Abstract. We consider approximation algorithms for the problem of computing an inscribed rectangle having largest area in a convex polygon on \( n \) vertices. If the order of the vertices of the polygon is given, we present a deterministic approximation algorithm that computes an inscribed rectangle of area at least \( 1 - \epsilon \) times the optimum in running time \( O(\frac{1}{\epsilon} \log \frac{1}{\epsilon} \log n) \). Furthermore, a randomized approximation algorithm is given that works with high probability and achieves a running time of \( O(\frac{1}{\epsilon} \log n) \).

1 Introduction

Much work has been devoted to computing inscribed objects of maximum area in a polygon in the past. Most contributions to this problem focus on objects that are again polygons, e. g., largest axis-aligned rectangles in convex or non-convex polygons [1, 5], largest squares and equilateral triangles in convex polygons [6], and largest empty rectangles on point sets [4].

In this paper, we consider the following problem: Given a convex polygon \( P \) with \( n \) vertices, approximate a rectangle of largest area that is inscribed in \( P \). To our knowledge, no algorithm that solves this problem is published so far. However, there is a straightforward way to compute a largest rectangle in a convex polygon in \( O(n^4) \) time.

A rectangle is said to be fat if the aspect ratio of the rectangle is bounded by a constant. Hall-Holt et al. [7] consider the problem of computing a fat rectangle with maximum area in a simple polygon and present a PTAS for this problem assuming that the largest inscribed rectangle is indeed fat. Our results show for convex polygons that fatness is not required for approximating a largest inscribed rectangle.

Furthermore, the dependence of the algorithm of Hall-Holt et al. is linear in the size of the input polygon. Since the algorithm is not described in detail

* Work by Schlipf, Schmidt and Tiwary was supported by the Deutsche Forschungsgemeinschaft within the research training group “Methods for Discrete Structures” (GRK 1408).
in [7], it is not clear if a sublinear running time can be obtained when the ordering of vertices of the input polygon is known. Under the assumption that this ordering is given, we obtain approximation algorithms whose running times are only logarithmically dependent on \( n \).

Such assumptions on the input are common when handling polygons. In fact, Alt et. al show under this assumption that the largest axis-aligned inscribed rectangle inside a convex polygon can be computed in logarithmic time [1]. When the ordering is not known, it can be easily computed using standard convex hull algorithms in \( O(n \log n) \) time; throughout the paper we will assume that the ordering is given.

The main result of this paper can be stated as follows.

**Theorem 1.** Let \( P \) be a convex polygon with \( n \) vertices. Suppose the vertices of the polygon are given in clockwise order. Then, an inscribed rectangle in \( P \) with area of at least \((1 - \epsilon)\) times the area of a largest inscribed rectangle can be computed

- in \( O(\frac{1}{\epsilon} \log \frac{1}{\epsilon} \log n) \) deterministic time.
- with high probability in \( O(\frac{1}{\epsilon} \log n) \) time.

Here, with high probability (w.h.p.) means that the randomized algorithm computes in deterministic time \( O(\frac{1}{\epsilon} \log n) \) an inscribed rectangle that has with probability \( \geq \frac{3}{4} \) an area of at least \((1 - \epsilon)\) times the area of the largest inscribed rectangle. Probability amplification, i.e., iterating the algorithm constantly many times can raise the probability of success to an arbitrary high constant \( c < 1 \) without increasing the asymptotic running time.

### 2 Preliminaries

We denote the area of a polygon \( P \) by \( |P| \). A line segment through two points \( a \) and \( b \) is denoted by \( \overline{ab} \) and its length by \( |\overline{ab}| \). For a given convex polygon \( P \), let \( R_{\text{opt}} \) be a largest inscribed rectangle. Note that in general the largest inscribed rectangle is not unique; we will use \( R_{\text{opt}} \) to denote any one of the largest inscribed rectangles.

