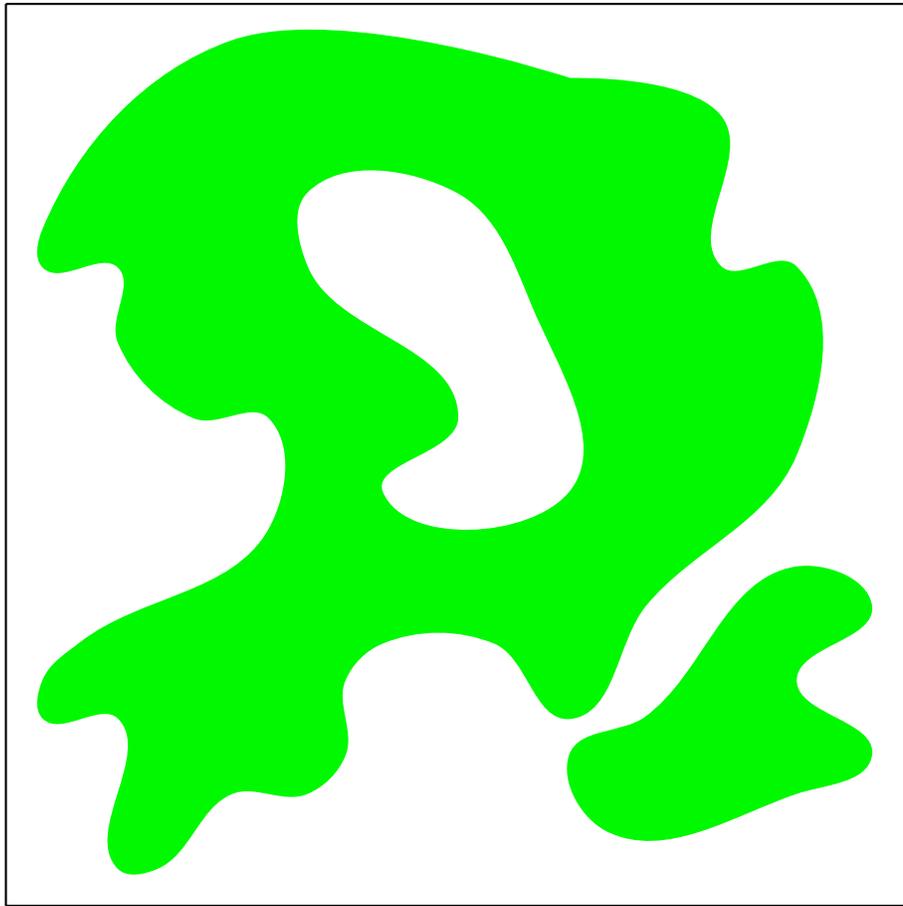
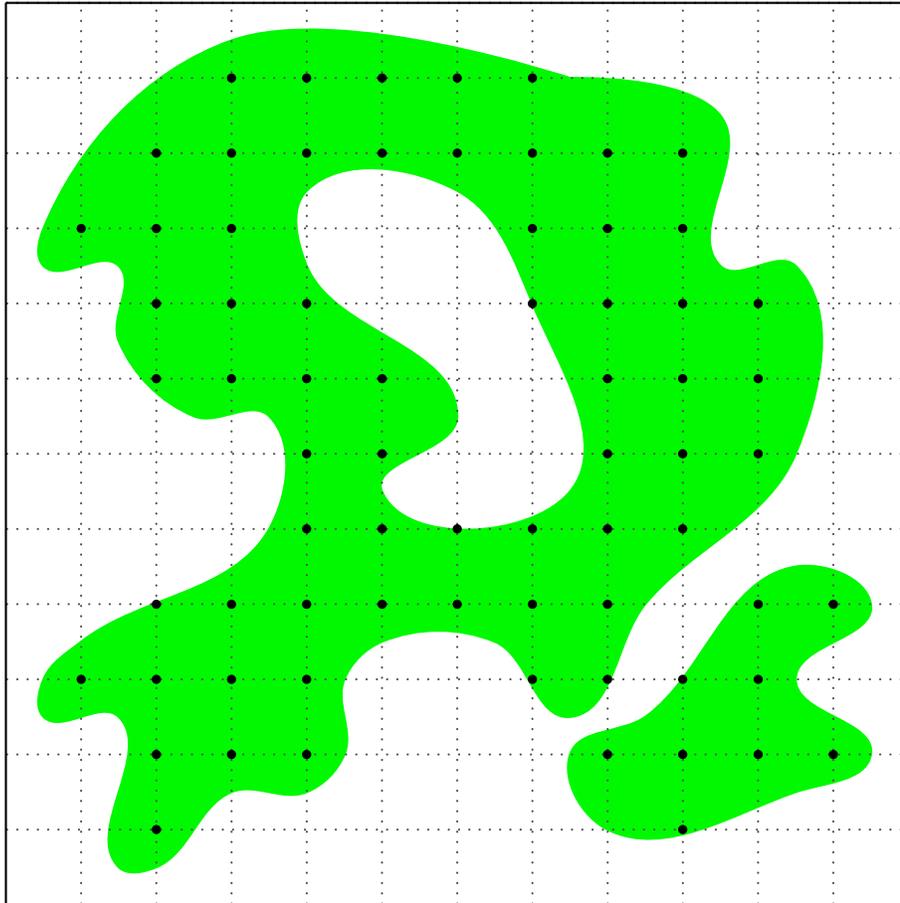


