Finding and fixing geometry problems

4/5/2023

KKL

What are we looking for?

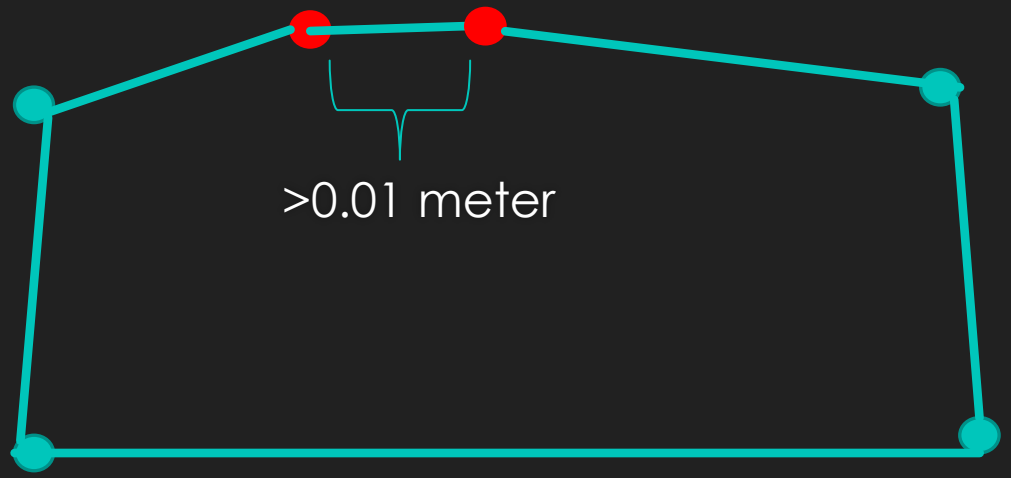
Geometrical problems in polygons, and spatially cadaster parcels in kkl layers

What are the problems we looking for?

- Vertices proximity
- Checking self-intersected and short segments geometry's
- Curves – non geometrical
- Overlapping parcels
- Holes
- Pseudo nodes

How do we find the problems?

- Vertices proximity - Exploring the arrays of each polygons, find vertices that are in proximity of (0.01) meters to each other



How do we find the problems?

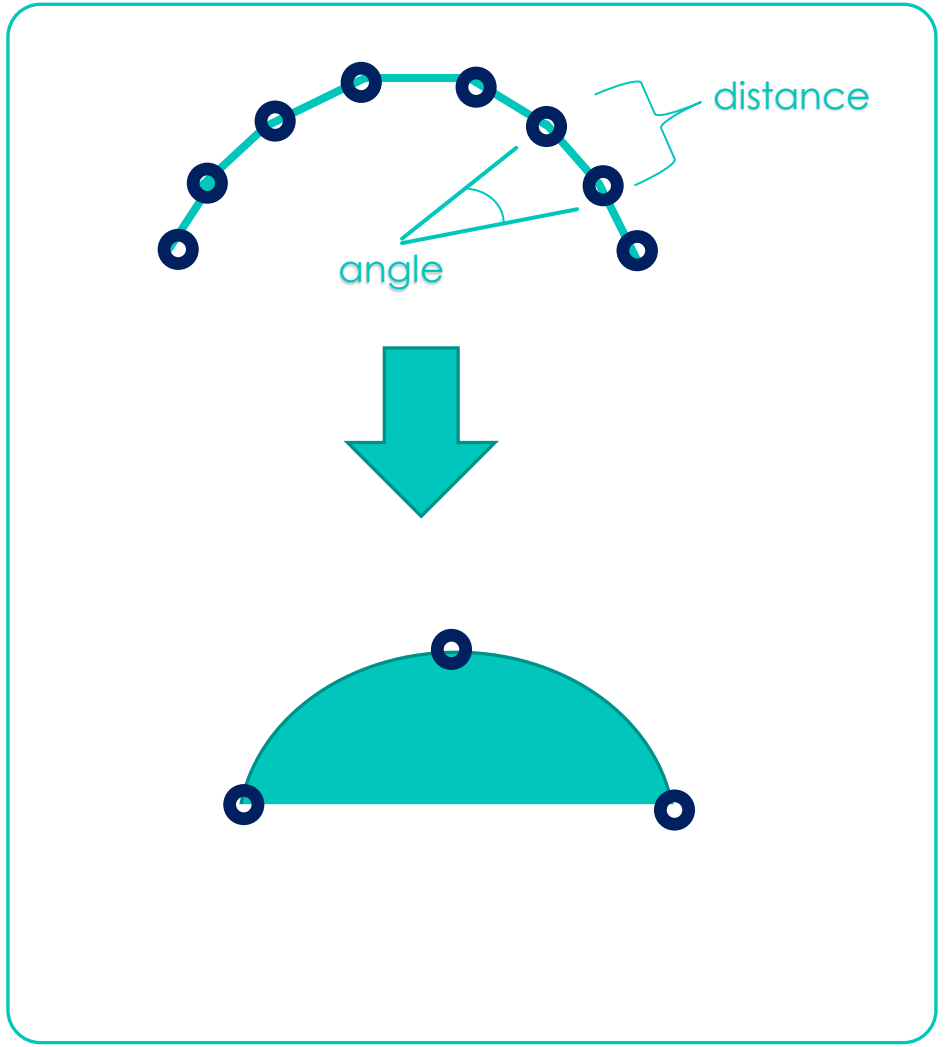
- Checking self-intersected and short segments geometry's in the array

