Certified Approximation Algorithms for the Fermat Point and n-Ellipses

Anonymous author
Anonymous affiliation

Anonymous author
Anonymous affiliation

Anonymous author
Anonymous affiliation

Anonymous author
Anonymous affiliation

Anonymous author
Anonymous affiliation

Abstract

Given a set A of n points in $\mathbb{R}^d$, the sum of distances is $\varphi(\boldsymbol{x}) = \sum_{\boldsymbol{a}\in A} \|\boldsymbol{x} - \boldsymbol{a}\|$. A classic problem in facility location, that dates back to 1643, is to find the *Fermat point* $\boldsymbol{x}^*$, the point that minimizes the function φ. In general, the Fermat point $\boldsymbol{x}^*$ cannot be computed exactly, so finding fast approximation algorithms has been of particular interest. In this work, we present algorithms to compute an ε-approximation of the Fermat point $\boldsymbol{x}^*$, that is, a point $\widetilde{\boldsymbol{x}}^*$ satisfying $\|\widetilde{\boldsymbol{x}}^* - \boldsymbol{x}^*\| < \varepsilon$. Our approximation scheme differs from the usual $\varphi(\widetilde{\boldsymbol{x}}^*) \leq (1+\varepsilon)\varphi(\boldsymbol{x}^*)$ approximation considered in the literature, which approximates the distance function. Our ε-approximation of the Fermat point directly implies an ε-approximation of the distance function, whereas the converse is not possible.

Our algorithms are based on the *subdivision* paradigm, which we enhance with Newton methods, used for *certification*, in the sense of *interval methods*, and for speed-ups. Moreover, we consider the problem of constructing *n-ellipses*, which are the r-level sets $\varphi^{-1}(r)$. The notion of an n-ellipse is a generalization of the classic (2-)ellipse and the circle (1-ellipse). Using the subdivision paradigm, we design an ε-isotopic approximation algorithm to compute n-ellipses in $\mathbb{R}^2$. We have implemented our algorithms and we provide an experimental analysis using different point configurations and heuristics for speed-ups. The obtained results suggest the practicality of our approaches especially in low dimensions and for small epsilon.

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.CVIT.2016.23

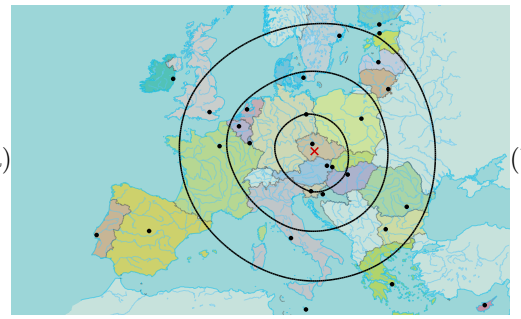
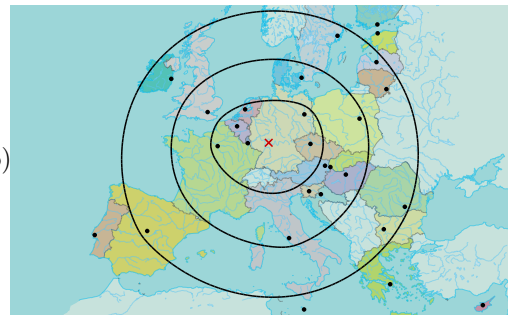
(a)  (b) 

Figure 1 The Fermat point of the 28 EU-capitals (pre-Brexit), highlighted with (×), along with three 28-ellipses of different radii. (a) Unweighted case. (b) Each capital has the weight of the country's population. The map is borrowed from `https://www.consilium.europa.eu`.

42nd Conference on Very Important Topics (CVIT 2016).
Editors: John Q. Open and Joan R. Access; Article No. 23; pp. 23:1–23:29
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

A classic problem in Facility Location, see e.g., [17, 35], 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 Fig. 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

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

where $\|\boldsymbol{x}\|$ is the Euclidean norm in $\mathbb{R}^d$. The global minimum value of φ is called the *Fermat radius* of A and denoted r^*; any point $\boldsymbol{x} \in \mathbb{R}^d$ that achieves this minimum, $\varphi(\boldsymbol{x}) = r^*$, is called a *Fermat point* and denoted $\boldsymbol{x}^* = \boldsymbol{x}^*(A)$. 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 can assume that A is not collinear, and so φ is a strictly convex function [27, 29].

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, such that the total transportation cost to the demand points is at most some specified r, as in Fig. 1.

The question of approximating the Fermat point is of great interest as its coordinates are the solution of a polynomial with exponentially high degree [3], thus when $n > 4$ the exact solution cannot be found in the general case. We address the problem of computing an ε-approximation $\widetilde{\boldsymbol{x}}^*$ to the Fermat point $\boldsymbol{x}^*$. This can be interpreted in 3 senses: (A) $\|\widetilde{\boldsymbol{x}}^* - \boldsymbol{x}^*\| \le \varepsilon$, (B) $\varphi(\widetilde{\boldsymbol{x}}^*) \le \varphi(\boldsymbol{x}^*) + \varepsilon$, and (C) $\varphi(\widetilde{\boldsymbol{x}}^*) \le (1 + \varepsilon)\varphi(\boldsymbol{x}^*)$. In Appendix A, we show that we can reduce (B) and (C) to (A), whereas the converse is not possible. In this paper we consider approximations in the sense (A) that are stronger. To the best of our knowledge, only approximations (B) and (C), have been considered in the literature (e.g., [7, 13]), which are essentially approximations of the Fermat radius.

In this work we introduce certified algorithms for approximating the Fermat point and n-ellipses, combining a subdivision approach with interval methods (cf. [26, 38]). The approach can be formalized in the framework of "soft predicates" [46]. 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 [19], we call it *the Fermat point problem*. Other common names are *Fermat-Weber problem* and *Geometric median problem*. Apart from the Facility Location application introduced by Weber [47], 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 [21].

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 [23] using a geometric construction. For $n = 4$, solutions were given by Fagnano [16] and Cieslik [11]. The first general method, for arbitrary n, is an iterative scheme proposed by Weiszfeld [48] in 1937. It was later

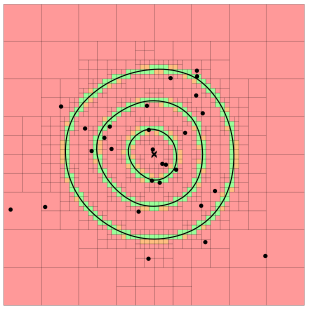
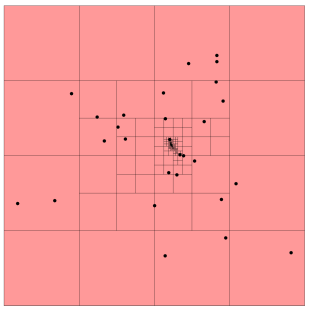
(a)  (b) 

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

corrected and improved by Kuhn [25] and Ostresh [35]; see Beck and Sabach [4] for a review. The method which is essentially a gradient descent, implies an iterative algorithm with no asymptotic runtime complexity, but which can behave quite well in practice.

A plethora of approximation algorithms for the Fermat point, in senses (B) and (C), can be found in the literature using various methods. There are algorithms based on semidefinite programming [36], on interior point methods [13, 50], via sampling [2, 13], geometric data structures [7] and coresets [20] among others [10, 18]. Moreover, special configurations of foci have been considered [6, 12], a continuous version of the problem [17], and the problem of finding the Fermat point of planar convex objects [1, 9, 15].

The literature on n-ellipses is smaller but equally old: Nagy [30] proved that n-ellipses are convex curves, dating them back to 1695 [45, p. 183]. Further, he characterizes the singular points of the n-ellipses as being either foci or the Fermat point. Another early work is by Sturm [43]. Sekino [41] showed that the 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. [34] 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 $\mathbb{R}^d$. To the best of our knowledge, this is the first algorithm to compute an ε-approximation of the actual Fermat point and not only of the Fermat radius. We also compute an ε-approximate n-ellipse; a problem not considered in computational literature before. Our contributions are summarized as follows:

- We introduce the first certified algorithms [28, 44] for approximating the Fermat point and n-ellipses.
- Our notion of ε-approximate Fermat point appears to be new; in contrast, several recent algorithmic work focuses on ε-approximation of the Fermat radius. Approximate Fermat radius can be reduced to approximate Fermat point; the converse reduction is unclear.
- Based on the *PV construction* [37], we design an algorithm, computing 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 Fig. 1).
- We implement our algorithms and experiment with different datasets and speedups. Each method is evaluated based on different values of the input parameters.

Various details and proofs which are omitted, due to lack of space, may be found in the Appendix.

2 Preliminaries

Vector variables are written in bold font: thus $\mathbf{0}$ is the origin of $\mathbb{R}^d$ and $\boldsymbol{x} = (x_1, \ldots, x_n)$. 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 $\nabla f(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x}))^T$ 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.

Our approach is fundamentally analytic rather than algebraic. As such, we consider analytic properties of a scalar function $f : \mathbb{R}^d \to \mathbb{R}$, mainly from the viewpoint of convex analysis [31, 27]. Although we are mainly interested in the case where f is the Fermat distance function for some weighted set A, it is important to see the general setting of our problem. For instance, this shows us that the Fermat point problem (resp., n-ellipsoid problem) reduces 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 $\mathbb{R}^d \setminus A$, since we assume A is non-collinear.

Most basic properties which we assert regarding the Fermat point are well-known and may be found in many of our references such as [25, 27, 31, 35, 48]. 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 $\left\|\nabla \varphi_{A\setminus \boldsymbol{a}}(\boldsymbol{a})\right\| \leq w(a)$. Thus, testing if the Fermat point $\boldsymbol{x}^*$ is in A can be done in $O(n^2)$ time (see Appendix H for details). If $\boldsymbol{x}^*$ is not one of the foci, then $\nabla f(\boldsymbol{x}^*) = 0$, and it can be reduced to general finding zeros of a system of equations (e.g., [49]). But 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.

P1. 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.

P2. Approximate Isotopic n-Ellipses: Given $\varepsilon > 0$, a 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.

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 [40]. A box can be specified by d intervals as $B = I_1 \times I_2 \times ... \times I_d$.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 $c \cdot B$ represents 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 boxes *(children)*, one for each orthant. We use $\textsc{split}_2$ to indicate that we do two successive levels of $\textsc{split}_1$, resuling in $(2^d)^2 = 4^d$ children. 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 [5]. Without maintaining smoothness, the amortized cost can be $\Omega(\log n)$ [5].