A boundary approximation algorithm.

William K. Allard

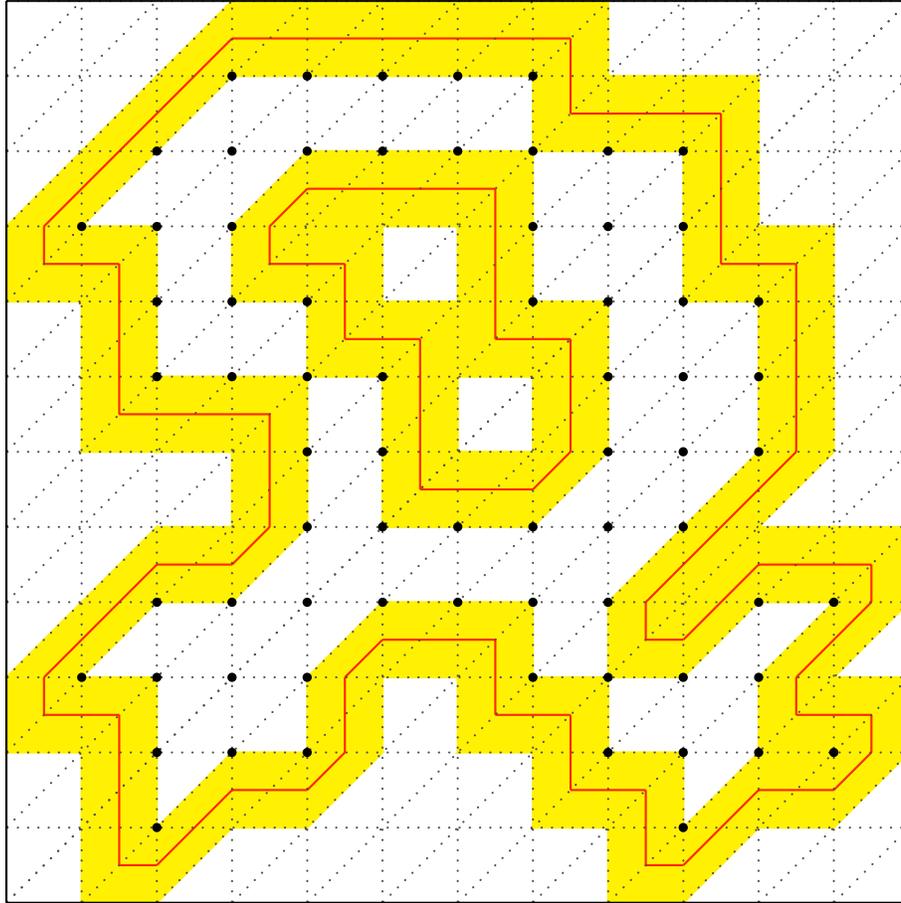


Let R be the planar region colored green inside the square with the black border.