1) [[X1,Y1], [X3,X3], [X2,Y2]] [[X1,Y1], [X2,X2], [X1,Y1]]

2) Short segments

How do we find the problems?

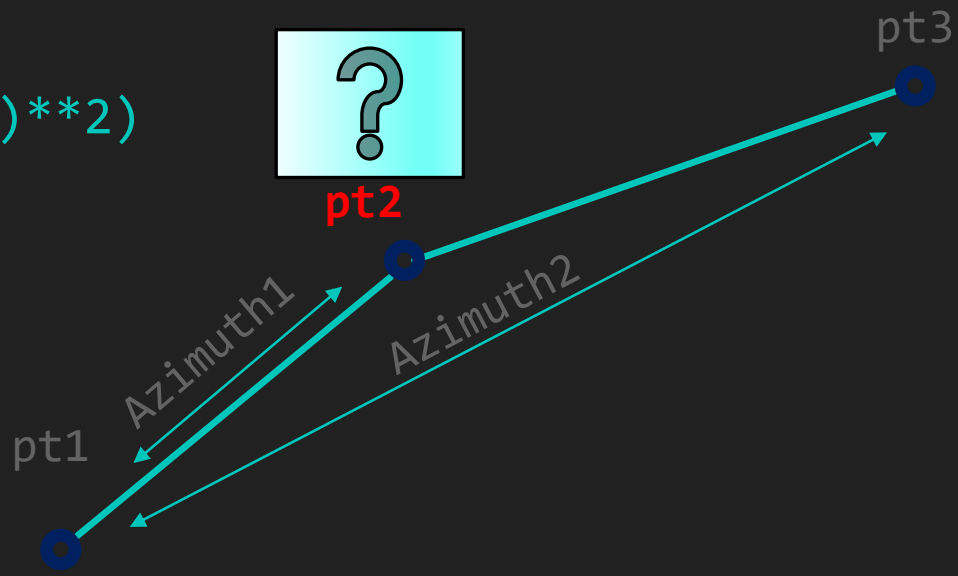
- Curves – finding curves problems by distance between vertices and angle



What can we do? Pseudo nodes

Pseudo nodes – delete vertices with no geometrical importance

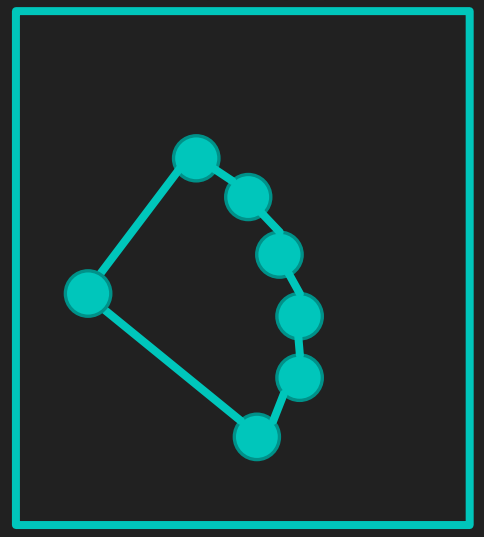
```
Azimuth1 = math.degrees(math.atan2((pt1.X-pt2.X),(pt1.Y-pt2.Y)))
Azimuth2 = math.degrees(math.atan2((pt1.X-pt3.X),(pt1.Y-pt3.Y)))
dAz = abs(Azimuth1- Azimuth2)
distance = math.sqrt((pt3.X-pt1.X)**2+(pt3.Y-pt1.Y)**2)
tmpTh = math.sin(dAz * (math.pi) / 180) * distance
if dAz <= 1: Delete pt2
```



