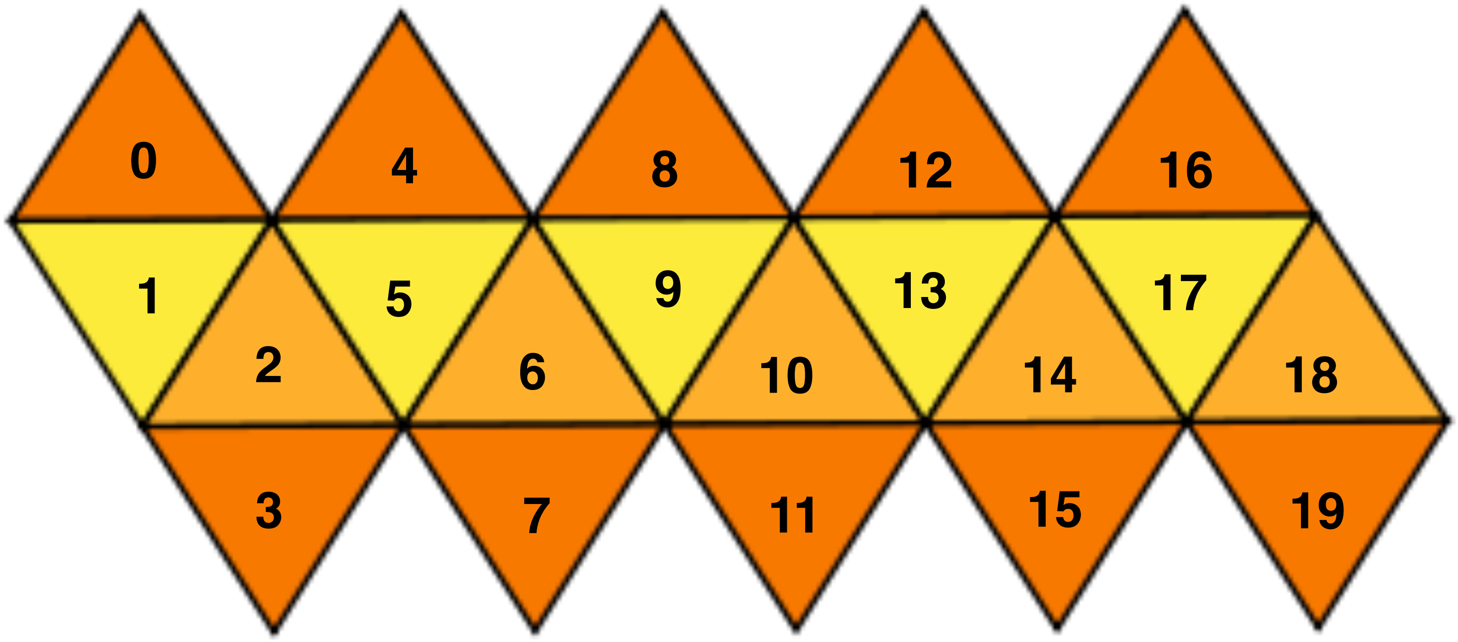
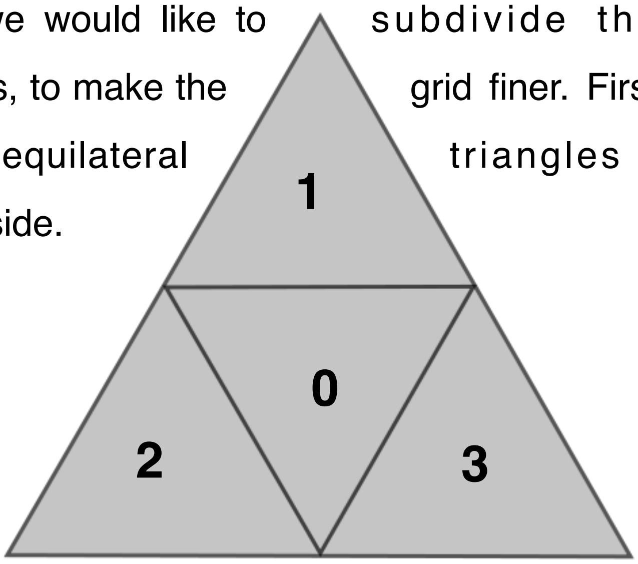


To build an icosahedron, take the above net, join the triangles in the top together (0 to 4 to 8 to 12 to 16 to 6), those in the bottom together in an analogous manner and join triangle 1 to triangle 18.

Our interest lies in using the icosahedron to approximate a sphere or, rather, the points on a sphere in order to simulate reaction-diffusion systems on the sphere. For this reason, we would like to subdivide the triangles of the icosahedron: that is, to make the grid finer. First step: subdivide each triangle in four equilateral triangles by connecting the midpoints of each side.



Now each of the original triangles had it’s neighbors specified in a particular order (0th through 2nd). The subdivision implemented (and visualized above) numbers the SubSites as in the diagram when the 0th neighbor is at the bottom of the parent triangle, the 1st to the left and the 2nd to the right.

So then: the only step now necessary is to designate neighbors for each SubSite. I did this by brute force: for the 0th SubSite of each parent triangle the solution is clear (it’s neighbors are the 3 other children of its parent); for the others, I determined which of its parent’s edges it lay at the vertex of, who the “parent neighbors" were and, finally, who the neighboring SubSites were (this last step was done by determining which edge of the “neighbor parent” was involved).

Questions:

1. Does this make sense? In particular, how to define the neighbors of the SubSites? Would it help if I included an example of this in the write-up?
2. Is the code correct? How can we “easily” check this?
3. Can we use the same code to make the grid finer? That is, can it be used recursively? I think “yes”, once I turn it into a function with argument “SiteCollection” that spits out a “SiteCollection”.