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., $P : \square\mathbb{R}^d \to$

[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 $\mathbb{R}^2$.

$\{\texttt{true}, \texttt{false}\}$. 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 [46] 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 $\texttt{true}$ or $\texttt{false}$. For instance, we call P an *exclusion predicate* if we associate the 0- and $\uparrow$-value with $\texttt{false}$ and the other values with $\texttt{true}$. For the *inclusion* predicate, we associate the 0-value with $\texttt{true}$, others with $\texttt{false}$. 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 $P(\boldsymbol{a}) = T(B_i)$ for i large enough.*

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 usually require some deeper principle such as the Brouwer fixed point theorem [8].

Interval arithmetic. We construct soft predicates using functions of the form $F : \square\mathbb{R}^d \to \square(\mathbb{R} \cup \{-\infty, \infty\})$ that approximates 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 $\square f$. E.g., f has an arithmetic expression E, we can simply evaluate E using interval arithmetic. More sophisticated methods may be needed for performance.

▶ **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 $\mathbf{0} \notin \boldsymbol{f}(B)$. 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 $T(B) : \mathbf{0} \notin \square\boldsymbol{f}(B)$. If $\boldsymbol{f} = (f_1, \ldots, f_m)$, then this reduces to $0 \notin \square f_i(B)$ for some $i = 1, \ldots, m$.

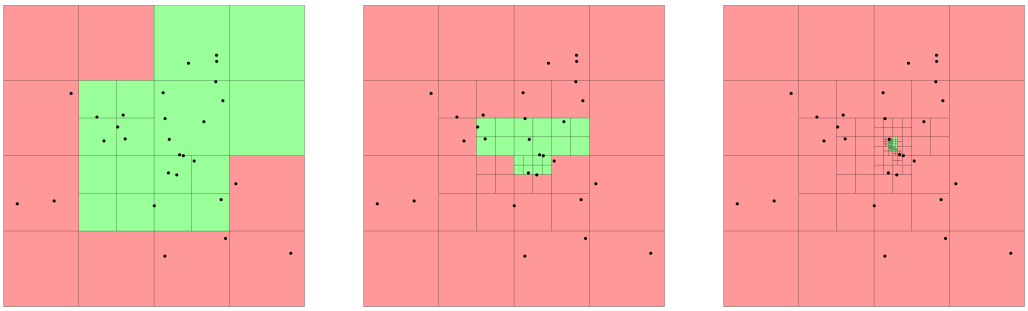


Figure 3 Different steps during the the execution of Algorithm 1. The red boxes cannot contain the Fermat point, whereas the green boxes may contain it.

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. $\boldsymbol{x}^* \notin A$, see Appendix H.

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 [25]. We use a function Initial-Box(A) which, in $O(n)$ time, computes an axis-aligned bounding box with corners having the minimum and maximum x, y coordinates.

We define the following exclusion predicate using interval arithmetic and Lemma 3. Refer to Appendix E for details on its soft version.

▶ **Definition 4.** *Given a box B, the* gradient exclusion predicate $C^{\nabla}(B)$ *returns true if and only if* $\mathbf{0} \notin \nabla\varphi(B)$.

▶ **Lemma 5.** *The soft gradient exclusion predicate $\square C^{\nabla}(B)$ is convergent, i.e., for any monotone sequence of boxes $(B_i)_{i\in\mathbb{N}}$ that converges to a point $\boldsymbol{p}$, the point $\boldsymbol{p}$ is not the Fermat point if and only if $\square C^{\nabla}(B_i) =$ `success` for large enough i.*

In Algorithm 1, using the exclusion predicate we discard boxes that are guaranteed not to contain $\boldsymbol{x}^*$ (**red** in Fig. 3) and we split boxes that might contain $\boldsymbol{x}^*$ (**green** in Fig. 3). While subdividing, we test whether we can already approximate $\boldsymbol{x}^*$ well enough by putting a bounding box around all the (green) boxes not excluded yet, using the following predicate.

▶ **Definition 6.** *Given a set of boxes Q which 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 $\widetilde{\boldsymbol{x}}^*$.

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.

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

Input: Foci set A, constant $\varepsilon > 0$. **Output:** Point $\widetilde{\boldsymbol{x}}^*$.

```
1 B_0 ← INITIAL-BOX(A);   Q ← QUEUE();   Q.PUSH(B_0);
2 while not C^ε(Q) do
3 |  B ← Q.POP();
4 |  if not □C^∇(B) then
5 |  |  Q.PUSH(SPLIT_1(B));
6 return x̃* ← Center of the bounding box of Q;
```

The Newton operator. Newton-type algorithms have been considered in the past but usually independently of other methods, thus suffering from lack of global convergence (see Appendix B for an example). Numerically, such methods face the *precision-control problem*. Our algorithm integrates subdivision with the Newton operator (an old idea that goes back to Dekker [14] in the 1960's), thus ensuring global convergence.

We want to find the Fermat point, i.e. the root of $\boldsymbol{f} = \nabla\varphi$. The Newton-type predicates are well-studied in the interval literature, and they have the form $N : \square\mathbb{R}^d \to \square\mathbb{R}^d$. There are two well-known versions, the simpler formula by Moore [28] and Nickel [32] 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 other formula by Krawzcyk [24, 42] 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 $J_{\boldsymbol{f}}^{-1}(m_B)$.

These Newton box operators have the following properties, which are consequences of Brouwer's Fixed Point Theorem [8, 33, 42].

1. $N(B) \subseteq B \Rightarrow \boldsymbol{x}^* \in N(B)$ **2.** $\boldsymbol{x}^* \in B \Rightarrow \boldsymbol{x}^* \in N(B)$ **3.** $N(B) \cap B = \emptyset \Rightarrow \boldsymbol{x}^* \notin B$

▶ **Definition 7.** *Given a box B, the* Newton inclusion predicate $C^N(B)$ *returns true if and only if $N(2B) \subseteq 2B$.*

As we work with a soft version of $\nabla\varphi$, we can only compute a *soft Newton inclusion predicate* $\square C^N(B)$, i.e., $\square N(2B) \subseteq 2B$. Observe that we use $2B$ instead of B; this is essential to avoid boundary issues. More precisely, if box B satisfies $\square N(B) \subseteq B$, then it must contain $\boldsymbol{x}^*$. But if $\boldsymbol{x}^*$ is on the boundary of box B, then $\square N(B) \subseteq B$ does not typically hold, and this issue persists even after splitting B.

We enhance Algorithm 1 by the soft inclusion predicate $\square C^N(B)$, as sketched in Algorithm 2. If $\square C^N(B)$ 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. In order to guarantee that boxes in the new queue are smaller than B, we do two recursive SPLIT operations of box $\square N(2B)$.