What can we do? Curves

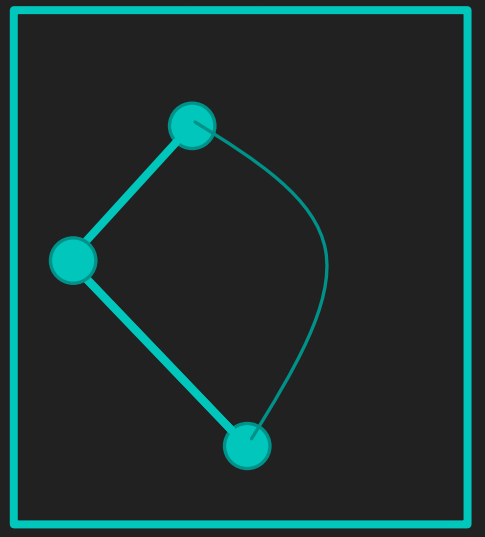
- Replacing array of vertices with geometrical entities
- Checking after each replacement that we don't change the area and shape of the polygon

A - polygon



[[x1,y1],[x2,y2],[x3,y3],[x4,y4]…]

B - polygon

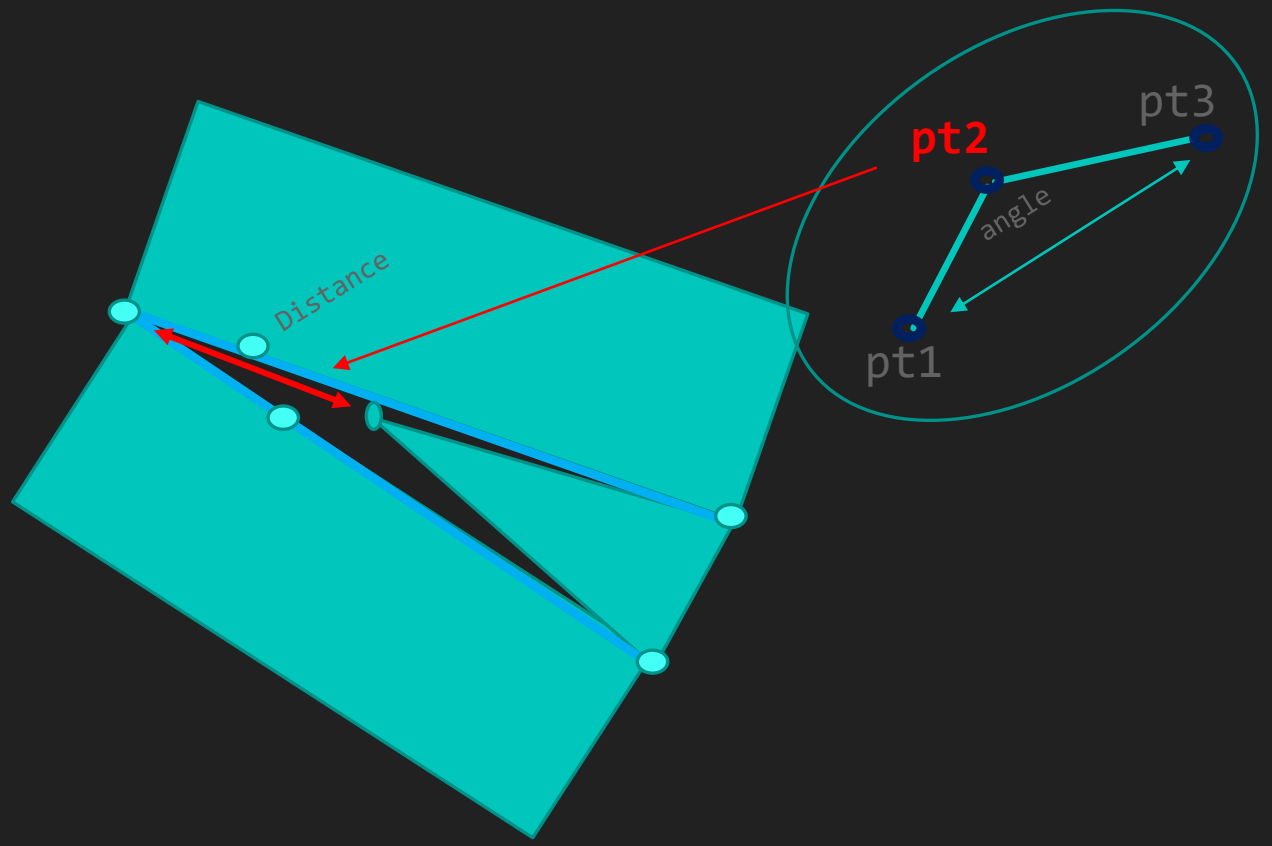


[[x1,y1],{'c':[x,y],[x,y]}]