We want to approximate the largest inscribed rectangle in a convex polygon. If we know the direction \( d_{\text{opt}} \) of one of the sides of \( R_{\text{opt}} \), we can compute the largest rectangle \( R_{\text{opt}} \) itself in \( O(\log n) \) time by applying the algorithm of Alt et al. [1]. The general idea of our algorithm is to approximate the direction of alignment of a largest inscribed rectangle and to prove that the area of the largest inscribed rectangle aligned along this direction also approximates \( |R_{\text{opt}}| \). For the computation, we construct a set of candidate directions and find the largest inscribed rectangle along each of these directions using the algorithm of Alt et al. [1]. The number of candidate directions will be \( O(\frac{1}{\epsilon} \log \frac{1}{\epsilon}) \) for the deterministic version of our algorithm, and \( O(\frac{1}{\epsilon}) \) for the randomized one.
3 Approximating the direction of $R_{opt}$

We want to find a direction "close enough" to the direction of any side of $R_{opt}$. This direction will be called an $\epsilon$-close direction, for a fixed $\epsilon > 0$. To define what $\epsilon$-close means, first suppose that we know $R_{opt}$ and denote the intersection of its diagonals as its center $s$. Let $\overline{ab}$ be one of the two shortest sides of $R_{opt}$ and let $d$ be the midpoint of the segment $\overline{ab}$, see Figure 1. Then $\angle(asb) \leq \frac{\pi}{2}$ and we can define the triangles $T_1$ and $T_2$ as the two triangles with vertices $s$ and $d$ and the third vertex being either $f_1 := d + \epsilon(b-d)$ or $f_2 := d - \epsilon(b-d)$.

Analogously, choosing the side of $R_{opt}$ opposite of $\overline{ab}$ gives the two triangles $T_3$ and $T_4$ having the same area. The area for each triangle $T_i$, $1 \leq i \leq 4$, is an $\epsilon/8$-fraction of $|R_{opt}|$.

**Lemma 1.** A largest inscribed rectangle $R_{apx}$ that is aligned to an $\epsilon$-close direction, contains an area of at least $(1 - 8\epsilon)|R_{apx}|$.

**Proof.** Let us consider the triangle $T = asb$ in $R_{opt}$ (see Figure 2) and an $\epsilon$-close direction $d_{apx}$ that intersects w.l.o.g. $\overline{df_1}$ (the case for $\overline{df_2}$ is symmetric). We first rotate $T$ around $s$ until $\overline{sd}$ is aligned with $d_{apx}$ and denote the angle between $d_{opt}$ and $d_{apx}$ by $\theta$. Then we scale the triangle such that $s$ and $\overline{sd}$ are preserved and the triangle is still isosceles, aligned with $\overline{sb}$ and fits completely into $T$. This rotation and scaling maps $a$ to a point $a'$, $b$ to a point $b' \in \overline{sb}$ and we get a new, smaller triangle $T' = a'sb'$ that lies completely inside $T$. Consider the midpoint $d'$ of the segment $\overline{a'b'}$. The segment $\overline{sd'}$ is aligned with the direction $d_{apx}$. Let $\alpha$ be the angle between $d_{apx}$ and $\overline{sb}$.

Instead of comparing $|R_{apx}|$ with $|R_{opt}|$ directly, we now compare the triangles $T$ and $T'$. If we can show that $|T'| \geq (1 - c\epsilon)|T|$ for some constant $c$, then...
the largest rectangle aligned to $d_{apx}$ has at least an area of $(1 - \epsilon)|R_{opt}|$. The reduction to triangles does not matter for the approximation, as $|R_{opt}| = 4|T|$ and $|R_{apx}| \geq 4|T'|$.

Recalling some elementary trigonometry we see that

$$
\epsilon = \frac{\tan \theta}{\tan \theta + \alpha}
$$

(\*)

$$
|T| = |sa|^2 \sin(\alpha + \theta) \cos(\alpha + \theta), \text{ and}
$$

$$
|T'| = |sa'|^2 \sin(\alpha) \cos(\alpha) = |sa|^2 \frac{\cos^2(\alpha + \theta) \sin(\alpha) \cos(\alpha)}{\cos^2(\alpha - \theta)}
$$

We want to show that

$$
|T'| \geq (1 - \epsilon)|T|
$$

$$
\iff \frac{|T'|}{|T|} \geq (1 - \epsilon)
$$

$$
\iff \frac{|sa|^2 \cos^2(\alpha + \theta) \sin(\alpha) \cos(\alpha)}{|sa|^2 \sin(\alpha + \theta) \cos(\alpha + \theta)} \geq (1 - \epsilon)
$$

$$
\iff \frac{\sin(\alpha) \cos(\alpha)}{\cos^2(\alpha - \theta) \tan(\alpha + \theta)} \geq (1 - \epsilon)
$$

for a constant $c$. This last equation can be proven by using (\*) and the following Lemma.

