J. Chem. Sci., Vol. 121, No. 5, September 2009, pp. 935–950. © Indian Academy of Sciences.

An improved numerical approximation for the first derivative[†]

ROBERT DE LEVIE
Chemistry Department, Bowdoin College, Brunswick ME 04011, USA
e-mail: rdelevie@bowdoin.edu

Abstract. The traditional numerical computation of the first derivative $f'(x)$ of a given function $f(x)$ of a single argument x by central differencing is known to involve aspects of both accuracy and precision. By analysing both we arrive at an algorithm that closely approximates the most accurate answer obtainable by this method, typically with at least 9 accurate decimals, while preserving a minimal footprint. The results apply to software based on the IEEE-754 specification, and are illustrated with Excel.

Keywords. Numerical differentiation; algorithms.

1. Introduction

Since Newton and Leibnitz introduced calculus, differentiation and differential equations have been at the core of physics, and of virtually all physical sciences. In formal mathematics, differentiation is usually the more straightforward process; consequently, most attention and ingenuity is typically lavished on integration. In numerical analysis, the reverse is true: numerical integration is relatively straightforward, while numerical differentiation is not. Yet the latter field has generally been underserved, and an important component of it, *central differencing*, will therefore be discussed here in more detail than would otherwise be justified. And we will see how numerical differentiation, even of closed-form algebraic expressions, introduces statistical effects, and forces us to consider the interplay of systematic and indeterminate errors. Moreover, we will illustrate how even the lowly spreadsheet can be used to find interesting new solutions to old problems. Incidentally, the approach described below is equally applicable to its less accurate but sometimes unavoidable companions, *forward* and *backward differencing*, but the latter will not be discussed here in order to keep the present communication within manageable size.

We will here consider taking the numerical derivative *dF/dx* of a mathematically specified function $F(x)$; in the case of a function of multiple arguments, i.e. for the function $F(\mathrm{x})$ where x is a vector of various arguments x_i, numerical differentiation yields the partial derivatives that define its Jacobian matrix. The first derivative, as the best linear approximation to a function, is used in many root-finding methods, such as the Newton–Raphson algorithm, as well as in all optimization schemes, such as in the Levenberg[1]-Marquardt[2] or generalized reduced gradient[3,4] routines. While it is in principle possible to use analytical expressions for such derivatives, this is often impractical, and therefore bypassed, e.g. the original Fortran version of the Lasdon–Waren routine[4] includes the option of user-provided analytic derivatives, but its popular spreadsheet implementation Solver only uses numerical derivatives. (One would need to buy the special Premium Solver from Frontline Systems to allow analytic derivatives.)

Numerical differentiation is usually inspired by the mathematical definition of the derivative as a limit,

$$\frac{\mathrm{d}f(x)}{\mathrm{d}x} = \lim_{\delta\to 0}\frac{f(x+\delta)-f(x)}{\delta}, \tag{1}$$

which has led to three main types of closely related approaches, called forward, central, and backward differencing. In their simplest form they use the expressions

$$\frac{\mathrm{d}f(x)}{\mathrm{d}x} \approx \frac{f(x+\delta)-f(x)}{\delta} \quad \text{for} \quad \delta << x \tag{2}$$

$$\frac{\mathrm{d}f(x)}{\mathrm{d}x} \approx \frac{f(x+\delta)-f(x-\delta)}{2\delta} \quad \text{for} \quad \delta << x \tag{3}$$

and

[†]Dedicated to the memory of the late Professor S K Rangarajan

$$\frac{\mathrm{d}f(x)}{\mathrm{d}x} \approx \frac{f(x)-f(x-\delta)}{\delta} \quad \text{for} \quad \delta << x, \tag{4}$$

respectively, replacing the mathematical limit $\delta \to 0$ by a sufficiently small numerical difference $\delta << x$. One can extend these difference formulas, all limited to $\delta << x$, by including more input data, typically at equal intervals δ, such as

$$\frac{\mathrm{d}f(x)}{\mathrm{d}x} \approx \frac{-f(x+2\delta)+4f(x+\delta)-3f(x)}{2\delta}, \tag{5}$$

$$\frac{\mathrm{d}f(x)}{\mathrm{d}x} \approx \frac{\begin{bmatrix}-f(x+2\delta)+8f(x+\delta)\\ -8f(x-\delta)+f(x-2\delta)\end{bmatrix}}{12\delta}, \tag{6}$$

and

$$\frac{\mathrm{d}f(x)}{\mathrm{d}x} \approx \frac{\begin{bmatrix}25f(x)-48f(x-\delta)+36f(x-2\delta)\\ -16f(x-3\delta)+3f(x-4\delta)\end{bmatrix}}{12\delta}. \tag{7}$$