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

As in Algorithm 1 but replace line 5 with the following:

```
5.1 |  if □C^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,

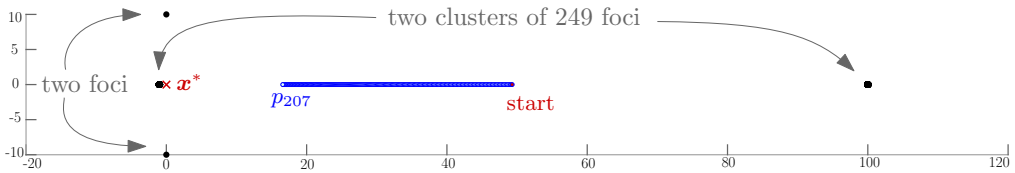


Figure 4 Weiszfeld's scheme on 500 foci, stopped when $\left\|\boldsymbol{p}_{i-1} - \boldsymbol{p}_i\right\| \leq 1/10$, after 207 steps (blue points). The distance $\|\boldsymbol{x}^* - \boldsymbol{p}_{207}\|$ can be arbitrarily big ($\|\boldsymbol{x}^* - \boldsymbol{p}_{207}\| > 15$ in this case).

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 (see Appendix F). 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.

We conclude the discussion on subdivision algorithm with (see Appendix F for details):

▶ **Theorem 8.** *Algorithm 1 and Algorithm 2 terminate and return an ε-approximate Fermat point. (Refer to Appendix F.)*

3.3 Certifying the Weiszfeld method

Weiszfeld's iterative method [25, 35, 48] describes a sequence of points that converges to the Fermat point $\boldsymbol{x}^*$, for any starting point. It is defined as follows:

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

Note that if $\boldsymbol{x} \notin A$, then $T(\boldsymbol{x})$ is simply $T(\boldsymbol{x}) = \dfrac{\sum_{i=1}^{n} w(\boldsymbol{a}) \frac{\boldsymbol{a}_i}{\|\boldsymbol{x}-\boldsymbol{a}_i\|}}{\sum_{i=1}^{n} w(\boldsymbol{a}) \frac{1}{\|\boldsymbol{x}-\boldsymbol{a}_i\|}}$.

The above implies an easy algorithm to approximate $\boldsymbol{x}^*$, by constructing a sequence of points $\boldsymbol{p}_i$ $(i = 0, 1, \ldots)$ where $\boldsymbol{p}_{i+1} = T(\boldsymbol{p}_i)$. This simple 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 Fig. 4 (see also Appendix B).

We augment this idea 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 Fig. 5. If $\square N(B) \subseteq B$, then the Fermat point $\boldsymbol{x}^*$ lies in $\square N(B)$, see Section 3.2. 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.

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 (see Appendix G for the choice 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 negatively effect the algorithm, 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., $\boldsymbol{p}_0 = \frac{1}{W} \sum_{\boldsymbol{a}\in A} w(\boldsymbol{a})\, \boldsymbol{a}$. Note that $\varphi(\boldsymbol{p}_0)$ itself is a 2-approximation of $r^* = \varphi(\boldsymbol{x}^*)$ [13].

▶ **Theorem 9.** *Algorithm 3 terminates and returns an ε-approximate Fermat point. (Refer to Appendix G.)*

With respect to the runtime, the point sequence $T(\boldsymbol{x})$ converges linearly in ε towards $\boldsymbol{x}^*$ [22] but in order for Algorithm 3 to terminate the test $\square N(B) \subseteq B$ must succeed. Similar

Algorithm 3 Certified Weiszfeld (CW)

Input: Foci set A, constant $\varepsilon > 0$. **Output:** Point $\widetilde{\boldsymbol{x}}^*$.

$\boldsymbol{p} \leftarrow \boldsymbol{p}_0$; $l \leftarrow \varepsilon$;
while TRUE **do**
  $B \leftarrow$ Box $B(m_B = p,\ \omega(B) = l)$;
  if $\square N(B) \subseteq B$ **then** // Fig. 5a
    return $\widetilde{\boldsymbol{x}}^* \leftarrow \boldsymbol{p}$;
  else if $\square N\left(\frac{B}{10}\right) \cap \frac{B}{10} = \emptyset$ **then** // Fig. 5b
    $l \leftarrow \min\{10 \cdot l, \varepsilon\}$;
  else $l \leftarrow \frac{1}{10} \cdot l$; // Fig. 5c
  $\boldsymbol{p} \leftarrow T(\boldsymbol{p})$;

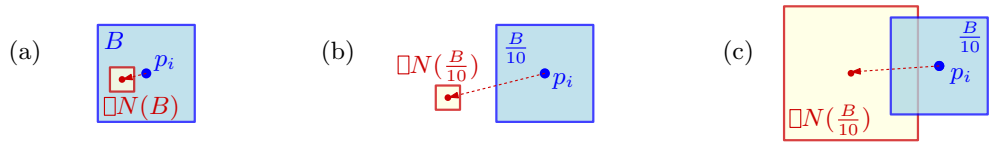


Figure 5 The three cases of Algorithm 3. (a) $N(B) \subseteq B$, (b) $N(B) \cap B = \emptyset$ and (c) $N(B) \cap B \neq \emptyset$.

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.

4 Approximating n-ellipses

In this section, we describe an algorithm to construct approximate n-ellipses, based on the subdivision paradigm. The complete algorithm, with all details, can be found in Appendix D.

The Plantinga and Vegter (PV) construction [37] approximates the zero set of a function $F : \mathbb{R}^d \to \mathbb{R}$ where $d \in \{2, 3\}$. Assuming that the curve $S = F^{-1}(0)$ is regular, i.e., the gradient ∇F is non-zero at every point of S, their approximation is isotopic to S. Our goal is to use this construction to get an approximation of an n-ellipse by using $F(p) = \varphi(p) - r$. We assume that the radius r of the n-ellipse is bigger than the Fermat radius r^*. In the next definition we use the notation $\langle \cdot, \cdot \rangle$ for the scalar product.

▶ **Definition 10.** *Given a box B we define the following tests:*

1. *The* exclusion test *is defined by $C_0^{Ex}(B) =$ `success` if and only if $0 \notin \square F$.*
2. *The* inclusion test *is defined by $C_0^{In}(B) =$ `success` if and only if F evaluated at the corners of B admits negative and positive values.*
3. *The* normal variation test *is defined by $C_1(B) =$ `success` if and only if $\langle \square \nabla F, \square \nabla F \rangle > 0$.*

The success of the test $C_0^{\mathrm{Ex}}(B)$ implies that the n-ellipse does not pass through B. The success of the test $C_0^{\mathrm{In}}(B)$ implies that the n-ellipse passes through B. The success of the test $C_1(B)$ implies that the angle between the gradient of any two points in B is at most 90°. This implies that the n-ellipse does not have a big curvature and F is monotone in either x- or y-direction within the box. In Fig. 6a, boxes are: **red** if they pass the C_0^{Ex} test, **green** if they pass both C_1 and C_0^{In}, **orange** if the pass only C_1, 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 [41]; in that case a direct PV construction is not possible. To tackle this problem we develop a variation, where we

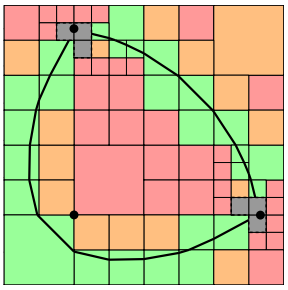
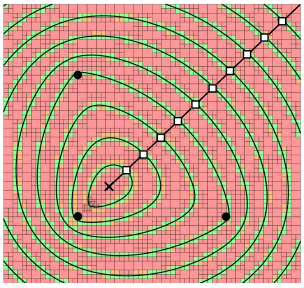
(a)  (b) 

Figure 6 (a) A 3-ellipse passing through two foci. Components of gray boxes (temporarily) surround the foci. (b) A 3-elliptic contour plot with 10 contour lines, *nicely distributed* in space.

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 C_0^{Ex} (**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 C_1 and have width smaller than $\varepsilon/2$ (**green** or **orange**): We immediately draw edges in each of these boxes.
3. The remaining boxes (**gray**): Such boxes occur near foci and need more careful attention, as we cannot apply the standard PV construction. Instead, for each connected component of gray boxes, we check if a set of conditions is satisfied. If so, edges are immediately drawn, otherwise the boxes are further split for classification.

Contour Plotting. As an application, we can use the above 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, 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 Fig. 6b.

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; refer to Appendix I for more details. 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 Fig. 8(a) and (b). Despite their similarity, the two datasets present strong differences. As we later see, Unif-2 is significantly difficult to

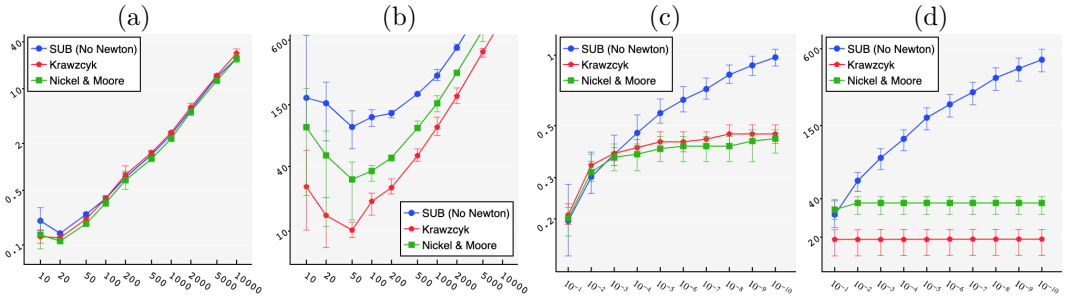


Figure 7 Comparing Newton operators: (a)-(b) Time as a function of n, with $\varepsilon = 10^{-4}$. (c)-(d) Time as a function of ε with $n = 100$. (a)-(c) UNIF-1 datasets. (b)-(d) UNIF-2 datasets.

solve, and UNIF-1 resembles nicely real-world datasets. We experimented with more types of synthetics datasets but the results are similar to UNIF-1 or UNIF-2; see Appendix I.

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 Fig. 7. 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 smaller box $N(B)$, 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 Fig. 8(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.

$\varepsilon = 10^{-3}$, $n =$	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

