AP3

Murmani Akhaladze

TarnNished

In this project I wrote a small python program that compares exact and numerical (finite difference) derivatives for two different functions. One of them has one variable and another has two. Using both methods, the code finds tangent line or plane and also checks how the step size h changes the accuracy.

1. Functions

I choosed these two functions:

$$f(x) = \sin(x) + x^2, \qquad g(x, y) = x^2 + y^2 - xy.$$

They are simple but not too boring, and both have curvy shapes so it's easier to see how tangent and normal behave. I used `sympy` library for symbolic math because it can take exact derivatives very easy.

2. Exact Derivatives

For the first one $f(x)$ the exact derivative is

$$f'(x) = \cos(x) + 2x,$$

and for the second one the gradient (two partials) is

$$\nabla g(x, y) = (2x - y,\ 2y - x).$$

I calculated them at some point like $x_0 = 1.0$, $y_0 = 2.0$ and used those to draw tangent line and plane. Tangent line is just a small line that touches the curve and goes with same slope, and tangent plane does same thing but in 3D.

3. Finite Difference Derivatives

Then I did the same thing but numerically using finite difference. The formula was

$$f'(x_0) \approx \frac{f(x_0 + h) - f(x_0 - h)}{2h}.$$

I also used this for the partials of $g(x, y)$. Central difference gives smaller error because it's symmetric. I made two functions called `finite_diff_1d` and `finite_diff_2d` so they can be reused for any other function.

4. Error Test

To check how good the method works, I tried different step sizes $h = 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}$. For each h I compared my finite diff result with the exact one and measured the error

$$Error = |f'_{approx} - f'_{exact}|.$$

Then I printed a small table and also plotted the error in log scale. What I found is when h is too big, accuracy is bad. When it's too small, rounding errors in the computer start messing it up again. So the middle values of h are the best. This shows the classic trade-off in numerical differentiation.

5. Tangent Visualization

The code also draws the function and tangent line for the 1D case, and surface with tangent plane for 2D case. For 2D I used `matplotlib` 3D plotting. You can see clearly that the plane just touches the surface at the point and has same direction as the gradient. That gradient is basically the normal vector of that plane.

6. Summary

So, in short, my code:

- defines two functions (one 1D, one 2D),
- finds exact and finite diff derivatives,
- builds tangent line and plane,
- compares accuracy for different h,
- and finally plots everything.

It fully matches what Problem 3.1 asked for and also shows how small changes in h affect numerical accuracy. Overall, I think the code works fine and results make sense.