Central differencing, as in (3) and (6), is the most accurate of the above three methods for a given number of equidistant data samples; forward and backward differencing are usually reserved for special cases where the function either has a singularity and/or discontinuity at x (as tan(x) has at $x = \pi/2$), or is undefined at and/or on one side of x (as log(x) is at $x \le 0$, and $\sqrt{x}$ at $x < 0$). Forward and backward differencing are part of a larger group of what might be called *lateral* (as distinct from central) differencing methods, including extrapolative differencing that might be appropriate for a singularity. Here we will only consider central differencing, because the corresponding treatments for lateral differencing involve no new concepts, and follow similar logic.

We will assume that the function $f(x)$ exists and is differentiable at all sample values $x + n\delta$ used (where, in order to keep the math simple, n will be integer), and that the same applies also to any of its higher derivatives that we may need. For a general description of such differentiation routines the reader is referred to the *Numerical Recipes.*[5] Here we will focus on *compact* central differencing, i.e. on those central difference expressions that use a minimal number of equidistant samples to achieve a given order of magnitude of their associated errors.

The computer implementation of such difference equations leads to an interesting dilemma. As δ approaches zero, the differences in the numerators of these equations, such as $f(x + \delta) - f(x - \delta)$ in (3) or the corresponding difference $f_1 - f_{-1}$ in the expression for $j = 3$ in table 1, become so small that their numerical evaluation becomes subject to cancellation errors. Such errors are pseudo-random, i.e. they appear to be random, even though they are fully deterministic in the sense of being reproducible for a given computer, software, and input, in the same way as computer-generated pseudo-random numbers are fully deterministic. Cancellation errors lead to ostensibly *imprecise* results, and can indeed be reduced by using extended numerical precision.

On the other hand, when δ is too large, systematic errors resulting from any nonlinearity of $f(x)$ as a function of x will dominate, causing *inaccurate* answers. At the optimum δ-value, the total error (pseudo-random noise plus systematic bias) should be minimized in order to obtain the highest achievable accuracy. This requires modelling of both types of error, which will therefore be our main focus.

2. Systematic errors

In order to find the *systematic* errors we use the Taylor expansions

$$f(x+\delta)\Big|_{x=x_0} = f(x)\Big|_{x=x_0} + \frac{\delta^3}{3!}\frac{\mathrm{d}^3f(x)}{\mathrm{d}x^3}\Big|_{x=x_0}$$
$$+ \frac{\delta^2}{2!}\frac{\mathrm{d}^2f(x)}{\mathrm{d}x^2}\Big|_{x=x_0} + \frac{\delta^3}{3!}\frac{\mathrm{d}f^3(x)}{\mathrm{d}x^3}\Big|_{x=x_0}$$
$$+ \frac{\delta^4}{4!}\frac{\mathrm{d}^4f(x)}{\mathrm{d}x^4}\Big|_{x=x_0} + \frac{\delta^5}{5!}\frac{\mathrm{d}^5f(x)}{\mathrm{d}x^5}\Big|_{x=x_0} + \ldots \tag{8}$$

and

$$f(x-\delta)\Big|_{x=x_0} = f(x)\Big|_{x=x_0} - \frac{\delta}{1!}\frac{\mathrm{d}f(x)}{\mathrm{d}x}\Big|_{x=x_0}$$
$$+ \frac{\delta^2}{2!}\frac{\mathrm{d}^2f(x)}{\mathrm{d}x^2}\Big|_{x=x_0} - \frac{\delta^3}{3!}\frac{\mathrm{d}f^3(x)}{dx^3}\Big|_{x=x_0}$$
$$+ \frac{\delta^4}{4!}\frac{\mathrm{d}^4f(x)}{\mathrm{d}x^4}\Big|_{x=x_0} - \frac{\delta^5}{5!}\frac{\mathrm{d}^5f(x)}{\mathrm{d}x^5}\Big|_{x=x_0} + \ldots \tag{9}$$

