Activity Sheet Week 3

Grade: check (upgrade to check + by addressing the indicated questions)

Comparing Floating Point Numbers

1. Initializes two floating point numbers and then sees if they are equal using the == operator. If they are it prints “x1 and x2 are equal” and if they are not equal It prints “x1 and x2 are not equal”.
2. The errors are different for both floating point numbers because they were calculated. Applying an error to a floating point number will make it differ from exactly what it should be. The original errors in calculation stem from the machine precision. To check what x1 and x2 are I added two cout statements printing x1 and x2 with setprecision(10) in front so I could see the difference in decimals.
3. To see if they are equal instead of x1 == x2 you would want to check if

(x1-eps) < x2 < (x1+eps). Meaning x2 is in between x1 – eps and x1 + eps because that would account for the error in machine and calculation.

Numerical Derivatives Pass 1

1. I don’t fully understand the fabs, why we take the log and why in the forward diff function, the parameter f is a pointer and it receives an address as it should but to use it its not dereferenced. I believe the log of h, log of the percent difference for f diff and the log of percent difference for c diff are being printed into a file.
2. Done
3. Done
4. Yes, I believe so. The central difference algorithm is better because it has the ability to have the smallest relative error between the two algorithms.
5. For the forward h I plugged in 2-23 for Em and e-1 for f(2) which gave me about .0008, which is off by a factor of about 2. I estimate f\_diff’s optimal h to be at around log(-8) I think you mean h = 10^-8, or log h = -8. and because of how the log graph looks the difference between -8 and the real location could account for the factor of 2, so I believe this is a good match. I plugged in same thing for machine precision and -1e-1 for f(3) but did not get as close of a match. If the graph showed it was closer to -4 it would be very close.
6. Switching to single precision would shift the minimas to the left. Would the slopes change?

Makefiles for multiple project files

1. Yes, the functions are clearly defined in the file. It is easier to have them in a separate file for debugging and organization. If one of the functions is wrong you only have to look through a single file. It is cleaner for the user as well because they just need to know the type, parameters and function name and nothing about the signature. You can also modify just that file and leave all of the test cases the same in the main program if you want to upgrade the algorithms.
2. Yes, each column is the absolute error for each algorithm.
3. Divided by answer to get relative
4. For trapezoid and simpson’s the relationship between N and relative are pretty clear in their respective regions but in the lower right region it seems to almost take on a uniform distribution. The slope on the log scale for trapezoid equals roughly -2 which matches the relationship between E and number of intervals which is N^-1/2 and same goes for simpsons except the slope is roughly -4.
5. To evenly space the points I multipled by 2 instead of incrementing by 2 with addition.

Finding the approximation error from a log-log plot

1. Turned off the logscale in the plt file and added log10 in front of each answer including the N
2. Trapezoid m = -2. 17748 and simpson’s m = -0.521221
3. The trapezoid fit is as the answer should be -2 but simpsons is off by a little. Fitting the non linear regions come out to -2.31108 and -1.00486 respectively. Not sure i understand what you mean by the non-linear regions. What slopes do you get in a) the region where error is dominated by truncation/algorithm error, and b) the region where you’re dominated by roundoff?