**Lemma 2.** The function $\frac{\sin(\alpha) \cos(\alpha)}{\cos^2(\alpha - \theta)} + c \tan(\theta) - \tan(\theta + \alpha)$ is positive for $0 \leq \alpha \leq \frac{\pi}{4}, 0 \leq \theta \leq \frac{\pi}{2}$ and a constant $c \geq 8$.

See the appendix for a proof. Lemma 2 shows that $|T'| \geq (1 - 8\epsilon)|T|$. Thus, $|R_{apx}| \geq (1 - 8\epsilon)|R_{opt}|$. \[\square\]
4 How to get an \( \epsilon \)-close direction

It only remains to show how to compute an \( \epsilon \)-close direction to \( d_{\text{opt}} \) efficiently. We get such an \( \epsilon \)-close direction as follows: Assume first that we know the center \( s \) of \( R_{\text{opt}} \). If we choose \( O(\frac{1}{\epsilon}) \) random points uniformly distributed inside \( P \), with high probability at least one of them lies in the triangle \( T_i \) (\( i = 1, 2, 3 \) or 4), which gives us immediately an \( \epsilon \)-close direction (see Figure 1). As we do not have the information about the location of \( s \), assume that any other point \( p \) inside \( R_{\text{opt}} \) is given. Then there is a translated copy \( T'_i \) of \( T_i \), where the translation maps \( s \) to \( p \). And this triangle lies inside \( R_{\text{opt}} \) and so also inside \( P \). Picking a point \( q' \) in this translated copy and taking the direction \( pq' \) has the same effect as picking a point \( q \) in \( T_i \) and taking the direction \( sq \). So we do not have to compute \( s \) explicitly, and we just have to find a point inside \( R_{\text{opt}} \).

Even though we do not know \( R_{\text{opt}} \), picking points from it essentially amounts to picking points from the input polygon because the area of the largest inscribed rectangle in a convex polygon is at least a constant factor of the area of the polygon. More formally,

**Lemma 3 ([10]).** Let \( P \) be a convex polygon and \( R_{\text{opt}} \) be a largest inscribed rectangle in \( P \), then \( |R_{\text{opt}}| \geq |P|/2 \).

4.1 Randomized algorithm

It follows from Lemma 3 that if we pick \( k \) points sampled uniformly at random from a convex polygon \( P \), the expected number of points inside \( R_{\text{opt}} \) is \( \frac{k}{2} \). All these points are distributed uniformly at random inside \( R_{\text{opt}} \). Moreover, if we pick \( O(\frac{1}{\epsilon}) \) points uniformly at random, the expected number of points inside the triangle \( T'_i \) is \( O(1) \). Thus, we have the following lemma.

**Lemma 4.** Let a convex polygon \( P \) and a source of random points in \( P \) be given. Then we can compute a \((1 - \epsilon)\) approximation \( R_{\text{apx}} \) for the largest inscribed rectangle in \( P \) with high probability in time \( O(\frac{1}{\epsilon^2} \log n) \).

We can achieve the same running time without random points in \( P \) being given. It is easy to see that with a preprocessing of \( O(n \log n) \) we can create a data structure for a (not necessarily convex) polygon \( P \) that returns a point distributed uniformly at random inside \( P \) in \( O(\log n) \) time per sample. This can be achieved by first computing a triangulation of the point set and then creating a balanced binary tree with the triangles as leaves, where the weight of any node is the sum of areas of all triangles contained in the subtree rooted at that node. Sampling a random point from \( P \) then amounts to traversing this tree from root to a leaf and following the left or the right child at any node with the probability proportional to their weights.

Since the ordering of the vertices of \( P \) is given and we want to avoid any preprocessing for \( P \), we will not sample points from \( P \) uniformly at random. Instead, we take a uniform distribution over a square and “fit” these points inside the polygon. Thus, the sampling from \( P \) will simulate the sampling of
random points from a square. Let \( v_t, v_b \) be the topmost and the bottommost vertices of \( P \), we can determine them in \( O(\log n) \) time. We pick a height \( h \) between the two vertices uniformly at random. We take the longest horizontal segment that fits inside \( P \) at this height, again we can compute this segment in \( O(\log n) \) time, and pick a point uniformly at random on this segment. This will be our sample point in \( P \). We can repeat this process as many times as desired to get a large set of sample points that are in \( P \). Each of these sample points can be generated in \( O(\log n) \) time assuming that the ordering of vertices of \( P \) is known in advance.

