Notice all $\alpha_i < 180^\circ$
Lemma 1

\[ \alpha(j, j+k) = \bigcup_{i=j}^{j+k-1} \alpha_i \]

\(-l_2\) falls in some wedge \(\alpha_i\) within \(\alpha(j, j+k)\).

Certificate of non-monotonicity

Similarly, \(-l_1\) passes all local tests \(\alpha(j, j+k) < 180^\circ\) or no hope.
Given a monotone polygon, there is a gap between $\alpha(i, j-1)$ & $\alpha(j, i-1)$.
Two wedges are not in this diagram

Actually they must cover the gaps (closed path)

but the gaps are covered exactly once

The opposite also holds.

On the full polar diagram if we find opposite angles covered exactly once then the polygon is monotone.
Maintain angular ranges with 0, 1, or >2 wedge overlaps (3 types)

Build incrementally

Constructed range type 1
(type 0 not shown)
Extend range type 1
Don't need interior of range
No need for interior
Extension covers previous type 1 & creates a new one.
Triple-covering some parts but we don't care
Triple-covering some parts but we don't care
We get $O(n)$ time because we delete interior polar diagram borders and we stop incrementing at 2.
DECOMPOSITION OF POLYGON INTO $x$-MONOTONE PIECES

(partition)

(why?  ➔ many reasons: e.g. triangulation)

We already know how to triangulate monotone pieces
Clearly no "piece" can contain \( \text{cusp} \) or \( \text{cusp} \) \( \cup \) \( \text{cusp} \)...

So one method could be to identify these vertices and remove them by adding diagonals.

\( \Rightarrow \) time: \( O(n) \) per cusp \( \ldots \) assumes a diagonal will "resolve" a cusp and not create a new one.

\( \text{true} \)

So \( O(n^2) \)

Might as well triangulate by removing ears.

Recall, can always anchor a diagonal on a reflex vertex. Still reflex but not a cusp.
Monotone pieces in $O(n \log n)$ time

→ Why? → Each monotone piece can be triangulated in linear time

→ Heuristic to partition into larger pieces

Our plan is to remove cusps
Idea:

this should not be looking to join to

this
Now we have monotone pieces, but using Steiner vertices.

Can we find a resolving diagonal for every cusp?
Every "piece" has a left & right vertical wall w/ a cusp vertex on it.

except extreme

if $\approx$ then both get resolved.
If \( 1 > 2 \), then \( 1 \) can be resolved. Again the proof is constructive.
Can also handle extreme

All cases are quite similar
Assume we have constructed the verticals.

Find diagonals in \( O(n) \) time total \( \rightarrow \) sweep in disjoint regions

So the expensive part is getting those verticals.
PARTITION INTO TRAPEZOIDS with sweep line

Assume no 2 vertices w/ same x-coord: easy to deal with
Sweep direction: \( \overrightarrow{tx} \)

Stop at every vertex

\( \downarrow \) means we must sort by \( x \)

(polygon gives us no info)

At all times we maintain an ordered list of intersections with polygon
While you don’t sweep through any vertices, the only changes come from slopes.

Also, no verticals are added.
When you hit the next vertex, \( v \), look at edges incident to \( v \).

- If they are \( x \)-monotone, just update with new slope.

Number of segments on sweep line remains same.

Otherwise

- Merge
- OR
- Split
Do merge or split with binary search

- Maintain a balanced search tree
  - Keys: edges
  - Decision: vertex above/below edge?

- Merge
  - Find gap containing v.
  - Delete leaf & update other

- Split
  - Find segment containing v.
  - Update leaf & create new.

should have been more specific here. "containing" isn't the right word.
Balanced tree: \( O(bgn) \) time

Of course when we hit a vertex, we create a vertical.

This is why the slopes are useful: but really just maintain the edge.

compute from previous vertex & slope

co ordinates known
Result: $O(\log n)$ per vertex swept

$O(n \log n)$ to get trapezoidal decomposition

$O(n)$ total to triangulate after this.

Notice that it is easier to get rid of cusps if we add verticals at all vertices. [every trapezoid is empty and has a vertex on both verticals]