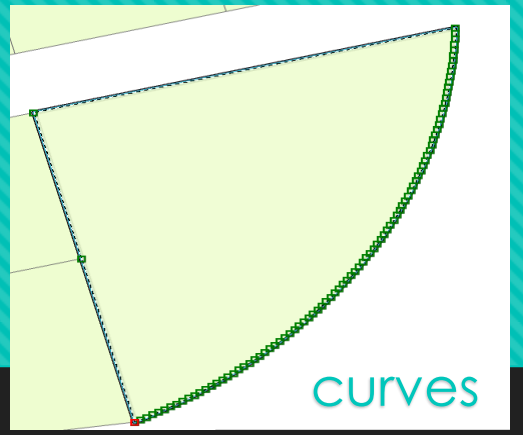
Finish Middle

What can we do? Holes

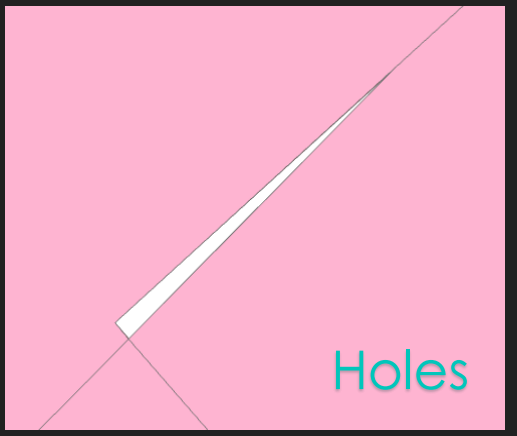
- Find the distance and if needed the angle between adjacent polygons, then creating a snapping tool that understand the weight of each vertex's movement.

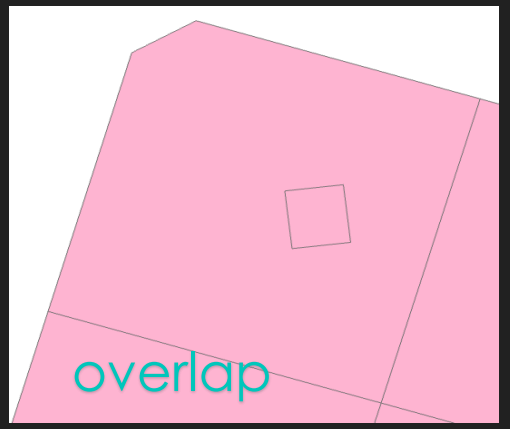


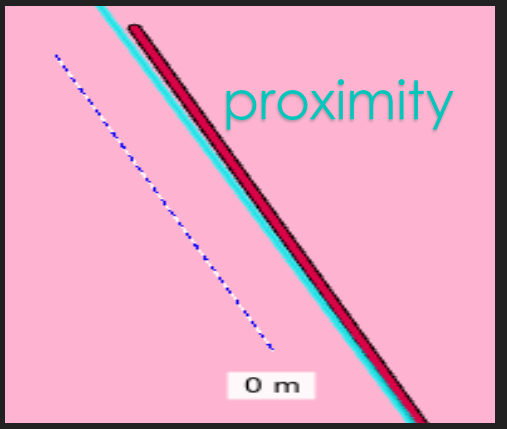
Results



- Curves - 73% of kkl parcels vertices are not needed
- Vertices proximity - +600 vertices problems
- Self intersect - +120
- Overlaps parcels - 24 overlaps
- Short segment - 4 parcels