We need to show that such a sampling works for our algorithm. Recall that we need two points \( p \) and \( q \) from \( P \) such that \( p \) lies in a largest inscribed rectangle \( R_{opt} \) and \( q \) lies in a triangle of area \( \Omega(\epsilon) \) that is a translated copy of one of the \( T_i \)'s (see Figure 1). We will show that with our sampling method, the probability of a sample point to lie in any convex region \( Q \) of area \( \epsilon |P| \) is at least \( \frac{\epsilon}{2} \).

Let \( L_h \) be the length of the largest horizontal segment inside \( P \) at height \( h \), and \( l_h \) be the length of the largest horizontal segment inside \( Q \) at height \( h \). Also, assume that the bottommost and topmost points in \( P \) are at heights 0 and 1 respectively (see Figure 3). It holds \( \frac{|Q|}{|P|} = \frac{\int_0^1 l_h \, dh}{\int_0^1 L_h \, dh} \). The probability that a sample point using the above sampling method lies in \( Q \) is \( \int_0^1 \frac{l_h}{L_h} \, dh \). Since for any value of \( h \) we can find a quadrilateral that fits inside \( P \) and has area at least \( \frac{L_h}{2} \), we have that \( \frac{L_h}{2} \leq \int_0^1 L_h \, dh \). Therefore,

\[
\int_0^1 \frac{l_h}{L_h} \, dh \geq \frac{1}{2} \int_0^1 \frac{l_h}{L_h} \, dh
\]

Since each of these sample points can be generated in logarithmic time, the complexity of our algorithm is \( O(\frac{1}{\epsilon} \log n) \).

We summarize the steps in Algorithm 1.
Algorithm 1

1: Take $O(1)$ points in $P$ with the aforementioned distribution and store them in $U$.
2: Take $O(1/\epsilon)$ points in $P$ with the aforementioned distribution and store them in $V$.
3: $|\text{Rapx}| = 0$
4: for all $u \in U$ do
5:   for all $v \in V$ do
6:     Compute the largest inscribed rectangle $S$ that is aligned to $uv$.
7:     if $|S| \geq |\text{Rapx}|$ then
8:        $\text{Rapx} = S$
9: return $\text{Rapx}$

4.2 Deterministic algorithm

The results of this section are summarized in the following lemma

Lemma 5. Let a convex polygon $P$ and the cyclic order of vertices in $P$ be given. Then we can compute a $(1-\epsilon)$ approximation $\text{Rapx}$ for the largest inscribed rectangle in $P$ in $O(1/\epsilon \log 1/\epsilon \log n)$ time.

It remains to show how the algorithm for the deterministic case computes sample points in $P$. First, we compute an enclosing rectangle $R_e$, such that $P \subset R_e$ and $|R_e| \leq 2|P|$.

Fig. 4. A polygon $P$ and an enclosing rectangle $R$, $|P| \geq |Q|$.

Lemma 6. Let $P$ be a convex polygon with vertices of $P$ given in cyclic order. Then there is an algorithm that computes an enclosing rectangle $R$ such that $P \subset R$ and $|R| \leq 2|P|$ in $O(\log n)$ time.

Proof. An antipodal pair of points of the vertices of $P$ is computed in $O(\log n)$ time. Let $v_i$ and $v_j$ be the antipodal points and let $l = |v_i v_j|$ (see Figure 4). Since $v_i$ and $v_j$ form an antipodal pair, there are two parallel supporting lines
for the polygon one passing through \( v_i \) and the other through \( v_j \). Clearly, the distance between these two hyperplanes is at most \( l \).

Take the direction of these two supporting lines as \( y \)-axis and compute a vertex \( v_i \) of \( P \) with highest \( y \)-value and a vertex \( v_b \) of \( P \) with lowest \( y \)-value. Let the vertical distance between \( v_b \) and \( v_t \) be \( k \). We know that \(|P| \geq \frac{k^2}{2} \) and the rectangle \( R \) through vertices \( v_i, v_j, v_b \) and \( v_t \) has area \(|R| \leq kl \). Therefore, \(|P| \geq \frac{|R|}{2} \). \( \square \)

Creating a grid of constant size in an enclosing rectangle \( R \) of Lemma 6 allows us to ensure a constant number of grid points in \( P \). This is proven in Lemma 7 by using Pick’s theorem.

