Project 2

Moses Horwitz, Louis Feinberg, Jerome Howitz

The Idea

This method generates two *coterminal rays* by setting y as a function of x using two pieces:

$$y = \begin{cases} a(x-u)+v & \text{if } x < u \\ b(x-u)+v & \text{if } x \geq u \end{cases}$$

This will not intersect the given points. The objective is to minimize the sum of the vertical squared distances between the points at the line above or below it. This property is dependent on the Cartesian axes, and is not consistent under rotation. It takes the simplicity of a best-fit line and gives it more flexibility to accommodate a "bend" in the data.

The Computation

The collection of data is split by a vertical line between them, bisecting halfway between the extreme ends and dividing the data points into two subsets. Once this is done, a best-fit line is made separately for the each subset. We observe where they intersect, and set the x-value to u.

If this intersection happens to fall between the two subsets, we keep it and use the resulting rays. If it does not, we split along u to make two new subsets.

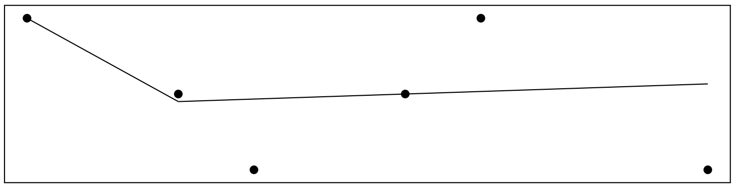
If we have fewer than two points in the left subset, we just set $u = x_2$, make a best-fit line for $\{x_2, ..., x_n\}$ on the right, and set the left segment to pass through (u, v) and (x_1, y_1). We do a similar act if there are fewer than two points on the right.

We repeat the process, then use whatever results transpire after the third iteration.

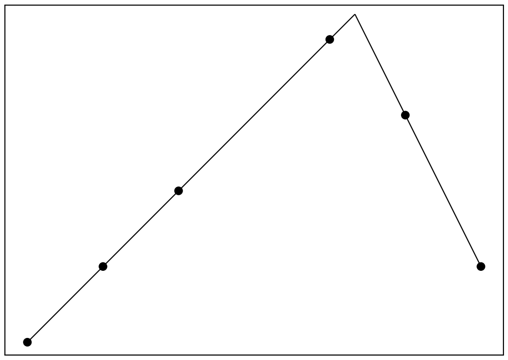
See attached for the code. It is written in Python, and uses the given `best_fit_line` function.

The Results

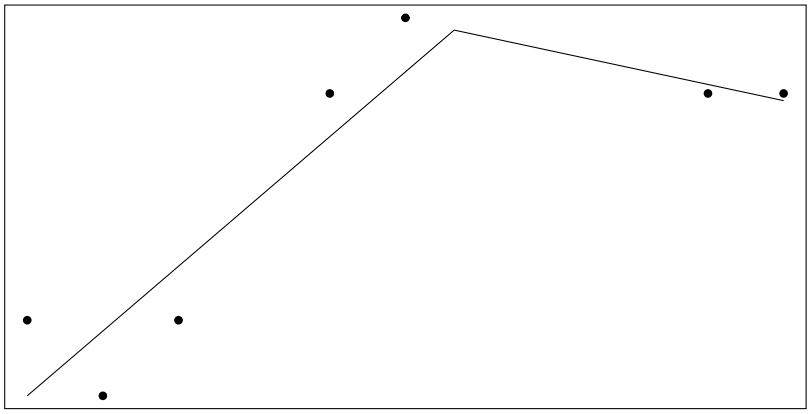
Graph 1:



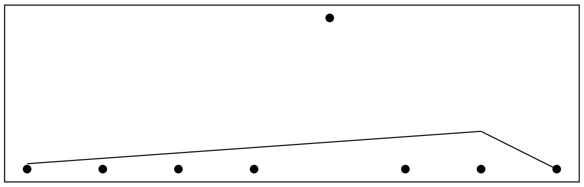
Graph 2:



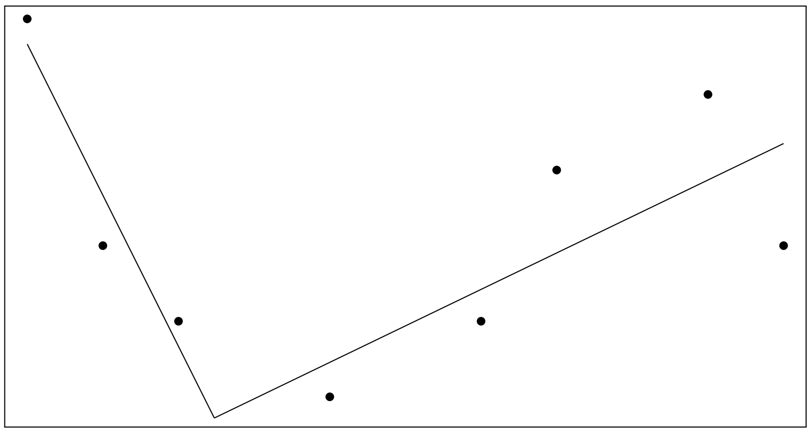
Graph 3:



Graph 4:



Graph 5:



Possible Issues

We have not proven that this will assuredly find the *best corner location*, in terms of least squared distances.

We do not have a full understanding of how this iteration works; it is possible that it may not settle in certain cases.

This method breaks down if it is ever given two parallel lines. If the lines are near-parallel, the corner will be thrown to one side and may give a less than optimal result.

Possible Extensions

This method might be extended to other best-fit functions, though finding intersections might be more difficult.

One possible idea is to allow for more than one corner, placing three or more coterminal rays/segments through the data.