Real Datasets Inspired by the applications in facility location we chose to experiment with instances of the well-known *Traveling Salesman Person Library* [39] 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, UNIF-1 are realistic datasets for the evaluation of different algorithms. In our experiments illustrated in Fig. 10(a), 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. The similarity in 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 Fig. 9. The running time of all methods shows a linear dependence 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. In contrast, Algorithm 2 takes more time in the subdivision phase, but once the Newton tests succeeds, the algorithm terminates very fast. 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.

n-ellipses. Finally, we evaluated the runtime of n-ellipses algorithm. It 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. We summarize our experiments in the following figure. In Fig. 10(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 Fig. 10(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 $3/2$.

Concluding remarks. In this work, we 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 was done using a simple-to-implement subdivision approach. All of our algorithm are certified in the sense of interval arithmetic. Moreover, we certified the famous point-sequence algorithm of Weiszfeld [48] to guarantee that it finds an ε-approximate Fermat point. Especially for *difficult* instances and very small ε the Newton-based subdivision algorithm is preferable, due to its eventual quadratic convergence. For high dimensions, the point-sequence algorithm would probably be favourable, due to the dependency of the subdvision methods on d.

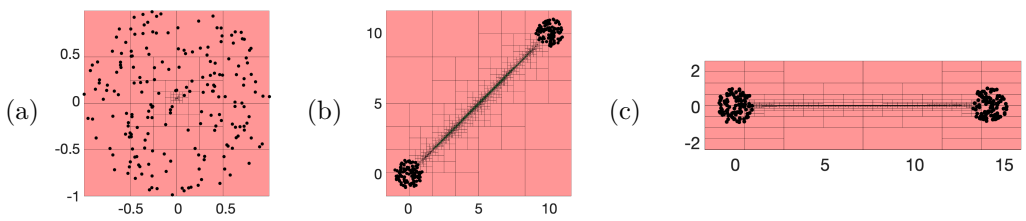


Figure 8 A box subdivision for $n = 200$ foci: (a) Unif-1, (b) Unif-2 and (c) Unif-2 after PCA.

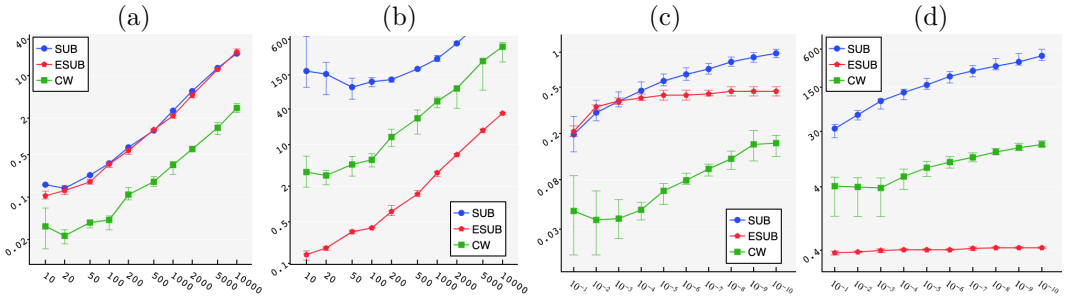


Figure 9 Comparing the Fermat point algorithms: (a)-(b) Time as a function of n, with $\varepsilon = 10^{-4}$. (c)-(d) Time as a function of ε with $n = 100$. (a)-(c) UNIF-1 datasets. (b)-(d) UNIF-2 datasets.

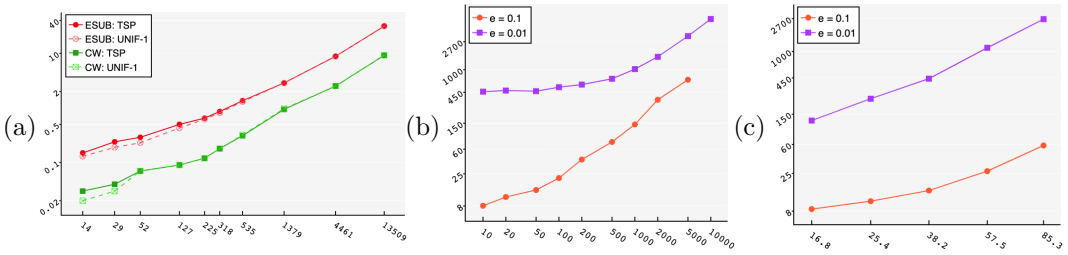


Figure 10 (a) Fermat point with time as a function of n, with $\varepsilon = 10^{-4}$. (b)-(c) n-ellipse on UNIF-2 with time as a function of (b) n and (c) the length of the n-ellipse.

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.
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 and Chee Yap. Amortized analysis of smooth quadtrees in all dimensions. *Computational Geometry*, 63:20–39, 2017.
6. Bhaswar B. Bhattacharya. On the Fermat-Weber point of a polygonal chain and its generalizations. *Fundamenta Informaticae*, 107(4):331–343, 2011.
7. 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.
8. Luitzen Egbertus Jan Brouwer. Über Abbildung von Mannigfaltigkeiten. *Mathematische Annalen*, 71(1):97–115, 1911.
9. 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.
10. 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.
11. Dietmar Cieslik. *Steiner minimal trees*, volume 23. Springer Science & Business Media, 2013.
12. Ernest J. Cockayne and Zdzislaw A. Melzak. Euclidean constructibility in graph-minimization problems. *Mathematics Magazine*, 42(4):206–208, 1969.
13. 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.
14. 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.
15. 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.
16. Giovanni Francesco Fagnano. Problemata quaedam ad methodum maximorum et minimorum spectantia. *Nova Acta Eruditorum*, pages 281–303, 1775.
17. Sándor P Fekete, Joseph SB Mitchell, and Karin Beurer. On the continuous Fermat-Weber problem. *Operations Research*, 53(1):61–76, 2005.
18. 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.
19. Horst Hamacher and Zvi Drezner. Facility location: applications and theory. *Science & Business Media: Springer*, 2002.
20. Sariel Har-Peled and Akash Kushal. Smaller coresets for k-median and k-means clustering. *Discrete & Computational Geometry*, 37(1):3–19, 2007.
21. 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.
22. I. Norman Katz. Local convergence in Fermat's problem. *Mathematical Programming*, 6(1):89–104, 1974.
23. Jakob Krarup and Steven Vajda. On Torricelli's geometrical solution to a problem of Fermat. *Journal of Management Mathematics*, 8(3):215–224, 1997.

24 Rudolf Krawczyk. Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken. *Computing*, 4(3):187–201, 1969.

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

26 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.

27 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.

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

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

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

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

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

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

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

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

36 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.

37 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.

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

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

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

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

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

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

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

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

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

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

48 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.

499 **49** Juan Xu and Chee Yap. Effective subdivision algorithm for isolating zeros of real systems
500 of equations, with complexity analysis. In *Proc. International Symposium on Symbolic and*
501 *Algebraic Computation*, pages 399–406. ACM, 2019.
502 **50** Guoliang Xue and Yinyu Ye. An efficient algorithm for minimizing a sum of Euclidean norms
503 with applications. *SIAM Journal on Optimization*, 7(4):1017–1036, 1997.

Appendix content

The appendix is organized as follows.

A Notions of ε-approximations of the Fermat point

This section compares the three different notions of ε-approximation of the Fermat point:
(A) $\left\|\widetilde{\boldsymbol{x}}^* - \boldsymbol{x}^*\right\| \leq \varepsilon$
(B) $\varphi(\widetilde{\boldsymbol{x}}^*) \leq \varphi(\boldsymbol{x}^*) + \varepsilon$
(C) $\varphi(\widetilde{\boldsymbol{x}}^*) \leq (1+\varepsilon)\varphi(\boldsymbol{x}^*)$

The following lemmas show that notion (A) is stronger than notions (B) and (C). There is no function f in n, ε and W such that an $f(\varepsilon, n, W)$ approximate Fermat point in sense (B) or (C) implies that it is an ε approximate Fermat point in the sense (A).

▶ **Lemma 11.** *For any $\varepsilon > 0$ there exists an instance of 4 foci such that an ε-approximation of the Fermat point in the sense (B) or (C) can have distance 1 to the Fermat point.*

Proof of Lemma 11 (Version 1). Let $\varepsilon > 0$ and choose $c \leq \frac{\varepsilon}{2\sqrt{2}-2}$. Consider the foci $\boldsymbol{a}_1 = (1,0)$ $\boldsymbol{a}_2 = (0,1)$ $\boldsymbol{a}_3 = (-1,0)$ $\boldsymbol{a}_4 = (0,-1)$ with weights $w(\boldsymbol{a}_1) = w(\boldsymbol{a}_3) = 1$ and $w(\boldsymbol{a}_2) = w(\boldsymbol{a}_4) = c$ for which the Fermat point is $\boldsymbol{x}^* = (0,0)$ for symmetry reasons, and hence $\varphi(\boldsymbol{x}^*) = 2 + 2c$, see Figure 11. The point $p = (1,0)$ is an ε-approximation of $\boldsymbol{x}^*$ in the sense (B) and (C), see Inequalities 2 and 3, but it has a distance of 1 to $\boldsymbol{x}^* = (0,0)$.

$$\varphi(p) = \sum_{i=1}^{4} w(\boldsymbol{a}_i)\|p - \boldsymbol{a}_i\| = 2 + 2\sqrt{2}c \leq 2 + 2c + \varepsilon$$
$$\leq \varphi(\boldsymbol{x}^*) + \varepsilon \tag{2}$$
$$\leq (1+\varepsilon)\varphi(\boldsymbol{x}^*) \tag{3}$$

The last inequality holds because $\varphi(\boldsymbol{x}^*) = 2 + 2c > 1$. ◀