**Theorem 2 (Pick’s Theorem [9]).** Let an integer grid and a simple polygon \( P \) with all vertices lying on the grid points be given. Let \( i \) be the number of grid points contained in \( P \) and \( b \) be the number of grid points on the boundary of \( P \). Then \(|P| = \frac{b^2}{2} + i - 1 \).

**Lemma 7.** For a convex set \( S \) and every enclosing rectangle \( R \) of \( S \) with \(|R| \leq c|S| \) for a constant \( c \), \( S \) contains \( \frac{1}{2c} k^2 \) grid points of an \( k \times k \) grid on \( R \).

**Proof.** Let \( G \) be a \( k \times k \) grid on \( R \) and let \(|R| = 1 \). We shrink \( S \) to a maximum area polygon \( S' \subseteq S \) having all vertices on grid points of \( G \). Because of convexity, \(|S| - |S'| \) is at most the area of \( 4k \) grid cells.

\[
|S'| \geq |S| - 4k \frac{1}{k^2} \geq \frac{1}{c} - \frac{4}{k} \\
|S'| = \left( \frac{b}{2} + i - 1 \right) \frac{1}{k^2} \geq \frac{1}{c} - \frac{4}{k} \quad \text{(by Pick’s theorem)} \\
b + i \geq \frac{1}{c} k^2 - 4k + 1
\]

Thus, at least \( \frac{1}{2c} k^2 \) grid points lie in \( S \), for a big enough constant \( k \). \( \square \)

It follows from Lemma 7 that choosing a grid with constant size on the rectangle \( R \) implies that \( R_{opt} \) in \( P \) contains a grid point. We additionally have shown that every \( \epsilon \)-fraction of \( P \), in particular every triangle \( T'_i \), contains many grid points for a big enough grid on \( R \).

The idea is to take two grids \( G_1 \) and \( G_2 \) on \( R \) of size \( O(1) \times O(1) \) and \( O(\frac{1}{\epsilon}) \times O(\frac{1}{\epsilon}) \), respectively, iterate through all pairs of grid points in \( G_1 \times G_2 \) and get at least one pair \((u,v)\) with \( u \in R_{opt} \) and \( v \in T'_i \) using Lemma 7. Hence, \( \overline{uv} \) has an \( \epsilon \)-close direction. We summarize the steps in Algorithm 2.

**Algorithm 2** is deterministic with running time \( O(\frac{1}{\epsilon^2} \log n) \). We can further reduce the running time to \( O(\frac{1}{\epsilon} \log \frac{1}{\epsilon} \log n) \) by using the tools from the theory of \( \epsilon \)-nets. Here we just give an outline of how these tools can be used, and we refer the reader to [8] for more details.

A subset \( S' \) of a given set \( S \) of \( N \) points is called an \( \epsilon \)-net for \( S \) with respect to a set of objects, if any object containing at least \( \frac{\epsilon}{2} N \) points of \( S \) contains
Algorithm 2

1: Compute an enclosing rectangle $R_e$ with area $|R_e| \leq 2|P|$
2: Compute a $O(1) \times O(1)$ grid on $R_e$. Let $G_1$ be the set of grid points.
3: Compute a $O(1/\epsilon) \times O(1/\epsilon)$ grid on $R_e$. Let $G_2$ be the set of grid points.
4: $|R_{\text{apx}}| = 0$
5: for all $u \in G_1$ do
6: for all $v \in G_2$ do
7: Compute the largest inscribed rectangle $S$ that is aligned to $uv$
8: if $|S| \geq |R_{\text{apx}}|$ then
9: $R_{\text{apx}} = S$
return $R_{\text{apx}}$

a point of $S'$. For objects with VC-dimension $d$, a subset $S'$ of size $O(\frac{1}{\epsilon} \log \frac{1}{\epsilon})$ always exists and can be computed in deterministic time $O(N^{2d})$. Rectangles and triangles have a constant VC-dimension, and we consider the set $S$ of grid points of a $\frac{1}{\epsilon} \times \frac{1}{\epsilon}$ grid, so $N = \frac{1}{\epsilon^2}$. Thus, we can compute an $\epsilon$-net for $S$ with size $O(\frac{1}{\epsilon} \log \frac{1}{\epsilon})$ in polynomial time to $\frac{1}{\epsilon}$.

