Only lines of gcode with an X or Y need to be modified, all others can be ignored

Final format for a line should be G(x) X(x) Y(x) A(x) B(x) then possibly also Z(x) C(x) F(x)

Starting Points : (The subscripts are just for identification purposes and will not be included in the final gcode)

Every point will have a corresponding point, this means that every to line segment will also have a corresponding to line segment.

Any code starting points are just ideas and can be disregarded if there is a better way.

Current max straight line distance allowed between any point and its corresponding point is 100mm. Anything less than this will be considered “close” and anything greater than this will be called “far”. The same goes for the distance from a current point to a future point. Greater than 100mm will be “far” and less than 100mm will be “close”.

There are four main situations of moves for X and Y which will need to be accounted for:

1. to > 100mm & to > 100mm
2. to > 100mm & to < 100mm
3. to < 100mm & to < 100mm & to < 100mm
4. to < 100mm & to < 100mm & to > 100mm

For #1: “Cutting the corner” - Bisect the angle between the lines from to and to . will then be placed 100mm from along the bisecting line.

Code starting point:

* from sympy.geometry import Point, Circle, Triangle
* p1, p2, p3 = Point(), Point(), Point ()
* t = Triangle(p1, p2, p3)
* r = 100
* c = Circle(p2, r)
* p4 = c.intersection(t.bisectors()[p2])

p4 =

For #2: Same as #1 but instead of being 100mm from along the bisecting line it will need to be half the distance from to along the bisecting line.

Code starting point:

* from sympy.geometry import Point, Circle, Triangle, Line
* p1, p2, p3 = Point(), Point(), Point ()
* t = Triangle(p1, p2, p3)
* l = Segment(p2, p3)
* r = (l.length)/2
* c = Circle(p2, r)
* p4 = c.intersection(t.bisectors()[p2])

p4 =

For #3: When is close to then the distance from to all the following points will need to be calculated until one is found to be more than 100mm away. Call this point with the last point still within 100mm now called . The total distance traveled by the system will need to calculated and added up between points and . Call this distance for now. If the distance from to is also less than 100mm then the midpoint between and will need to be calculated. Call this point . The distance, , of the line from to will then need to be calculated. Next will need to be divided by to get the distance scaling factor. The length of the line segment from to will then be divided by this scaling factor to determine the length of line from to with being placed that distance from along the line to which was determined earlier. At this point and will now become the new and and this whole process will be run again from the new starting points until reaching a point where is more than 100mm from .

Code starting point:

* Loop checking distance from to is less than 100mm. This loop can also sum the distance traveled as it checks each point in order. Should probably also save the distance from to as its own variable since that number will be needed later.
* Once is found, can be determined by finding the midpoint from and using:
  + from sympy.geometry import Point
  + p1, p2 = Point(), Point()
  + p1.midpoint(p2)
* Divide the total by the segment from to then use that product to divide the segment from to
* That length can then be used as the radius for a circle at point and then intersected with the line from to to find

For #4: Everything about this one will be the same as #3 except that will be placed at the “cutting the corner” location between and . The loop will then continue until coincides with point . This should result in either a #1 or #2 type of movement case.

All new points will need to checked to make sure they are still within 100mm of their corresponding point. If they are more than 100mm away then they will need to me moved along a straight line to 100mm away from . This would probably involve using the .intersection function of a 100mm radius circle centered at intersecting a line from to the current to give the new point.