which we combine to

Table 1. Compact central differencing formulas for the first derivative, f_0^{I}, with the leading term of its systematic error, for j = 3(2)17, where the number of data points j listed in the first column includes f_0. The term *compact* indicates use of the smallest possible number j of equidistant data. The results shown in tables 1 through 4 were computed with the spreadsheet approach illustrated in section 9.2.5 of ref. 6. For $j > 9$ this required higher-precision matrix inversion to get sufficiently accurate answers, for which we used Volpi's BigMatrix freeware, see ref. 6 section 11.9.

j	Formula for f_0^{I}	Leading term of systematic error
3	$(-f_{-1} + f_1)/(2\delta)$	$-f^{\mathrm{III}}\,\delta^2/6$
5	$(f_{-2} - 8f_{-1} + 8f_1 - f_2)/(12\delta)$	$+f^{\mathrm{V}}\delta^4/30$
7	$(-f_{-3} + 9f_{-2} - 45f_{-1} + 45f_1 - 9f_2 + f_3)/(60\delta)$	$-f^{\mathrm{VII}}\,\delta^6/140$
9	$(3f_{-4} - 32f_{-3} + 168f_{-2} - 672f_{-1} + 672f_1 - 168f_2 + 32f_3 - 3f_4)/(840\delta)$	$+f^{\mathrm{IX}}\delta^8/630$
11	$(-2f_{-5} + 25f_{-4} - 150f_{-3} + 600f_{-2} - 2100f_{-1} + 2100f_1 - 600f_2 + 150f_3 - 25f_4 + 2f_5)/(2520\delta)$	$+f^{\mathrm{XI}}\,\delta^{10}/2772$
13	$(5f_{-6} - 72f_{-5} + 495f_{-4} + 2200f_{-3} + 7425f_{-2} - 23760f_{-1} + 23760f_1 - 7425f_2 + 2200f_3 - 495f_4 + 72f_5 - 5f_6)/(27720\delta)$	$+f^{\mathrm{XIII}}\,\delta^{12}/12012$
15	$(-15f_{-7} + 245f_{-6} - 1911f_{-5} + 9555f_{-4} - 35035f_{-3} + 105105f_{-2} - 315315f_{-1} + 315315f_1 - 105105f_2 + 35035f_3 - 9555f_4 + 1911f_5 - 245f_6 + 15f_7)/(360360\delta)$	$+f^{\mathrm{XV}}\delta^{14}/51480$
17	$(7f_{-8} - 128f_{-7} + 1120f_{-6} - 6272f_{-5} + 25480f_{-4} - 81536f_{-3} + 224224f_{-2} - 640640f_{-1} + 640640f_1 - 224224f_2 + 81536f_3 - 25480f_4 + 6272f_5 - 1120f_6 + 128f_7 - 7f_8)/(720720\delta)$	$+f^{\mathrm{XVII}}\,\delta^{16}/218790$

$$f_0^{\mathrm{I}} = \frac{-f_{-1}+f_1}{2\delta} - \frac{f_0^{\mathrm{III}}\delta^2}{3!} - \frac{f_0^{\mathrm{V}}\delta^4}{5!} - \frac{f_0^{\mathrm{VII}}\delta^6}{7!} - \frac{f_0^{\mathrm{IX}}\delta^8}{9!} - \ldots \qquad (10)$$