Proof of Lemma 11 (Version 2). Let $\varepsilon > 0$, and 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 weight for which the Fermat point is $\boldsymbol{x}^* = (1,0)$ for symmetry reasons, and hence $\varphi(\boldsymbol{x}^*) = 4\sqrt{1+h^2}$. See Figure 11. The point $p = (2,0)$ is an ε-approximation of $\boldsymbol{x}^*$ in the sense (B) and (C), see

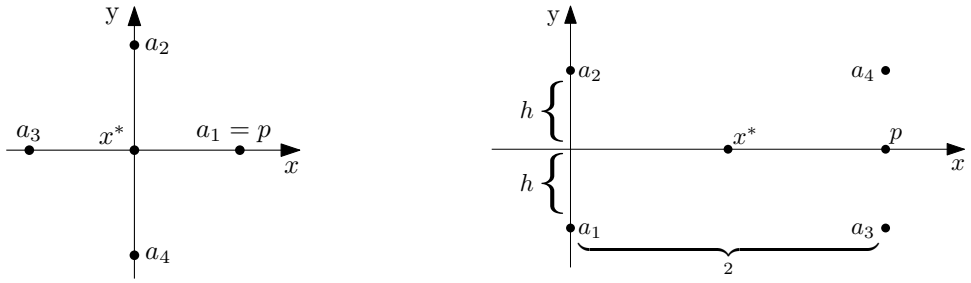


Figure 11 A good approximation of the Fermat point in sense (B) or (C) does not imply a good approximation in sense (A).

Inequalities 4 and 5, but it has a distance of 1 to $\boldsymbol{x}^* = (1, 0)$.

$$\varphi(p) = \sum_{i=1}^{4} \|p - \boldsymbol{a}_i\| = 2\sqrt{4+h^2} + 2h \leq 4\sqrt{1+h^2} + \varepsilon$$

$$\leq \varphi(\boldsymbol{x}^*) + \varepsilon \tag{4}$$

$$\leq (1+\varepsilon)\varphi(\boldsymbol{x}^*) \tag{5}$$

The last inequality holds because $\varphi(\boldsymbol{x}^*) = 4\sqrt{1+h^2} > 1$. ◀

▶ **Lemma 12.** *An ε-approximation $\widetilde{\boldsymbol{x}}^*$ of $\boldsymbol{x}^*$ in the sense $\left\|\widetilde{\boldsymbol{x}}^* - \boldsymbol{x}^*\right\| \leq \varepsilon$ is also a $W\varepsilon$-approximation in the sense $\varphi(\widetilde{\boldsymbol{x}}^*) \leq \varphi(\boldsymbol{x}^*) + W\varepsilon$.*

Proof. By the triangle inequality we have

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

◀

▶ **Lemma 13.** *An ε-approximation $\widetilde{\boldsymbol{x}}^*$ of $\boldsymbol{x}^*$ in the sense $\varphi(\widetilde{\boldsymbol{x}}^*) \leq \varphi(\boldsymbol{x}^*) + \varepsilon$ is also a $\frac{2\varepsilon}{\varphi(g)}$-approximation in the sense $\varphi(\widetilde{\boldsymbol{x}}^*) \leq (1 + \frac{2\varepsilon}{\varphi(g)})\varphi(\boldsymbol{x}^*)$, where g is the center of gravity of the foci.*

Proof. The center of gravity g is a 2-approximation of the Fermat radius r^* (see [13]), i.e. $\varphi(\boldsymbol{x}^*) \geq \frac{1}{2}\varphi(g)$.

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

◀

B Problems of two non-subdivision based approaches

B.1 Pure Newton: Lack of global convergence

In this section we describe an instance, in which the point Newton method fails to converge to the Fermat point. Consider the following 10 foci in $\mathbb{R}^3$:

$a_1 = (0.38462, 0.58299, 0.25181)$
$a_2 = (0.29044, 0.61709, 0.26528)$
$a_3 = (0.82438, 0.98266, 0.73025)$
$a_4 = (0.34388, 0.58407, 0.10777)$
$a_5 = (0.90631, 0.87965, 0.81776)$
$a_6 = (0.26073, 0.59436, 0.022513)$
$a_7 = (0.42526, 0.31272, 0.16148)$
$a_8 = (0.17877, 0.42289, 0.094229)$
$a_9 = (0.59852, 0.47092, 0.69595)$
$a_{10} = (0.69989, 0.63853, 0.033604)$

each with a weight of 1. If we start with the center of mass $p_0 = \frac{1}{n}\sum_{\boldsymbol{a}\in A}\boldsymbol{a}$ then for $f = \nabla\varphi$ the pure Newton method $p_{i+1} = p_i - J_f^{-1}(p_i)\cdot f(p_i)$ does not terminate. In particular, for big enough i the sequence keeps revisiting the following 4 points:

$p_{4i} = (0.40089, 0.58085, 0.23502)$
$p_{4i+1} = (0.37393, 0.58077, 0.25124)$
$p_{4i+2} = (0.43552, 0.58899, 0.24779)$
$p_{4i+3} = (0.32493, 0.56753, 0.22338)$

B.2 Weiszfeld's point sequence scheme: Weak stopping criterion

▶ Remark 14. The example in Figure 4 shows 500 foci (black points), two of which have coordinates $(0, 10)$ and $(0, -10)$ and the others are two very dense clusters of 249 foci around $(0, 0)$ and $(100, 0)$, respectively. This configuration yields a Fermat point $\boldsymbol{x}^* \simeq (0.019, 0)$ (red 'x'), which is very far away from the center of mass $\simeq (49.3, 0)$, used as a start point (red point). Setting $\varepsilon = 1/10$ Weiszfeld's scheme (blue points) was stopped after 207 steps, because the step length at iteration i was smaller than ε, i.e. $\|\boldsymbol{p}_{i-1} - \boldsymbol{p}_i\| \leq 1/10$. However, the distance $\|\boldsymbol{x}^* - \boldsymbol{p}_{207}\| > 15$ is still very big. This example shows that a very small current step length is no indicator for the Fermat point to be close.

C Details on box approximations

In this section we need to introduce a bit more notation. Let $()^T$ denote the transpose operation. The *Hessian* $\nabla^2 f : \mathbb{R}^d \to \mathbb{R}^{d\times d}$ of f is given by the matrix $\nabla^2 f(\boldsymbol{x}) = (f_{ij}(\boldsymbol{x}))_{i,j=1}^d$ where $f_{ij} = \partial_i\partial_j f$.

The box approximations described in this section generalize for higher dimensions. For simplicity we describe them in $\mathbb{R}^2$.

C.1 Box approximation of the gradient $\nabla\varphi$

For any point $p = (p_x, p_y)^T$, let $\sin(p) := p_x/\|p\|$ and $\cos(p) := p_y/\|p\|$. Clearly,

$$\nabla\varphi(p) = \begin{pmatrix} \sum_{\boldsymbol{a}\in A} w(\boldsymbol{a})\sin(p-\boldsymbol{a}) \\ \sum_{\boldsymbol{a}\in A} w(\boldsymbol{a})\cos(p-\boldsymbol{a}) \end{pmatrix}.$$

We want to develop formulas for $\sin(B-\boldsymbol{a})$ and $\cos(B-\boldsymbol{a})$. By symmetry, we consider only $\sin(B-\boldsymbol{a})$. The four corners of B are given by $m_B + \frac{\omega(B)}{2}\begin{pmatrix}\pm 1\\ \pm 1\end{pmatrix}$. Let $Corners(B)$ denote

this set of four points. Then

$$\sin(B-\boldsymbol{a}) = \begin{cases} [-1,1] & \text{if } \boldsymbol{a} \in B, \\ [\min(\sin(Corners(B)-\boldsymbol{a})),1] & \text{if } \boldsymbol{a} \text{ is below } B, \\ [-1,\max(\sin(Corners(B)-\boldsymbol{a}))] & \text{if } \boldsymbol{a} \text{ is above } B, \\ [\min(\sin(Corners(B)-\boldsymbol{a})),\max(\sin(Corners(B)-\boldsymbol{a}))] & \text{else.} \end{cases}$$

In other words, $\sin(B-\boldsymbol{a})$ can be computed from the sinus of at most four angles. Similarly for $\cos(B-\boldsymbol{a})$.

Now, we extend these formulas: for instance,

$$\square\nabla\varphi(B) = \begin{pmatrix} \sum_{\boldsymbol{a}\in A} w(\boldsymbol{a})\sin(B-\boldsymbol{a}) \\ \sum_{\boldsymbol{a}\in A} w(\boldsymbol{a})\cos(B-\boldsymbol{a}) \end{pmatrix}.$$

The following is immediate:

▶ **Lemma 15.** *$\square\nabla\varphi$ is a soft predicate, i.e. it is conservative and convergent.*

Evaluating $\square\nabla\varphi$ as described above gives a very good soft version of $\nabla\varphi$ but takes exponential time in d. If the number of dimensions is higher, one can instead directly apply interval arithmetic to compute such soft versions in $O(nd)$ time.

C.2 Box approximation of φ

We use the concept of a Lipschitz constant in order to derive a box approximation of φ. We call $L(B)$ a Lipschitz constant for box B if $\forall p,q \in B: \quad |\varphi(p)-\varphi(q)| \le L(B)\cdot\|p-q\|$. A trivial Lipschitz constant is W because it bounds the maximum length of the gradient:

$$\|\nabla\varphi(p)\| \le \sum_{\boldsymbol{a}\in A} w(\boldsymbol{a}) \left\| \begin{pmatrix} \sin(p-\boldsymbol{a}) \\ \cos(p-\boldsymbol{a}) \end{pmatrix} \right\| = \sum_{\boldsymbol{a}\in A} w(\boldsymbol{a}) = W$$

▶ **Definition 16.** *We use $\square\varphi(B)$ as a box approximation of $\varphi(B)$ where:*

$$\square\varphi(B) = [\varphi(m_B) - L(B)\cdot r_B, \varphi(m_B) + L(B)\cdot r_B]$$

▶ **Lemma 17.** *$\square\varphi(B)$ is a soft predicate, i.e. it is conservative and convergent.*

Proof. The $L(B)$ is a Lipschitz constant of φ on box B, i.e. $\forall p \in B$:

$$|\varphi(p) - \varphi(m_B)| \le L(B)\cdot r_B$$

This implies $\varphi(p) \in [\varphi(m_B) - L\cdot r_B, \varphi(m_B) + L\cdot r_B]$ and hence $\square\varphi(B)$ is conservative. Let B_i be a sequence of boxes, which converges to a point. This implies $r_{B_i} \to 0$. The Lipschitz constant L can be bounded from above by W. Thus, $\omega(\square\varphi(B_i)) \le 2W\cdot r_{B_i} \to 0$. ◀

Using the Lipschitz constant W within all boxes B can result in very bad box approximations. Consider boxes near the Fermat point, for which the gradient of φ at every point is almost 0. In the rest of this section we compute a better Lipschitz constant for each individual box.

We partition the set of foci $A = A_1 \dot\cup A_2$ into foci which are "far" or "close" to box B:

$$\forall \boldsymbol{a} \in A_1: \quad \left\| \begin{pmatrix} \sin(B-\boldsymbol{a}) \\ \cos(B-\boldsymbol{a}) \end{pmatrix} \right\| \subset [-1,1] \quad \text{and } \forall \boldsymbol{a} \in A_2: \quad \left\| \begin{pmatrix} \sin(B-\boldsymbol{a}) \\ \cos(B-\boldsymbol{a}) \end{pmatrix} \right\| \not\subseteq [-1,1]$$

The length of an interval vector $I = (I_x, I_y)$ is computed by $\|I\| = \sqrt{I_x^2 + I_y^2}$, where we define the square root of an interval $J = [J_1, J_2]$ by:

$$\sqrt{J} = \begin{cases} \left[0, \sqrt{\max\{|J_1|, |J_2|\}}\right] & \text{if } 0 \in J \\ \left[\sqrt{\min\{|J_1|, |J_2|\}}, \sqrt{\max\{|J_1|, |J_2|\}}\right] & \text{if } 0 \notin J. \end{cases}$$

A box approximation of the length of the gradient of φ can then be achieved by:

$$\square\, \|\nabla\varphi(B)\| = \left\| \sum_{\boldsymbol{a} \in A_1} w(\boldsymbol{a}) \begin{pmatrix} \sin(B - \boldsymbol{a}) \\ \cos(B - \boldsymbol{a}) \end{pmatrix} \right\| + \left[-\sum_{\boldsymbol{a} \in A_2} w(\boldsymbol{a}), \sum_{\boldsymbol{a} \in A_2} w(\boldsymbol{a}) \right]$$

The maximal length of the gradient within box B is a Lipschitz constant of φ within box B. Hence, $L(B) = \max \square\, \|\nabla\varphi(B)\|$ can be used as Lipschitz constant for box B.

C.3 Box approximation of the Hessian $\nabla^2\varphi$

For any $p \in \mathbb{R}^2 \setminus A$ it holds:

$$\nabla^2\varphi(p) = \begin{pmatrix} \sum_{\boldsymbol{a} \in A} w(\boldsymbol{a}) \frac{(p_y - \boldsymbol{a}_y)^2}{\|p - \boldsymbol{a}\|^3} & -\sum_{\boldsymbol{a} \in A} w(\boldsymbol{a}) \frac{(p_x - \boldsymbol{a}_x)(p_y - \boldsymbol{a}_y)}{\|p - \boldsymbol{a}\|^3} \\ -\sum_{\boldsymbol{a} \in A} w(\boldsymbol{a}) \frac{(p_x - \boldsymbol{a}_x)(p_y - \boldsymbol{a}_y)}{\|p - \boldsymbol{a}\|^3} & \sum_{\boldsymbol{a} \in A} w(\boldsymbol{a}) \frac{(p_x - \boldsymbol{a}_x)^2}{\|p - \boldsymbol{a}\|^3} \end{pmatrix}.$$

▶ **Definition 18.** *We define the box approximation of $\nabla^2\varphi(B)$, denoted $\square\nabla^2\varphi(B)$ as follows.*

$$\square\nabla^2\varphi(B) = \begin{pmatrix} \sum_{\boldsymbol{a} \in A} w(\boldsymbol{a}) \frac{(B_y - \boldsymbol{a}_y)^2}{[\|m_B - \boldsymbol{a}\| - r, \|m_B - \boldsymbol{a}\| + r]^3} & -\sum_{\boldsymbol{a} \in A} w(\boldsymbol{a}) \frac{(B_x - \boldsymbol{a}_x)(B_y - \boldsymbol{a}_y)}{[\|m_B - \boldsymbol{a}\| - r, \|m_B - \boldsymbol{a}\| + r]^3} \\ -\sum_{\boldsymbol{a} \in A} w(\boldsymbol{a}) \frac{(B_x - \boldsymbol{a}_x)(B_y - \boldsymbol{a}_y)}{[\|m_B - \boldsymbol{a}\| - r, \|m_B - \boldsymbol{a}\| + r]^3} & \sum_{\boldsymbol{a} \in A} w(\boldsymbol{a}) \frac{(B_x - \boldsymbol{a}_x)^2}{[\|m_B - \boldsymbol{a}\| - r, \|m_B - \boldsymbol{a}\| + r]^3} \end{pmatrix}.$$

The following lemma is again immediate.

▶ **Lemma 19.** *$\square\nabla^2\varphi$ is conservative and convergent.*

D Approximating n-ellipses

D.1 Algorithm description

Our algorithm is described in Algorithm 4. The details skipped from the main part follow. More specifically, we simultaneously subdivide boxes and construct pieces of the n-ellipses. In the subdivision part we classify boxes in 3 categories:

(A1) boxes, which satisfy C_0^{Ex} (shown in **red**).
(A2) boxes, which satisfy C_1 and are smaller than $\varepsilon/2$ (shown in **green** or **orange**).
(A3) the remaining boxes (shown in **gray**).

(A1) boxes. These do not contain any piece of the n-ellipse, so they do not need to be further considered and are excluded in line 6.

(A2) boxes. In contrast to the normal PV construction, where the curve is drawn in the end, we immediately start drawing edges in each (A2) box in line 8. In order to draw edges in a box B we look at the sign of function F at B's corners and decide accordingly. Later, during 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.

Algorithm 4 Approximating an n-ellipse

Input: Foci set A, radius r, constant ε, box B_0. **Output:** Curve E.

1 $Q \leftarrow$ QUEUE(); Q.PUSH(B_0);
2 **while** $Q \neq \emptyset$ **do**
3 $\quad Q_{new} \leftarrow$ QUEUE();
4 $\quad$ **while** $Q \neq \emptyset$ **do**
5 $\quad\quad B \leftarrow Q$.POP();
6 $\quad\quad$ **if** ***not*** $C_0^{Ex}(B)$ **then**
7 $\quad\quad\quad$ **if** $C_1(B)$ ***and*** $\omega(B) < \varepsilon/2$ **then**
8 $\quad\quad\quad\quad E_{\cap B} \leftarrow$ ONLINE-PV(B);
9 $\quad\quad\quad$ **else** Q_{new}.PUSH(SPLIT$_4(B)$);
10 $\quad Q \leftarrow$ CONNECTED-COMPONENTS-ANALYSIS(Q_{new});
11 **return** E;

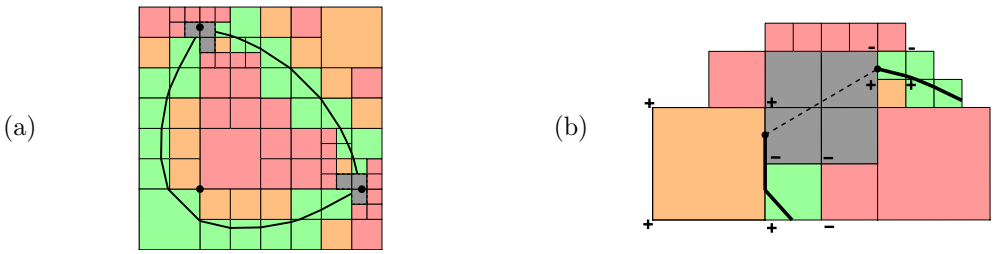


Figure 12 (a) A 3-ellipse passing through two foci. A connected component of gray boxes surrounds these foci. (b) If a component satisfies (B1) - (B3) we connect the two ingoing edges with an edge.

(A3) boxes. These boxes need more careful attention, as we cannot do the standard PV construction within those, and we therefore treat them separately in line 10. First given a set of gray boxes we distinguish them in connected components. This can be easily done in $O(|Q_{new}|)$ time using a DFS-type of algorithm. Then, to take a correct decision we want each component K_i to satisfy the following properties:

(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 a component 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 Fig. 12. Otherwise, the children of the boxes of K_i are put back in Q and then we start again classifying into (A1), (A2) or (A3) and so on.

Algorithm 5 Connected component analysis

Input: Queue Q_{new}, set A, constant ε. **Output:** Queue Q.

1 $Q \leftarrow \emptyset$; $\{K_1, ..., K_n\} \leftarrow$ connected components of Q_{new};
2 **for** $i = 1$ *to* n **do**
3 $\quad$ **if** K_i *does not satisfy (B1)∧(B2)∧(B3)* **then**
4 $\quad\quad Q$.push(boxes of K_i);
5 $\quad$ **else** Connect K_i's boundary vertices;
6 **return** Q;

Interpolating edges. The PV construction creates edges within a box B, which start and end from 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.

D.2 Correctness proof

▶ **Theorem 20.** *Algorithm 4 returns a regular isotopic ε-approximation of the n-ellipse $F^{-1}(0)$.*

Proof. The standard PV construction terminates for regular curves $S = F^{-1}(0)$. This implies that boxes of type (A3) can only survive in the neighborhood of foci. As time passes those neighborhoods become smaller and the neighborhoods of 2 different foci will become disjoint. That means that, eventually, properties (B1) - (B3) will be satisfied for each component and no box will be put back to queue Q in line 4 of Algorithm 5.

The property that the output is a regular isotopic approximation of the k-ellipse is inherited from the PV-construction of regular curves. In the following we show that it is also an ε-approximation of the k-ellipse.

Let $S = F^{-1}(0)$ and S^* its approximation derived by Algorithm 4. We prove that the distance from any point on S^* to S is at most ε. A green or orange box B contains an edge of S^* only if F admits different signs when evaluated at corners on B's boundary. In that case also the k-ellipse has to pass through B. The box radius of B is smaller than $\varepsilon/2$ and therefore any point on S^* in B has at most ε distance to S. Let p be a point on S^* in a gray box of component K. The component K has two ingoing edges and in particular two points on its boundary, which are on S, see Fig. 12b. Therefore the distance from p to S can be bounded by the diameter of K, which is smaller than ε. None of the red boxes contains a part of S^*.

Finally we prove that the distance from any point on S to S^* is at most ε. All the boxes, which might contain parts of S satisfy the C_1 predicate (green and orange) or are part of a small component of gray boxes. If a box satisfies the C_1 predicate but the function F has the same sign at all its corners, then the curve S might possibly enter the box but also has to leave the box on the same side of B [37] and any neighboring box on that side has different signs for F on its corners. Let p be a point on S in box B and let B_1 and B_2 be the next boxes which are reached by walking from p along S in both directions. Note that B_1 and B_2 might be the same box. If B is a gray box of component K, then the distance from p to S^* can be bounded by the distance between p and the edge of S^* in K. This distance is bounded by the diameter of K which is less than $\varepsilon/2$. If B is a green or orange box, then it satisfies the C_1 predicate and box B_1 and B_2 have different signs at their 4 corners. If B_1 or B_2 are green or orange then the approximation S^* passes through them and p is close enough to S^*. If both B_1 and B_2 are gray, then the edge of S^* through their components is close to p. Finally, B cannot be a red box by definition. ◀

