

Mark as done

Due: Saturday, March 16, 2024, 11:59 PM

Designing Your Own Interpolation Method

In this assignment, you are going to put together your own algorithm for generating a function from a sequence of points.

Objective:

While the last project had a quantifiable goal, this one will be more "creative".

We have seen many different ways to make functions from finite sets of points. While there are many differences between them, there are also common themes: drawing the curves close to (or exactly on) the points, avoiding "unnatural" corners and bends, simplicity of the computation, and flexibility to handle most situations.

Your task is to make one that is your own.

Think of the different contexts that such an automatic process might be used.

- You might want to limit the result to be a certain shape, or pieces of a certain shape.
- You might want to guide the slope and/or curvature to a gentle progression (or avoid certain values).
- You may aim for some measure of efficiency, such as shortest arc length, least acceleration, or even smallest area underneath.
- You can make it a goal to strive for simplicity -- but don't something so simple that it's obvious or uninteresting!

Expectations:

As with project #1, I prioritize demonstration that you have some idea what you're doing. I first and foremost want to see an honest appraisal of your own method. Having said that, to really get A+ marks, you need to do something that impresses me. Taking an already existing method and giving it a small tweak won't get you very far -- unless a lot of work went into proving how the precise tweak you made results in a very notable property!

Your submissions should include:

- Your idea. A description of what it does (or at least what it's supposed to do). Some baseline questions to answer include:
 - Are you building a single function, or is it piecewise?
 - Is your function Cartesian or parametric?
 - Is it "axis-agnostic"? In other words, if the figure of point were rotated, would your functions rotate consistently with them?
 - Do you connect with every point? Try to get as close to them as possible under certain constraints? Follow along the points in a different way? (Even avoid them??)
 - Does your curve have discontinuities? Corners? Certain types of bending? Does it have any pleasing "natural" properties?
- The computation: an explanation of how you went from idea to algorithm. What needed solving? How does your code work? Are there exceptions, and how are those handled? Can you prove anything about it?
- The code. Again, we're flexible about language. However, I will supply you with modules of python code that run some of the interpolation methods that we've seen already. You can reference those in your code (for example, if you don't want to reinvent the spline or best-fit equations to use them). Please attach this as a separate file.

By default, I encourage making a function/method called "my_interpolate(x_list, y_list)" that returns the parameters you need to make the curves, and a function "my_function(x or t, parameters)" that takes those parameters and draws them.

(It may be possible to come up with something that's so far out of the box that reducing it to a mere pair functions isn't good enough, ?

to implement it. If this is the case, I would expect a detailed documentation.)

- You must apply your method to a collection of preselected examples. The coordinates to run can be found in the attached demonstration.txt. You may assume that they are all ordered with strictly increasing x value. Produce a graph of each one, demonstrating the result of applying your method. You may use whatever rendering tools you like, as long as it's clear what's happening. You may embed it in your documentation or attach it separately (the examples have demonstrations of one way to do this in LaTeX)

Of course, if you want to include additional graphs to really show what it can do, you are welcome.

- Possible issues your code might have. Are there any (hopefully rare) cases that might break it? Is there any risk of inconsistent or off-the-graph results? Do you run an iteration that might not resolve?
Even if everything does work right, are there features that some might not like about your results? Does it use random numbers, or respond chaotically to small changes? Does it treat some points as more important than others? Does it veer wildly on the ends?
- Possible extensions to your idea. What fixes might exist to outstanding issues? In what small ways might someone wish to change your idea? Is it possible to use different curves in place of yours to get similar results? Can you think of a way to work in more degrees of freedom (or fewer)?

Keep it in moderation

It can be hard to predict how much work an idea can be to implement. It may be that your brilliant scheme ends up being too simple to really show me much of any work. Or, just the opposite: it might be that a simple idea leads down an impossibly difficult rabbit hole. It may be that you have to change or even scrap your idea altogether.

Try to get started early, keep an eye on the calendar, and avoid feature creep.

Working in groups:

(Same rules as before)

You are encouraged to work in groups, but it is not required. If you do, make sure you credit the names of all the other members of your group on your work. Your submission does not have to be identical across your group.

Examples:

Attached are a couple examples.

- Example 1 uses a best-fit *pair* of lines that intersect to make a piecewise function of "coterminal rays". The goal is to use the flexibility of this corner to reduce the least squared vertical distances away from the points. While the method will not always place the corner in the optimal location (which can be very complicated to find), it will feature the best choice of rays coming out of it.
- Example 2 takes consecutive triplets of points and ties them together with circular arcs. The goal is to hit each point exactly using only a certain shape. This is also an example of project scope: originally I wanted to write a report for algorithm that connected pairs of lines with *tangent* arcs. However, the code and write-up proved so long and full of problems that I scaled back to this simpler idea. The previous idea is mentioned under "possible extensions".

 [demonstration.txt](#)

 [example 2.1.pdf](#)


 [example 2.1.py](#)

 [example 2.1.tex](#)

 [example 2.2.pdf](#)

 [example 2.2.py](#)

 [example 2.2.tex](#)

 [method - best fit line.py](#)

 [method - best fit parabola.py](#)

 [method - best fit sinusoid.py](#)

February 12 2024, 11:45 AM

February 12 2024, 11:45 AM

February 12 2024, 11:45 AM

February 12 2024, 11:45 AM

February 12 2024, 11:45 AM

February 12 2024, 11:45 AM

February 12 2024, 11:45 AM

February 12 2024, 11:45 AM

February 12 2024, 11:45 AM

February 12 2024, 11:45 AM

Add submission