Problem I.

Modify `numderivative.cc` to calculate the derivative \( \exp(x)' \) at \( x = 1 \) to within 0.05 % relative error by using: 
\[
 f'(x) \approx \frac{f(x + h/2) - f(x - h/2)}{h}.
\]
What step size \( h \) will you need? What is an advantage of the above formula compared to the one we used in class [i.e. \( f'(x) \approx \frac{f(x + h) - f(x)}{h} \)]? Using `numderivative.cc`, find the optimum \( h \), and compare it to the optimum for the original `numderivative.cc`.

Solution

With this HW, it is Ok to arrive at the required \( h \) (and the key conclusions) ”experimentally”, by testing the code with various \( h \). One obvious advantage of the above formula is that it is symmetric around \( x \), and so is likely to produce a more accurate result for the same \( h \). Direct tests confirm the expectation. The moral of the story is that while there may be many mathematically equivalent ways to compute the same thing, some of them may work much better when used in a numerical computation. A good algorithm is worth the extra work!

The largest step size for which the relative error falls below 0.05 percent is \( h = 1 \times 10^{-1} \) with relative error of 0.04167

Problem II.

One can come up with more and more clever approximations for \( f'(x) \): the difference between the approximate \( f'(x) \) and the mathematically exact derivative can be made arbitrary small for a given small value of \( h \). In principle, there is no limit to how accurate your approximation can be in this respect. Now you use this approximate formula (which always contains subtraction of functional values) to numerically estimate \( f'(x) \) on your computer. Is there a limit to the accuracy of the result? Why?

0.1 Solution

Yes. Machine epsilon sets that limit. No matter what you do, the numbers you subtract to get the derivative will, at some point, become indistinguishable for the computer.
Problem III.

In numderivative.cc replace the "exp(x)" with "sin(1/x)" where appropriate to obtain a numerical estimate for the derivative of \( f(x) = \sin(1/x) \) at

a) \( x = 1/\pi \). Choose "h" so that the result is accurate to within at least 4 decimal points. What is your calculated result?

Solution

Use your table of generated results to find the specific h, where numerical solution matches exact solution up to four digits. Notice that the error goes up again, as h decreases further. This is due to rounding error becoming more dominant over truncation error.

The largest h for which gives the correct accuracy is \( h = 1 \times 10^{-6} \) with the value of numerical derivative : 9.8595565

b) What happens when you try the same code for \( x = 10^{-20}/\pi \)? Why? Use the chain rule to re-formulate the problem into a mathematically equivalent one that is free from the defect, modify the code, and re-compute. What do you get now?

Solution

The key is to recognize that \( \sin(1/x) \) is very pathological around \( x = 0 \): it goes hey-wire, and so do all of its derivatives, which also diverge as \( x \to 0 \). As a result, the errors in a straightforward numerical estimate of \( f'(x) \) become huge. (Recall that the expression for the total error contains \( f''(x) \)). Applying any numerical procedure around such pathology is a bad idea: one needs to reformulate the problem to single out the pathology (to be treated analytically), while applying numerical procedure to the remaining non-pathological part of the function. By the chain rule \( \sin(1/x)' = -(1/x^2) \times \sin(t)' \), where the derivative of \( \sin(t) \) is evaluated at \( t = 1/x = 10^{20}\pi \). Evaluation of \(- (1/x^2)\) is straightforward, even for small \( x \). But now, since \( \sin(t) \) is not pathological, numerical differentiation will work. It is a good idea to shift the argument away from the large \( t \): \( \sin(10^{20}\pi) = \sin(2\pi n) = \sin(0) \), so you can evaluate at \( t = 0 \).

It is also possible to approximate \((1/x)'\) with one-sided or two-sided approximation formulas and use it in the chain rule. In this case, of course, smaller values of \( h \) are needed to bring the relative error within the desired percentage.

Another option is a purely analytical derivative: \(- (1/x^2) \times \cos(x)\). Not as general as the above, but will still work. Those who recognize the pathology and find any way around it receive full points.
Common problems: Many students failed to identify the reason they did not get small errors the first time. You can just point out that the approximation with two points is not close to the derivative (tangent line) because the function is highly oscillatory at that point. Also, it is fine to try different ranges of $h$ values for this problem to see the behavior of the error, but you still cannot get to the desired accuracy without modifying the original problem.