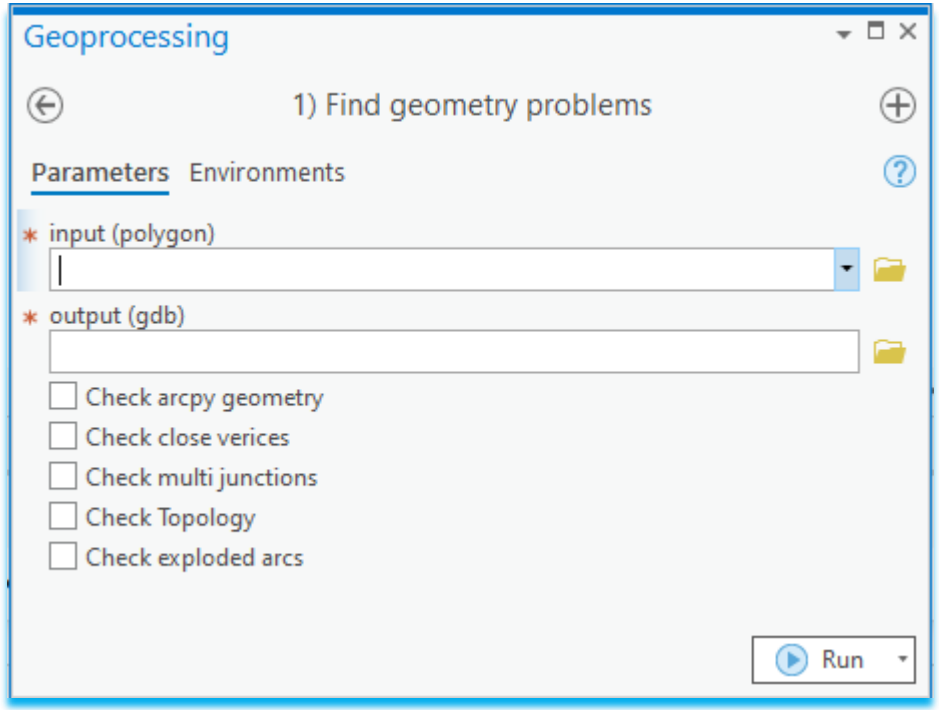
What can we do?

Automatic Fixing of geometry problems:

- Fixing vertices proximity – Finding the unnecessary vertices and delete from array
- Topology (holes , overlap) – creating filling rules for the gap\delete overlaps
- self-intersected – run over the array and find problems in vertices organization
- Curves – replace vertices with real curve geometry
- Pseudo nodes stand – Checking angle and "junction" between parcels, delete if the criteria

Geoprocessing tool

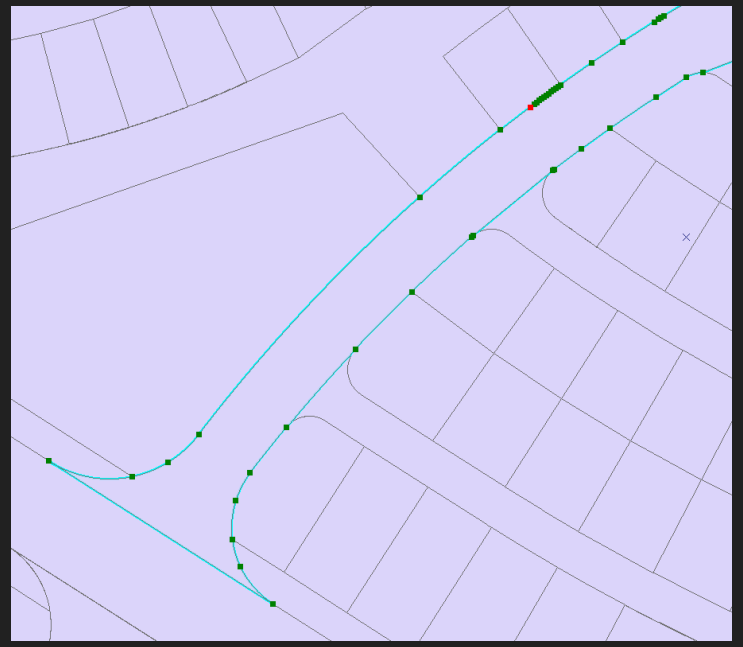
- Input – fc
- Input – gdb
- Input – check arcpy geometry
- Input – Check close vertices
- Input – Check multi junctions
- Input – Check Topology
- Input – Check exploded arcs