D.3 Elliptic Contour Plotting

Contour Plots help readers to quickly infer useful information about some parameter, e.g. on a map, and are very popular in domains as Cartography, Meteorology and Social Sciences among others. In this section, we combine the developed algorithms in order to produce a topologically correct, ε-approximate and visually nice *n-elliptic contour plot*. In our context, a *contour line* is an n-ellipse, a *contour plot* is a set of n-ellipses with different radii and our goal is to compute m many n-ellipses inside a bounding box B_0. We consider a contour plot to be visually nice when the m contour lines are equally distributed in space.

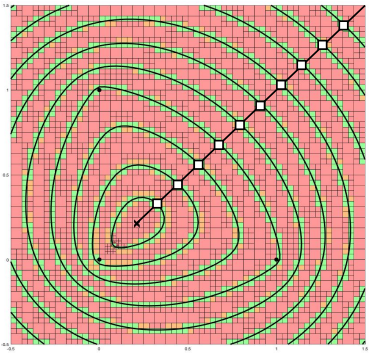
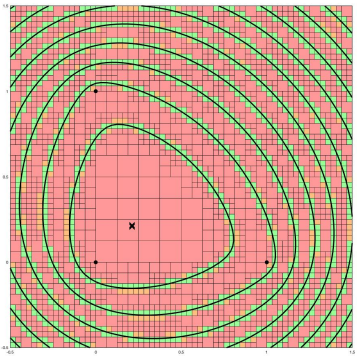
(a)  (b) 

Figure 13 A 3-elliptic contour plot with 10 contour lines. The foci are $(0,0),(1,0)$ and $(0,1)$. (a) Using radii of *equidistant points* as in Algorithm 6 $\{1.96, 2.05, 2.22, 2.46, 2.77, 3.13, 3.52, 3.94, 4.38, 4.83\}$. (b) Using *equidistant radii* $\{2.24, 2.54, 2.85, 3.15, 3.46, 3.76, 4.06, 4.37, 4.67, 4.98\}$.

Ellipses with different radii. Algorithm 4 can be adapted to plot several n-ellipses, corresponding to the same foci but with different radii, in the same box subdivision. This can be done by adding an additional condition to line 8. Within any box B, the exclusion predicate $C_0^{\mathrm{Ex}}(B)$ should be satisfied for all but one n-ellipse. With this additional condition we make sure that the n-ellipses are well separated in the subdivision diagram.

Contour plotting. We describe an algorithm for visually nice contour plots, see Fig. 13a for a sample output. We first approximate $\boldsymbol{x}^*$ and then compute the point $\boldsymbol{p}_{m+1}$, which maximizes φ in B_0. By the convexity of φ this point is a corner of B_0, so it suffices to evaluate φ in the four corners. Then, we split the segment $\overline{\boldsymbol{x}^*, \boldsymbol{p}_{m+1}}$ into $m+1$ congruent segments obtaining a sequence of points $\boldsymbol{x}^*, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_{m+1}$, and for each $\boldsymbol{p}_i$ we evaluate φ to obtain the radius $r_i = \varphi(\boldsymbol{p}_i)$ Finally, we apply the adapted version of Algorithm 4 which takes the set of all m radii and simultaneously computes all m ellipses. Refer to Algorithm 6.

Algorithm 6 Elliptic Contour Plotting

Input: Set A, constant $\varepsilon > 0$, box B_0, $m \geq 2$. **Output:** Family of curves $\mathcal{E}$.

1 $\boldsymbol{x}^* \leftarrow$ ALGORITHM-2(A, ϵ, B_0);
2 $\boldsymbol{p}_{m+1} \leftarrow$ Corner of B_0 maximizing φ;
3 $[\boldsymbol{x}^*, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_{m+1}] \leftarrow$ Sequence of equidistant points;
4 **return** $\mathcal{E} \leftarrow$*ALGORITHM-4*$(A, \{\varphi(\boldsymbol{p}_1), \ldots, \varphi(\boldsymbol{p}_m)\}, \varepsilon, B_0)$;

E Proof of Lemma 5

▶ **Lemma 5.** *The soft gradient exclusion predicate* $\square C^{\nabla}(B)$ *is convergent, i.e., for any monotone sequence of boxes* $(B_i)_{i\in\mathbb{N}}$ *that converges to a point* $\boldsymbol{p}$*, the point* $\boldsymbol{p}$ *is not the Fermat point if and only if* $\square C^{\nabla}(B_i) = \texttt{success}$ *for large enough* i*.*

Proof. First, let us assume that $\boldsymbol{p}$ is not a focus, i.e. $\|\nabla\varphi(\boldsymbol{p})\| \neq 0$. Then the boxes B_i do not contain a focus for big enough i. The box approximation $\square\nabla\varphi(B)$ is convergent and therefore $\square\nabla\varphi(B_i) \to \nabla\varphi(\boldsymbol{p})$. Finally, because $\|\nabla\varphi(\boldsymbol{p})\| \neq 0$ we know that $\exists i \in \mathbb{N}$ such that

$0 \notin \| \square\nabla\varphi(B_i)\|$. Hence $\square C^{\nabla}(B_i)$ succeeds. The other direction clearly holds true, indeed, if $\square C^{\nabla}(B_i)$ is true, then $\boldsymbol{p}$ is not the Fermat point $\boldsymbol{x}^*$, because $\|\nabla\varphi(\boldsymbol{x}^*)\| = 0$.

In the second part we consider the case where $\boldsymbol{p}$ is a focus $\boldsymbol{a} \in A$. If $\boldsymbol{a}$ is not the Fermat point, then $\left\|\nabla\varphi_{A\setminus\boldsymbol{a}}(\boldsymbol{a})\right\| > w(\boldsymbol{a})$ by [27]. In this case the norm of the box approximation of the gradient is computed by:

$$\square\,\|\nabla\varphi(B)\| = \left\|\,\square\nabla\varphi_{A\setminus\boldsymbol{a}}(B)\right\| + [-w(\boldsymbol{a}), w(\boldsymbol{a})]$$

In the previous part we already derived that $\square\nabla\varphi_{A\setminus\boldsymbol{a}}(B) \to \nabla\varphi_{A\setminus\boldsymbol{a}}(\boldsymbol{a})$ for $B \to \boldsymbol{a}$ and thus, we obtain

$$\square\,\|\nabla\varphi(B)\| \overset{B\to\boldsymbol{a}}{\longrightarrow} \underbrace{\left\|\nabla\varphi_{A\setminus\boldsymbol{a}}(\boldsymbol{a})\right\|}_{>w(\boldsymbol{a})} + [-w(\boldsymbol{a}), w(\boldsymbol{a})].$$

Therefore $\exists i \in \mathbb{N}$ such that $0 \notin \square\,\|\nabla\varphi(B_i)\|$ and hence $\square C^{\nabla}(B_i)$ succeeds. As above, the other direction is clearly true. ◀

F Termination of Algorithm 2

▶ **Definition 21.** *An box sequence B_n is called monotone if $B_{i+1} \subset B_i$. A monotone box sequence is convergent if $\lim_{i\to\infty} \omega(B_i) = 0$. A monotone convergent box sequence B_i converges quadratically if $\omega(B_{i+1}) = O(\omega(B_i)^2)$.*

Let $x_i \in \mathbb{R}^d$ be a quadratically convergent sequence of points, where $x_{i+1} = N(x_i)$ and N is the standard Newton operator for points. Then the sequence $B_{i+1} = B_i \cap N(x_i, B_i)$, where this time N is the Newton operator from Nickel [33], converges quadratically.

Let B_i be a box sequence, which converges quadratically to point p, and let $q_i \in B_i$ be the point with maximum distance to p. Then also the sequence $\|q_i - p\|$ converges quadratically, i.e. $\exists\lambda < 1$ such that $\|q_{i+1} - p\| \le \lambda\|q_i - p\|^2$. In particular, for big enough i we have $\|q_{i+1} - p\| \le \frac{1}{6}\|q_i - p\|$.

▶ **Definition 22.** *The* Newton zone *Z is a neighborhood of the Fermat point $\boldsymbol{x}^*$ with the property:*

$$\forall B \in Z: \quad \max_{p\in N(B)} d(\boldsymbol{x}^*, p) \le \frac{1}{6}\max_{p\in B} d(\boldsymbol{x}^*, p). \tag{6}$$

▶ **Lemma 23.** *If $\boldsymbol{x}^* \in B$ and B is contained in the Newton zone, then $N(2B) \subset 2B$.*

Proof. Because of the triangle inequality and the property that $\boldsymbol{x}^* \in B$, we have the following two inequalities:

$$\max_{p\in 2B} d(p, \boldsymbol{x}^*) \le r_{2B} + r_B = 3r_B \tag{7}$$

$$\min_{p\in\partial(2B)} d(p, \boldsymbol{x}^*) \ge \omega_{2B} - \omega_B = \frac{1}{\sqrt{2}}(r_{2B} - r_B) > \frac{1}{2}r_B \tag{8}$$

Box B is contained in the Newton zone and therefore we get:

$$\max_{p\in N(2B)} d(p, \boldsymbol{x}^*) \overset{(6)}{\le} \frac{1}{6}\max_{p\in 2B} d(p, \boldsymbol{x}^*) \overset{(7)(8)}{<} \min_{p\in\partial(2B)} d(p, \boldsymbol{x}^*).$$

This means that the distance from $\boldsymbol{x}^*$ to any point in $N(2B)$ is smaller than the minimum distance from $\boldsymbol{x}^*$ to a point on the boundary of $2B$ and hence the claim follows. ◀

▶ **Theorem 24.** *Algorithm 2 terminates and returns an ε-approximation of the Fermat point $\boldsymbol{x}^*$.*