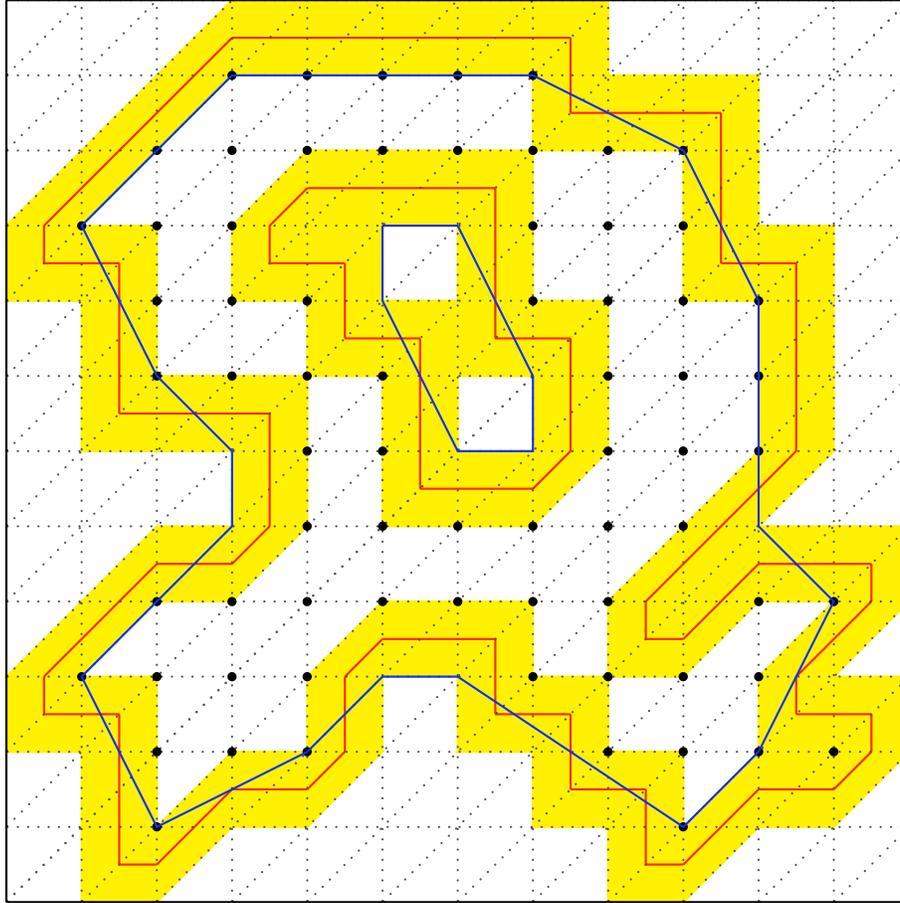


Lay down the rectangular grid as show. Mark each grid point lying inside the green region black.

Our goal is to compute the length of the boundary of the green region given the black dots. It is easy to see that the error will be at least $O(h)$ where h is the mesh spacing of the grid.