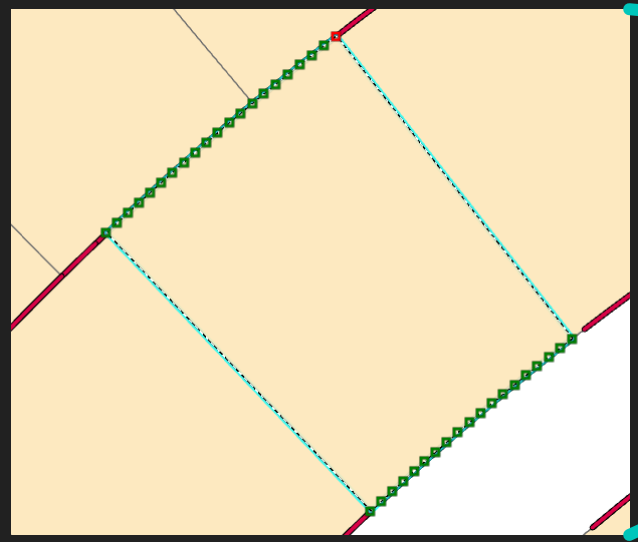
Results – vertices deleted

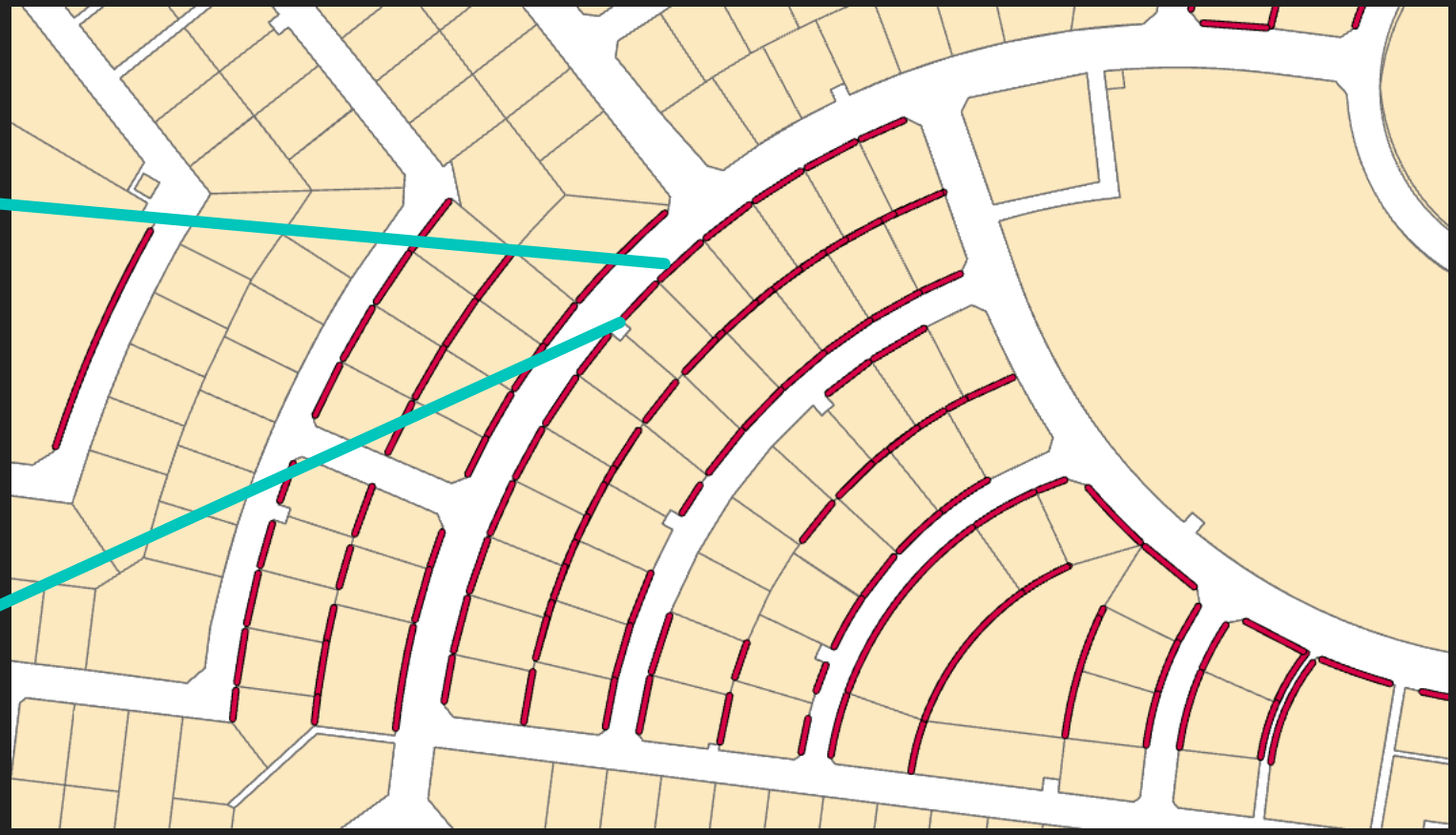
Before

After



Example – curves





- Thank for watching