5 Largest inscribed rectangles in simple polygons

The same ideas can be used to approximate the largest inscribed rectangle in a simple polygon with or without holes. It is easy to see that the largest inscribed rectangle $R_{\text{opt}}$ in a simple polygon (with or without holes) on $n$ vertices has an area of at least $\frac{1}{4n-2}$ times the area of the polygon. Moreover, a largest axis-aligned rectangle in a simple polygon can be computed in $O(n \log n)$ time [2] and in a simple polygon with holes in $O(n \log^2 n)$ [5]. Since $|R_{\text{opt}}|$ is an $\Omega(\frac{1}{\epsilon})$-fraction of $P$ and the area of each of the four triangles inside $R_{\text{opt}}$ is an $\Omega(\frac{1}{\epsilon})$-fraction of $P$, we get the following running times for computing an inscribed rectangle of area at least $(1-\epsilon)R_{\text{opt}}$.

- For simple polygons: W.h.p. in $O(\frac{1}{\epsilon} n^3 \log n)$ time
- For polygons with holes: W.h.p. in $O(\frac{1}{\epsilon} n^3 \log^2 n)$ time

In comparison with the algorithm of Hall-Holt et al. [7], which deals only with fat rectangles, our algorithm can handle general rectangles at the expense of a slower running time.

References


A Proofs of Proposition 2

Lemma 2. The function \( \frac{\sin(\alpha) \cos(\alpha)}{\cos(\alpha - \theta)} + c \tan(\theta) - \tan(\theta + \alpha) \) is positive for \( 0 \leq \alpha \leq \frac{\pi}{4}, 0 \leq \theta \leq \frac{\pi}{8} \) and a constant \( c \geq 8 \).

To prove this proposition, we need the following two propositions.

Proposition 1. \( \frac{1}{4} \tan(x) \leq \tan \left( \frac{x}{3} \right) \) for \( 0 \leq x \leq \frac{\pi}{4} \).

Proof. Consider the function \( f(x) = \frac{1}{4} \tan(x) - \tan \left( \frac{x}{3} \right) \). We have to show that \( f(x) \leq 0 \) for \( 0 \leq x \leq \frac{\pi}{4} \). The first and second derivatives of \( f(x) \) with respect to \( x \) are:

\[
f'(x) = \frac{1}{4} \sec^2(x) - \frac{1}{3} \sec^2 \left( \frac{x}{3} \right),
\]

\[
f''(x) = \frac{1}{2} \sec^2(x) \tan(x) - \frac{2}{9} \sec^2 \left( \frac{x}{3} \right) \tan \left( \frac{x}{3} \right)
\]

Since \( \tan(x) \geq \tan \left( \frac{x}{3} \right) \), we have

\[
f''(x) \geq 2 \tan \left( \frac{x}{3} \right) (\frac{1}{4} \sec^2(x) - \frac{1}{9} \sec^2 \left( \frac{x}{3} \right)).
\]

Let \( x' \) be the root of \( f'(x) = 0 \). That is, let \( x' \in [0, \frac{\pi}{4}] \) be such that

\[
\frac{1}{4} \sec^3(x') - \frac{1}{3} \sec^2 \left( \frac{x'}{3} \right) = 0
\]

Since \( \tan(x) \geq 0 \) in the domain \( 0 \leq x \leq \frac{\pi}{4} \), we have

\[
f''(x') \geq 2 \tan \left( \frac{x'}{3} \right) (\frac{2}{9} \sec^2 \left( \frac{x'}{3} \right)) \geq 0
\]

Therefore in the range \( [0, \frac{\pi}{4}] \), \( f(x) \) attains a minima whenever \( f'(x) = 0 \) and the maxima are attained only at the boundary. Since \( f(0) = 0 \) and \( f \left( \frac{\pi}{4} \right) < 0 \), \( f(x) \leq 0 \) for \( 0 \leq x \leq \frac{\pi}{4} \).

Proposition 2. If we choose \( \epsilon \leq \frac{1}{4} \), then \( \theta \leq \frac{\alpha}{2} \).