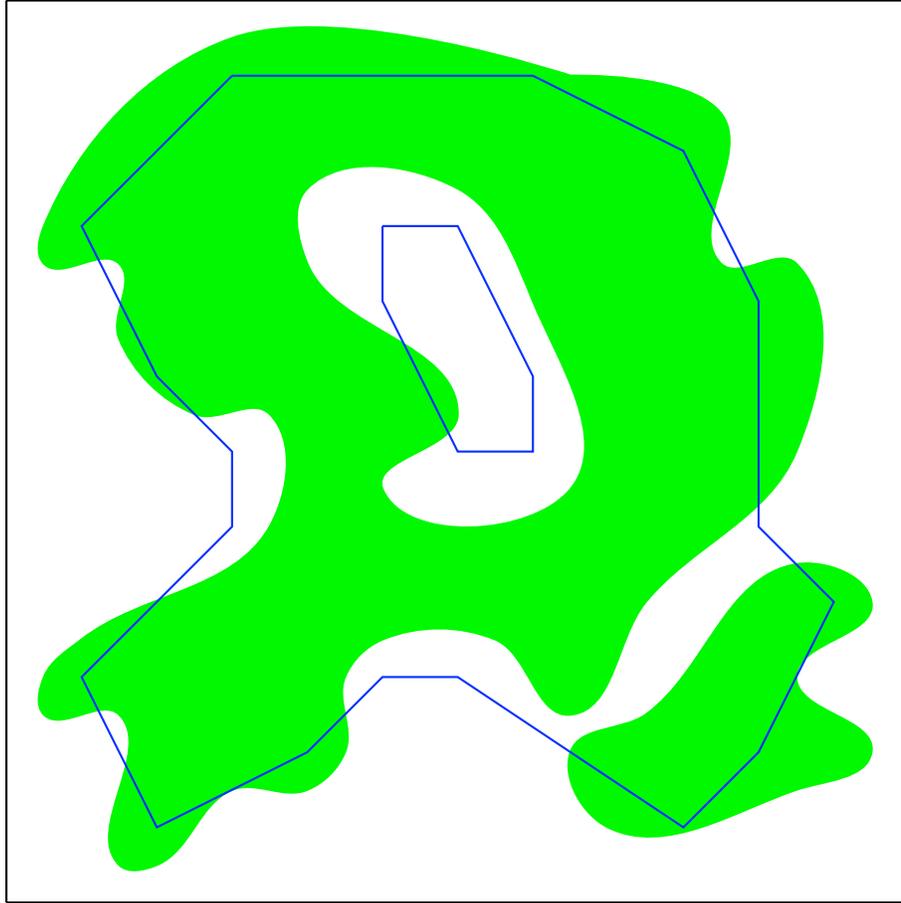


We introduce diagonals obtaining thereby a triangulation of the square as shown. We say a triangle is a *boundary triangle* if some of its vertices are black and some are not. The boundary triangles are colored yellow. Finally, the red contour is $\{f = 1/2\}$ where f is the the function which is piecewise linear with respect to the triangulation, one on the black vertices and zero on the remaining vertices. It is easy to see that the length of the red contour is not a good approximation to the length of the boundary of the green region.



The blue contour is obtained by “pulling tight” on the red contour while staying within the yellow region which is the union of the boundary triangles. More precisely, the blue contour minimizes length among contours with stay within the yellow region, are homotopic to the red contour and have vertices among the vertices of the grid. *I have already shown that the length of such a (blue) contour will differ from the length of the boundary of the green region by $KO(h)$ where K is a small multiple of the maximum absolute curvature of the boundary of the green region, although to show this one needs an additional but modest hypothesis on the green region.*

The red contour was computed. The blue contour was obtained by applying by hand a length shortening algorithm to the red contour. I have yet to program this algorithm or prove that it works. It will be quite easy to program and will run in $O(N)$ time where N is the number of segments in the red contour. I am reasonably sure I will be able to prove that the output of the algorithm is a minimizer. In lieu of this one could use other optimization algorithms to obtain a minimizer, but these may take considerably longer to execute.



Here again is the blue contour whose length is approximately the length of the boundary of the green region. To see more convincingly that the process given above for approximating the lengths of boundaries does indeed work as stated one needs grids that are bit finer than that considered here. For example, halving h will cause the smaller component on the lower right to be separated from the larger component on the upper left. But with a smaller h the preceding picture becomes very busy.

Indeed, to obtain better results the h should not exceed the so-called *reach* of the boundary, which, by definition, is the supremum of the the positive R such that $(x, r) \mapsto x + rN(x)$ is one to one as x ranges over the boundary, $0 < r < R$ and N is the unit outward pointing normal to the boundary. The reach of the green region is, roughly, $h/8$.