Where, for the sake of notational compactness, we write derivatives in Newton-like notation (but with Roman superscripts rather than dots or apostrophes), i.e. we abbreviate the derivatives of $f(x)$ at $x = x_0$ as $f_0^{\mathrm{I}} = \mathrm{d}f(x)/\mathrm{d}x|_{x=x_0}$, $f_0^{\mathrm{II}} = \mathrm{d}^2f(x)/\mathrm{d}x^2|_{x=x_0}$, $f_0^{\mathrm{III}} = \mathrm{d}^3f(x)/\mathrm{d}x^3|_{x=x_0}$, etc. and use the abbreviation f_n for the value of the function $f(x)$ at $x = x_0 + n\delta$. As it turns out, we will usually select small enough values of δ/x to make all but the first higher-order derivatives negligible, in which case (10) can be truncated to

$$f_0^{\mathrm{I}} = \frac{-f_{-1}+f_1}{2\delta} - \frac{f_0^{\mathrm{III}}\delta^2}{3!}, \qquad (11)$$

while the equivalent expression for (6) reads

$$f_0^{\mathrm{I}} = \frac{f_{-2} - 8f_{-1} + 8f_1 - f_2}{12\delta} + \frac{4f_0^{\mathrm{V}}\delta^4}{5!}. \qquad (12)$$

In general, the j-point central difference formula for f_0^{I} will have a systematic error of $-(b_j\delta^{j-1}f_0^{\mathrm{J}})$, where $j = 2n_{\max} + 1$ denotes the number of equidistant data needed, of which only the central sample f_0 at $\delta = 0$ will not be used directly in the expressions given in table 1. The superscript J on f, acting as an adjunct Roman capital, denotes the jth derivative (rather than power) of $f(x)$. The quantity b_j is the absolute value of the coefficient of $f_0^{\mathrm{J}}\delta^{j-1}$ of the leading error term: $b_3 = |-1/3!| = 1/6$, $b_5 = |+4/5!| = 1/30$, $b_7 = |-36/7!| = 1/140$, etc.

Below we will find it useful to consider the *magnitude* (i.e. absolute value) of the *relative* errors (for a value a, with an error Δ, the relative error is Δ/a) which, by extension of (11) and (12), can be written as

$$E_{\mathrm{syst}} = b_j\delta^{j-1}\left|\frac{f_0^{\mathrm{J}}}{f_0^{\mathrm{I}}}\right|, \qquad (13)$$

assuming that $f_0^{\mathrm{I}} \neq 0$. (When $f_0^{\mathrm{I}} = 0$, the definition of E must of course use the absolute error, and then read $E_{\mathrm{syst}} = b_j\delta^{j-1}\,|f_0^{\mathrm{J}}|$ instead.). Some numerical values of $1/b_j$ are listed in table 2.

Equation (13) shows the (dominant term of the) error avoided in the mathematical definition (1) by making δ go to zero. In numerical analysis, however, that limit cannot be reached, because cancellation errors will usually take over before δ can become sufficiently small to make E_{syst} less than,

Table 2. Approximate numerical values of the coefficients b_j, c_j, d_j and $\varepsilon^{1/j}$ for the compact central difference expressions of the first derivative f_0^{I} listed in table 1 for $\varepsilon = 2^{-52} \approx 2{\cdot}220446 \times 10^{-16}$. Note that c_j is fairly constant, at between 2 and 4 times ε, and that d_j approximately tracks $\varepsilon^{1/j}$.

j	$1/b_j$	c_j	d_j	$\varepsilon^{1/j}$
3	6	$4{\cdot}53246651836840 \times 10^{-17}$	$5{\cdot}14223539198791 \times 10^{-6}$	$6{\cdot}05545445239335 \times 10^{-6}$
5	30	$6{\cdot}09031888443656 \times 10^{-17}$	$8{\cdot}54949099450734 \times 10^{-4}$	$7{\cdot}40095979741405 \times 10^{-4}$
7	140	$6{\cdot}93498646345351 \times 10^{-17}$	$7{\cdot}70909150071808 \times 10^{-3}$	$5{\cdot}80466519194121 \times 10^{-3}$
9	630	$7{\cdot}48317772924943 \times 10^{-17}$	$2{\cdot}62377319790381 \times 10^{-2}$	$1{\cdot}82270162433768 \times 10^{-2}$
11	2772	$7{\cdot}87541807882321 \times 10^{-17}$	$5{\cdot}72923961016090 \times 10^{-2}$	$3{\cdot}77527951376390 \times 10^{-2}$
13	12012	$8{\cdot}17386474122853 \times 10^{-17}$	$9{\cdot}84684360844102 \times 10^{-2}$	$6{\cdot}25000000000000 \times 10^{-2}$
15	51480	$8{\cdot}41076437690960 \times 10^{-17}$	$1{\cdot}46562567567514 \times 10^{-1}$	$9{\cdot}04543273400236 \times 10^{-1}$
17	218790	$8{\cdot}60471943284807 \times 10^{-17}$	$1{\cdot}98734464180835 \times 10^{-1}$	$1{\cdot}20005835856849 \times 10^{-1}$