Proof.

\[
\frac{dg}{d\theta} = \frac{\tan(\theta)}{\tan(\alpha + \theta)} = \epsilon \leq \frac{1}{4}
\]

\[
\tan(\theta) \leq \frac{1}{4} \tan(\alpha + \theta) \leq \tan \left( \frac{\theta + \alpha}{3} \right)
\]

\[
\theta \leq \frac{\theta + \alpha}{3}
\]

\[
\theta \leq \frac{\alpha}{2}
\]

(*) using Proposition 1.

\( \square \)
Proof of proposition 2.

\[
\frac{\sin(\alpha) \cos(\alpha)}{\cos^2(\alpha - \theta)} + c \tan(\theta) - \tan(\theta + \alpha) = \tan(\alpha) \frac{\cos^2(\alpha) + c \tan(\theta)}{1 - \tan(\theta) \tan(\alpha)} - \frac{\tan(\alpha)}{1 - \tan(\theta) \tan(\alpha)}
\]

Since \(\frac{1}{1 - \tan(\theta) \tan(\alpha)} \leq \frac{1}{1 - \tan(\frac{\pi}{4})}\), it suffices to show that

\[
\tan(\alpha) \frac{\cos^2(\alpha)}{\cos^2(\alpha - \theta)} + c \tan(\theta) - \frac{\tan(\alpha)}{1 - \tan(\theta) \tan(\alpha)} \geq 0
\]

for some constant \(c' = c - 1.71 > 0\).

\[
\tan(\alpha) \frac{\cos^2(\alpha)}{\cos^2(\alpha - \theta)} + c' \tan(\theta) - \frac{\tan(\alpha)}{1 - \tan(\theta) \tan(\alpha)} = \tan(\alpha) \left( \frac{1}{1 - \tan(\theta) \tan(\alpha)} \right) \left( \frac{1 + \tan(\alpha) - \tan(\theta)}{1 + \tan(\theta) \tan(\alpha)} \right) + c' \frac{\tan(\theta)}{\tan(\alpha)} - \frac{1}{1 - \tan(\theta) \tan(\alpha)}
\]

Replacing \(\tan(\theta)\) by \(x\) and \(\tan(\alpha)\) by \(y\), we want to show that

\[
y \left( \frac{1 + (y - x)^2}{1 + y^2} + c' \frac{x}{y} - \frac{1}{1 - xy} \right) > 0
\]

\[
y \left( \frac{1 + (y - x)^2}{1 + y^2} + c' \frac{x}{y} - \frac{1}{1 - xy} \right) = \frac{y(1 - xy)(1 + x^2) + c' x (1 + xy)^2(1 - xy) - y(1 + xy)^2}{(1 + xy)^2(1 - xy)}
\]

Then, it suffices to show that

\[
y (1 - xy)(1 + x^2) + c' x (1 + xy)^2(1 - xy) - y(1 + xy)^2 > 0
\]

This is equivalent to

\[
(c' + 1) xy - 3y^2 - x^2 y^2 + c' + c' x^2 y^2 - c' x^3 y^3 - 2 c' x^2 y^2 - x y^3 > 0
\]

Since \(\tan(\alpha) \leq 1\) it suffices to show that \((-3 - x^2 + c' - c' x^3 - 2 c' x^2 - x) > 0\). That is \((c' - 3 - x - (2c' + 1) x^2 - c' x^3) > 0\). Since \(x = \tan(\theta) < \tan(\frac{\pi}{4})\) it suffices to pick \(c'\) such that \(c' - 3 - \tan(\frac{\pi}{4}) - (2c' + 1) \tan^2(\frac{\pi}{4}) - c' \tan^3(\frac{\pi}{4}) > 0\). So \(c' > \frac{3 + \tan(\frac{\pi}{4}) + \tan^2(\frac{\pi}{4})}{1 - \tan(\frac{\pi}{4}) \tan(\frac{\pi}{4})} \approx 6.12\). And so the function \(\frac{\sin(\alpha) \cos(\alpha)}{\cos^2(\alpha - \theta)} + c \tan(\theta) - \tan(\theta + \alpha)\) is positive for \(0 \leq \alpha \leq \frac{\pi}{4}, 0 \leq \theta \leq \frac{\pi}{8}\) and \(c \geq 8\). □