AMS 562 Homework 2

Due: Friday, 10/05, 11:59 pm

September 28, 2018

Part I: Numerical Differentiation:

For this part, you need to write a program that compute the derivative of the sin function at $x = \frac{\pi}{4}$, i.e. compute $y'|_{\frac{\pi}{4}}$, where $y = \sin(x)$. A typical way to do this is to use *finite difference formula*, i.e.

$$y = f(x) \tag{1}$$

$$y'_h = \frac{f(x+h) - f(x)}{h} \tag{2}$$

From college calculus, we know that as h approaches to 0, the resulting value will converge to the true derivative, i.e. $y' = \lim_{h\to 0} y'_h$.

The parameter h should be passed in as the **first** command line argument (excluding the executable binary), i.e. `argv[1]`. To convert c-string to floating number, you need to call `std::atof` that is defined in `<cstdlib>`.

In addition, a more accurate solution can be obtained by utilizing the *center difference scheme*, i.e.

$$y'_{2h} = \frac{f(x+h) - f(x-h)}{2h} \tag{3}$$

Sanity check of `argc` must be performed. Compute the derivative of sin at $\frac{\pi}{4}$ using both of these two schemes, and compare the performance by analyzing the absolute errors against the analytic value, i.e. $\cos(\frac{\pi}{4})$. Run the program with $h = 1\mathrm{e}^{-1}$, $1\mathrm{e}^{-2}$, $1\mathrm{e}^{-3}$, $1\mathrm{e}^{-4}$..., what about $h = 1\mathrm{e}^{-16}$?

Hints:

The sin function is declared in the *standard math library*, i.e. `<cmath>`, once you include it, you can do `const double sin0=std::sin(0);`. For $\cos(\frac{\pi}{4})$ and π, it's a good idea to compute them offline and store the values in your program, e.g. `const double pi=3.141592653589793, cos_pi_4=0.7071067811865476;`, of course, the cos function is define in `<cmath>`.

Part II: Finding the Extreme Arc Lengths on Sphere

For part 2, you need to write a program to determine the extreme *arc lengths* of a random sample point cloud on the **unit sphere**, i.e. $x^2 + y^2 + z^2 = 1$. The definition of arc length of two points on a sphere is the length of the arc that passes through the *great circle* that is defined by the two points.

The arc length can be computed with the following formula:

$$\gamma_{12} = \cos^{-1}(\boldsymbol{n}_1 \cdot \boldsymbol{n}_2) \tag{4}$$

Where $\boldsymbol{n}$ is the unit outward normal vectors on the points on the unit sphere and $\cdot$ is the inner product, i.e. $\boldsymbol{a} \cdot \boldsymbol{b} = a_1 b_1 + a_2 b_2 + a_3 b_3$. Notice the outward normal vectors and the points have the same coordinates for the unit sphere.

The number of the points is passed in as the first command line argument, say N. Then you need to **dynamically** allocate three arrays, `float *x, *y, *z;`, of size N and pass them to the black-box function that I write to get a collection of points on the unit sphere. Finally, write a `for`/`while` loop to find the maximal and minimal arc lengths and their corresponding indices with respect to the first point in the point cloud, i.e. `x[0]`, `y[0]`, and `z[0]` (the loop should start from the second point). Print out the results in the terminal.

Hints:

The $\cos^{-1}$, i.e. arccos, is also defined in the `<cmath>` library, simply do `double arc = std::acos(0);`, which gives $\frac{\pi}{2}$. Moreover, don't forget to relax the memory of the dynamic arrays.