say, 10^{-15}, or a similar number reflecting the maximum number of decimals displayed or bits carried, such as ε defined in the next section.

3. The IEEE 754 specifications and Excel

In order to describe the effect of finite numberlength we will first consider the IEEE-754 specification that is common to many software packages for personal computers. In this convention[7] a double precision floating point number is represented in 64 *binary units* or bits, of which one is devoted to the sign of the number, the next 11 bits to its exponent (properly biased by adding 1023 so that it cannot become a negative number), and the final 52 bits to the magnitude of its mantissa. A decimal equivalent would be similar to scientific notation, as in $-5{\cdot}43 \times 10^{21}$, where the sign is $-$, the exponent 21, and the mantissa 5·43 or, more precisely, as in -543×10^{19}, i.e. with the decimal point shifted to make the mantissa 543 a non-negative integer.

In the IEEE-754 specification, the non-negative mantissa ignores leading zeros in order to maintain a constant *relative* precision. Since a binary number only contains zeros and ones, the leading non-zero bit in the mantissa of a non-zero number will always be 1. Consequently, this first bit can be (and therefore is) implied, effectively making 53 bits available for the absolute value of the mantissa, so that this mantissa can represent 2^{53} non-negative integers. Any bits beyond those that can be accommodated in the mantissa are often simply ignored, so that the binary number is effectively truncated to fit the available space (unless the central processing unit has extra registers, in which case it can properly round, thereby reducing the uncertainty by 2, i.e. effectively adding one more bit). The relative uncertainty ε of these numbers, also called ulp for unit of last place or unit of least precision, is therefore (absent extra registers and rounding) at most 1 in 2^{53}, i.e., $2^{-53} = 1{\cdot}110223 \times 10^{-16}$.

We now focus specifically on Excel, simply because it is by far the most ubiquitous numerical software, and therefore presumably the most often used as such. (Many more accurate numerical software packages exist, but none with as wide a distribution as Excel. When people fly predominantly by jet, it matters little for the world that propeller-driven aircraft consume less fuel per passenger-mile and leave fewer sunlight-reflecting contrails.) The following simple experiment will establish that Excel uses $\varepsilon = 2^{-52} = 2{\cdot}22044604925031\text{E-}16$.

In a spreadsheet cell, say C2, deposit the number 1. In cell B3 place a sufficiently small number, such as 1E – 16. In cell C3 deposit the instruction = C2 + \$B\$3, and copy this down to, e.g. cell C200. Make columns B, C, and D wide enough to display all numbers in them in scientific notation with 14 decimals. As a result, the number 1·0000000000 0000 will show in all cells of the array C2 : C200.

Now gradually increase the value in cell B2, and observe what happens in column C. Nothing changes with, say, 1·11E-16, but when the value in B2 is increased to 1·111E-16 we find 1·000 00000000001 in cells C25 : C69, 1·00000000000002 in cells C70 : C114, 1·00000000000003 in C115 : C159, and 1·00000000000004 in the remainder of our column C2 : C200.

