:12-bsb10008248-3`.

56 Cong Wang, Yi-Jen Chiang, and Chee Yap. On soft predicates in subdivision motion planning. *Computational Geometry: Theory and Applications.*, 48(8):589–605, 2015.

57 Alfred Weber. Über den Standort der Industrien. *English translation by CJ Friedrich (1929) Theory of the Location of Industries*, 1909.

58 Endre Weiszfeld. Sur le point pour lequel la somme des distances de n points donnés est minimum. *Tohoku Mathematical Journal, First Series*, 43:355–386, 1937.

59 Juan Xu and Chee Yap. Effective subdivision algorithm for isolating zeros of real systems of equations, with complexity analysis. In *Proc. International Symposium on Symbolic and Algebraic Computation*, pages 399–406. ACM, 2019.

60 Guoliang Xue and Yinyu Ye. An efficient algorithm for minimizing a sum of Euclidean norms with applications. *SIAM Journal on Optimization*, 7(4):1017–1036, 1997.