Proof. The test $\square C^{\nabla}$ eventually removes all boxes from the queue Q which are not fully contained in the Newton zone. Let $B \in Q$ be the box, which contains the Fermat point $\boldsymbol{x}^*$. By Lemma 23 the test $\square C^{N}$ succeeds for box B. After splitting B, all of its 16 children remain in the Newton zone and one of them contains $\boldsymbol{x}^*$. Hence the algorithm becomes a repeated application of the Newton operator, which converges quadratically within the Newton zone. ◀

G Termination of Algorithm 3

▶ **Lemma 25.** *If B is contained in the Newton zone (see Definition 22) and $B \cap N(B) \neq \emptyset$, then $\boldsymbol{x}^* \in 3B$.*

Proof. We apply two times the triangle inequality to derive:

$$\begin{aligned} d(m_B, \boldsymbol{x}^*) &\leq \frac{\omega_B}{2} + d(B, \boldsymbol{x}^*) \\ &\leq \frac{\omega_B}{2} + \underbrace{d(B, N(B))}_{=0} + \sqrt{2}\omega_{N(B)} + d(N(B), \boldsymbol{x}^*) \end{aligned}$$

The term $d(B, N(B))$ is 0 because $B \cap N(B) \neq \emptyset$. We know that B is in the Newton zone and therefore $\omega_{N(B)} \leq \frac{1}{6}\omega_B$ and $d(N(B), \boldsymbol{x}^*) \leq \frac{1}{6}\max_{p \in B} d(\boldsymbol{x}^*, p)$.

$$\begin{aligned} d(m_B, \boldsymbol{x}^*) &\overset{(6)}{\leq} \frac{\omega_B}{2} + \frac{\sqrt{2}}{6}\omega_B + \frac{1}{6}\max_{p \in B} d(\boldsymbol{x}^*, p) \\ &\leq \frac{\omega_B}{2} + \frac{\sqrt{2}}{6}\omega_B + \frac{1}{6}\left(d(m_B, \boldsymbol{x}^*) + \frac{\sqrt{2}}{2}\omega_B\right) \end{aligned}$$

Therefore $d(m_B, \boldsymbol{x}^*) \leq \frac{3}{2}\omega_B$, which means that $\boldsymbol{x}^*$ is contained in $3B$. ◀

▶ **Theorem 9.** *Algorithm 3 terminates and returns an ε-approximate Fermat point. (Refer to Appendix G.)*

Proof. Let l_i (resp. p_i) be the sequence of box sizes (resp. box centers) generated by Algorithm 3. For i large enough, the box B with center p_i and width l_i is contained in the Newton zone, as the sequence p_i converges linearly to the Fermat point [35] [22]. When B is contained in the Newton zone and $\frac{B}{10} \cap N\left(\frac{B}{10}\right) \neq \emptyset$, we derive from Lemma 25 that $\boldsymbol{x}^* \in \frac{3B}{10} \subset \frac{B}{2}$ and further by Lemma 23 it follows that $N(B) \subset B$. This implies that the center of B is an ε approximation of $\boldsymbol{x}^*$ as the width of the boxes never exceeds ε during the algorithm.

◀

H Fermat point on a focus

H.1 Condition for the Fermat point being a focus

The next theorem was already shown in [48], but we quickly sketch a proof again.

▶ **Theorem 26.** *A focus $\boldsymbol{a}$ is the Fermat point if and only if $\left\|\nabla\varphi_{A\setminus\{\boldsymbol{a}\}}(\boldsymbol{a})\right\| \leq w(\boldsymbol{a})$.*

Proof. We need to show that $\boldsymbol{a}$ is the minimizer of the convex function φ if $\left\|\nabla\varphi_{A\setminus\{\boldsymbol{a}\}}(\boldsymbol{a})\right\| \leq w(\boldsymbol{a})$. Let $v \in \mathbb{R}^2$ be any unit vector. The directional derivative of φ in direction v at $\boldsymbol{a}$ is

$$\lim_{h\to 0}\frac{\varphi(\boldsymbol{a}+hv)-\varphi(\boldsymbol{a})}{h} = \lim_{h\to 0}\frac{\varphi_{A\setminus\{\boldsymbol{a}\}}(\boldsymbol{a}+hv)-\varphi_{A\setminus\{\boldsymbol{a}\}}(\boldsymbol{a})}{h} + \lim_{h\to 0}\frac{\varphi_{\{\boldsymbol{a}\}}(\boldsymbol{a}+hv)-\varphi_{\{\boldsymbol{a}\}}(\boldsymbol{a})}{h}$$
$$= \langle v, \nabla\varphi_{A\setminus\{\boldsymbol{a}\}}(\boldsymbol{a})\rangle + w(\boldsymbol{a}) \overset{\|v\|=1}{\geq} -\left\|\nabla\varphi_{A\setminus\{\boldsymbol{a}\}}(\boldsymbol{a})\right\| + w(\boldsymbol{a}) \geq 0$$

Recall that $\langle\cdot,\cdot\rangle$ denotes the scalar product. This implies that starting from $\boldsymbol{a}$ the function φ is non-decreasing in any direction. The minimum of the convex function φ therefore has to be $\boldsymbol{a}$. ◀

H.2 Time for testing for the Fermat point being a focus

For simplicity, we had assumed for algorithms 1 to 3 that the Fermat point is not a focus. Note that this assumption can be checked in advance by evaluating $\left\|\nabla\varphi_{A\setminus\{\boldsymbol{a}\}}(\boldsymbol{a})\right\| \leq w(\boldsymbol{a})$ for each focus $\boldsymbol{a}$, see Theorem 26, which would take $(n^2 d)$ time. We will explain two reasons, why that quadratic testing time in n can be avoided for both subdivision algorithms.

We added this assumption, because the Newton tests cannot succeed for a box B if B contains the Fermat point. This is because $\square\nabla^2\varphi(B)$ has the interval $[\|m_B-\boldsymbol{a}\|-r, \|m_B-\boldsymbol{a}\|+r]$ in its denominator, which contains 0 if $\boldsymbol{a}\in B$. Hence the box $N(B)$ covers the whole space if a focus is in B.

Instead of checking the assumption, one can run algorithms 1 and 2 anyway and rely only on the soft gradient exclusion predicate. There is a more elegant solution for this problem, described next. Instead of testing all foci in the beginning, if one of them is the Fermat point, this can be done during the subdivision process. We keep track of the number of foci, which are contained in non-discarded boxes. If that number falls below a constant, then we test these few constantly many remaining foci for being the Fermat point. That can now be done in $O(nd)$ time.

I More details on the experiments

Interval method vs Algorithm 1 We compared our Algorithm 1 with a naive approach, called *interval method*. It is based on the fact that if given two boxes B_1 and B_2, such that the intervals $\square\varphi(B_1)$ and $\square\varphi(B_2)$ are disjoint, then the box with bigger function values cannot contain the Fermat point. The interval method is a subdivision algorithm like Algorithm 1, where at any time we keep track of the smallest upper bound b of intervals $\square\varphi(B)$, for boxes B visited so far. The interval method replaces the soft gradient exclusion predicate in line 4 of Algorithm 1 by the other soft exclusion predicate $b < \square\varphi(B)$. We compared these two methods for different values of n and ε using the data sets UNIF-1. The results are summarized in the next two tables. In all tests the soft gradient exclusion predicate performed much better. Note, that for boxes B near the Fermat point the value $\square\varphi(B)$ is very similar. Hence, the interval method needs to do many splitting operations for small ε and work with very high internal precision. This explains, why that naive method did not terminate within 600 sec for $n = 100$ and $\varepsilon = 10^{-7}$.

$\varepsilon = 10^{-3}$, $n =$	10	100	1000	10000
Interval method	0.99	1.72	9.62	89.3
SUB	0.11	0.29	2.06	21.0

$n = 100$, $\varepsilon =$	10^{-1}	10^{-3}	10^{-5}	10^{-7}
Interval method	0.74	1.77	2.84	timeout
SUB	0.15	0.31	0.46	0.61

We remark that many more types of synthetic datasets were considered, as points in convex position, points which are vertices of regular n-gons, points on a grid, etc, see Fig. 14.

Most of the useful information about the behavior of the algorithms can be extracted by experimenting on UNIF-1, UNIF-2 and TSPLIB, so we chose to mainly experiment and analyze only these. As an example, the running times for Algorithm 2 with PCA for fixed $n = 100$ and $\varepsilon = 10^{-6}$ are given in following table.

$n = 100, \varepsilon = 10^{-6}$	clusters	convex position	n-gon	UNIF-1	UNIF-2
running times	0.37	0.26	0.33	0.34	0.33

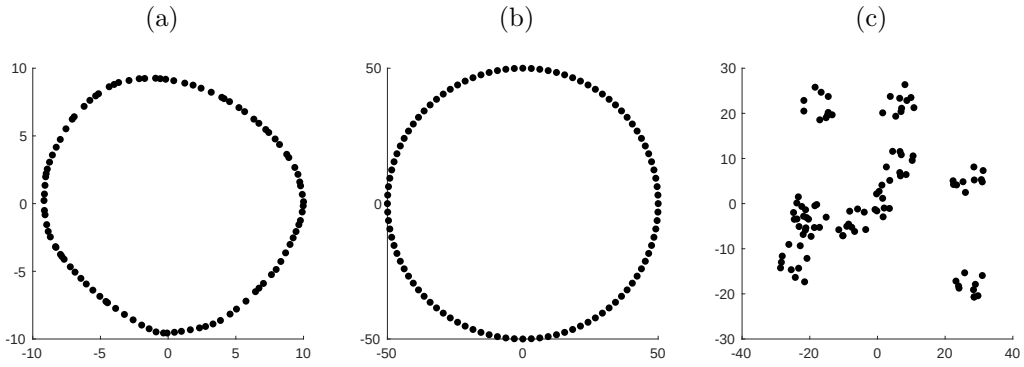


Figure 14 (a) 100 points in convex position (b) points of a regular 100-gon (c) 100 points split among 10 clusters

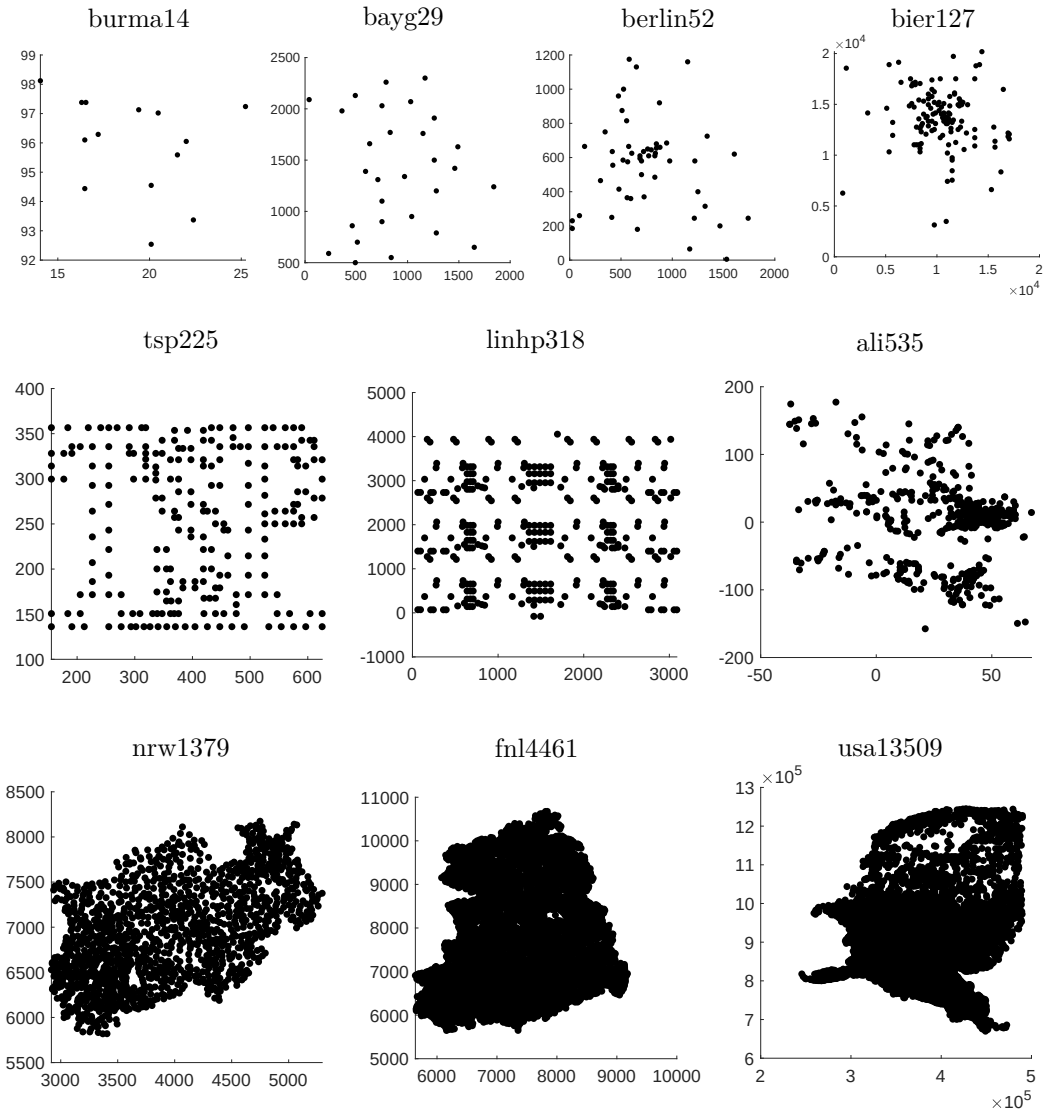


Figure 15 Foci sets of TSPLib used in our experiments. burma14: 14 cities in Burma, bayg29: 29 cities in Bavaria, berlin52: 52 locations in Berlin, bier127: 127 beer gardens in the Augsburg area (Germany), tsp225: writing of TSP with 225 points, linhp318: 318 cities, ali535: 535 airports around the globe, nrw1379: Nordrhein-Westfalen (Germany), fnl4461: the five Federal States of Germany (ex-GDR territory), usa13509: cities in the continental US with at least 500 population