Complex-step Differentiation

Motivation

Simple method for approximating derivative of $f(x)$:

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

for some small h.

Main disadvantage: you have to evaluate f twice and take a difference, which could lead to loss in significant digits with a computer (catastrophic cancellation).

Example:

$$\begin{gathered} h = 10^{-9} \\ f(x+h) \approx 0.123456789123456789 \\ f(x) \approx 0.123456789000000000 \\ \frac{f(x+h) - f(x)}{h} \approx 0.123456789 \end{gathered}$$

But computers can't hold infinitely many decimal places, so they might do something like this:

$$\begin{gathered} h = 10^{-9} \\ f(x+h) \approx 0.123456789123456789 \text{ which rounds to } 0.123456789 \\ f(x) \approx 0.123456789000000000 \text{ which rounds to } 0.123456789 \\ \frac{f(x+h) - f(x)}{h} = 0 \end{gathered}$$

The Method

Prerequisite: $f(z)$ is a **complex analytic function** (sometimes also called holomorphic), meaning it is "at every point of its domain, complex differentiable in a neighborhood of the point."

Consider some complex number $x_0 + ih$, where h is once again small. Consider the Taylor series expansion of $f(z)$ at $x_0 + ih$:

$$f(x_0 + ih) = f(x_0) + ih \cdot f'(x_0) - h^2 \cdot f''(x_0)/2! - ih^3 \cdot f^{(3)}(x_0)/3! + \dots$$

Take the imaginary part of both sides, divide by h, omit the trailing terms of the Taylor series expansion since it's an approximation:

$$\begin{gathered} \Im(f(x_0 + ih)) \approx h \cdot f'(x_0) \\ f'(x_0) \approx \frac{\Im(f(x_0 + ih))}{h} \end{gathered}$$

Benchmarking

One of the articles I referenced does a benchmark of evaluations of the derivative of $f(x) = \frac{e^x}{(cosx)^3 + (sinx)^3}$ at $x = \pi/4$ and yields the following results:

h	complex step	finite differerence
0.100000000000000	3.144276040634560	3.061511866568119
0.010000000000000	3.102180075411270	3.101352937655877
0.001000000000000	3.101770529535847	3.101762258158169
0.000100000000000	3.101766435192940	3.101766352480162
0.000010000000000	3.101766394249620	3.101766393398542
0.000001000000000	3.101766393840188	3.101766393509564
0.000000100000000	3.101766393836091	3.101766394841832
0.000000010000000	3.101766393836053	3.101766443691645
0.000000001000000	3.101766393836052	3.101766399282724
0.000000000100000	3.101766393836052	3.101763290658255
0.000000000010000	3.101766393836053	3.101785495118747
0.000000000001000	3.101766393836053	3.101963130802687
0.000000000000100	3.101766393836052	3.097522238704187
0.000000000000010	3.101766393836053	3.108624468950438
0.000000000000001	3.101766393836052	3.108624468950438
0.000000000000000	3.101766393836053	8.881784197001252

And just because I can, I decided to write my own code in C++ to verify this:

```
1.0000000000000000000 -0.95985465566562362742 2.79229414094489963578
0.1000000000000000000 3.14427604063455769169 3.06151186656811865813
0.0100000000000000000 3.10218007541126962619 3.10135293765587006370
0.0010000000000000000 3.10177052953584657695 3.10176225815879651909
0.0001000000000000000 3.10176643519293806367 3.10176635247917100954
0.0000100000000000000 3.10176639424962053760 3.10176639342249179190
0.0000010000000000000 3.10176639384018737466 3.10176639383189737425
0.0000001000000000000 3.10176639383609304194 3.10176639383569208185
0.0000000100000000000 3.10176639383605209921 3.10176639386171293399
0.0000000010000000000 3.10176639383605168916 3.10176639397013315146
0.0000000001000000000 3.10176639383605168504 3.10176639364487249971
0.0000000000100000000 3.10176639383605168569 3.10176642400253332909
0.0000000000010000000 3.10176639383605168569 3.10176602284772950945
0.0000000000001000000 3.10176639383605168569 3.10176363760295004134
0.0000000000000100000 3.10176639383605168569 3.10176146919860507034
0.0000000000000010000 3.10176639383605168591 3.10179399526377963547
0.0000000000000001000 3.10176639383605168613 3.09973401113605717704
0.0000000000000000100 3.10176639383605168569 3.08997619158368763714
0.0000000000000000010 3.10176639383605168504 2.81892564846231152885
0.0000000000000000001 3.10176639383605168591 6.50521303491302660491
```

Generalizations

From Wikipedia:

$$f^{(n)}(x) \approx \frac{\mathcal{C}^{(n)}_{n^2-1}(f(x + i^{(1)}h + \cdots + i^{(n)}h))}{h^n}$$

where $\mathcal{C}_k^{(n)}$ and $i^{(k)}$ are stuff dealing with multicomplex numbers. It's a whole other rabbit hole that seems intriguing and mostly went over my head. The fact that the Wikipedia article for multicomplex numbers is just 5 paragraphs long doesn't help.

So what is the best method/what are other options?

In terms of numerical differentiation methods:

- Taking multiple steps
- Differentiation by interpolation
- Something about Laplace transform

There are also other classes of differentiation methods altogether:

- Numerical Differentiation: use small h to approximate
- Symbolic Differentiation: computer actually attempts to do stuff with variables and manipulate equations
    - WolframAlpha/Mathematica
- Automatic Differentiation
    - Wikipedia: "AD exploits the fact that every computer program, no matter how complicated, executes a sequence of elementary arithmetic operations (addition, subtraction, multiplication, division, etc.) and elementary functions (exp, log, sin, cos, etc.). By applying the chain rule repeatedly to these operations, derivatives of arbitrary order can be computed automatically, accurately to working precision, and using at most a small constant factor more arithmetic operations than the original program."
    - TLDR: breaks down into elementary operations, spams chain rule
    - According to some comment I read, the complex-step differentiation method is simply using the complex numbers as "approximate dual numbers" in automatic differentiation
    - Dual numbers: $a + b\epsilon$, where $\epsilon^2 = 0$, but they're hypercomplex numbers so $\epsilon \neq 0$

Explanation of different methods: https://stackoverflow.com/questions/43455320/difference-between-symbolic-differentiation-and-automatic-differentiation

- Second answer claims symbolic and automatic differentiation are equivalent, the two seem to do the same thing just with different representations of what they're operating on

Sources

https://math.stackexchange.com/a/1080292

https://blogs.mathworks.com/cleve/2013/10/14/complex-step-differentiation/

https://timvieira.github.io/blog/post/2014/08/07/complex-step-derivative/

http://www.johnlapeyre.com/posts/complex-step-differentiation/

https://en.wikipedia.org/wiki/Numerical_differentiation#Complex-variable_methods

https://en.wikipedia.org/wiki/Automatic_differentiation#Automatic_differentiation_using_dual_numbers

http://www2.math.umd.edu/~dlevy/classes/amsc466/lecture-notes/differentiation-chap.pdf

https://stackoverflow.com/questions/43455320/difference-between-symbolic-differentiation-and-automatic-differentiation