In order to understand what happens we deposit the instruction = (C2–1) in cell D2, and copy this down to D200. In this way we can see beyond the 15-decimal display limit of the spreadsheet to inspect the underlying numbers. We see 2·2204 4604925031E-16 in cell D3, double of that in cell D4, and so on down the column. The first transition in the numbers displayed in column C occurs when the value 4·88498130835069E-15 in D24 goes to 5·10702591327572E-15 in D25, i.e. when the number shown in column C is properly rounded to 1·00000000000001E-14. Thereafter the display changes about every 45 cells: $10^{-14}/2^{-52} \approx 45$ where 10^{-14} is the ulp *of the display*. Increasing the value of ε in B3 has no further effect in columns C and D until it goes from 3·33E–16 to 3·331E-16, at which point all values shown in column D double, the value 1·00000000000001 appears in cells C14 : C35, the value 1·00000000000002 in C36 : C58, etc. All the above observations point to $\varepsilon = 2^{-52} =$ 2·22044604925031E-16.

A small detail: when the brackets in the instructions in column D are deleted, the first few numbers in column D show zeroes. This is the consequence of a 'compensation' cryptically described in Microsoft kb 78113 as correcting for round-off errors in additions or subtractions that result in near-zero results when zero would be expected.[8] Specifically, this compensation in the displayed result of $z = x \pm y$ kicks in when both x and y are non-zero, the binary exponent of $z + 50$ is smaller than or equal to that of x, and the addition or subtraction is the last operation involved. The brackets bypass this compensation by counting as an (empty) last calculation step. Kahan[9] had already noticed this behaviour.

4. Cancellation errors

In most numerical software systems, the dominant errors in evaluating differences between near-equal numbers are those that result from the finite number-length used. Since our concern here is the effect of such relative uncertainty on numerical differentiation, we will use the term *cancellation* errors in order to distinguish them from the *systematic* errors introduced by truncating the Taylor expansions.

We now consider the arguments of the functions $-f_{-1}$ and f_1 in (11) under conditions where the dominant error is the cancellation error in the relatively small difference between f_{-1} and f_1. Because of the presence of a separate sign bit, the mantissa always represents a non-negative integer. Software that lops off all bits beyond the effectively available 53 will result in relative errors that are randomly distributed between 0 and ε. Such errors are best described by a *uniform distribution* $\varepsilon \times U(0, 1)$ or, equivalently, by a bias term $\varepsilon/2$ plus a noise term $\varepsilon \times U(-1/2, 1/2)$. The uniform distribution $U(a, b)$ has a mean of $(b + a)/2$ and a standard deviation of $(b - a)/\surd 12$. In terms of relative errors we therefore have a bias of $\varepsilon/2$ and a standard deviation of $\pm\, \varepsilon/\surd 12$. In Excel, rounding also leads to the uniform distribution $U(-½, ½)$, without the bias term, and with an ε-value that is twice as large. Below we will assume that the bias term is present, in order to illustrate that it is immaterial for *central* differencing, where it cancels, so that the above two models give identical predictions as long as the appropriate value of ε is used.

A detailed analysis of cancellation errors requires a specific protocol. Here we will discuss the simple procedure followed in our Excel macro Deriv1 as it implements the formula for $j = 3$ in table 1. (The algorithms for larger j-values follow a similar logic, and merely involve more terms, i.e. more manipulations and different constants, but no new concepts.) Before Deriv1 can be used, make sure that the spreadsheet contains a cell holding a formula that computes the function $f(x_0)$, where the value of the parameter x_0 is stored in a second cell. For such an arrangement we will consider the following algorithm: (i) the macro reads the value of x_0 in the second cell, (ii) changes it to $x_0 + \delta$ by adding a constant δ (iii) reads and stores the resulting value of $f_1 = f(x_0 + \delta)$, (iv) then changes the value in the second cell to $x_0 - \delta$, (v) reads and stores this value as $f_{-1} = f(x_0 - \delta)$, and finally (vi) computes the first derivative $f^{\mathrm{I}}(x_0)$ as $[f(x_0 + \delta) - f(x_0 - \delta)]/2\delta$.

When the same calculation is made entirely in background, without a spreadsheet, we assume that similar steps are taken, and the intermediate results stored, which should yield equivalent results. In Excel, invasive sampling (see section 8.14.1 in ref. 6) and reconstructing the equation in background (see section 8.14.2) indeed yield identical answers.

The numerical computation therefore involves three distinct stages: we first generate the arguments $x + \delta$ of f_1 and $x - \delta$ of f_{-1}, then calculate the corresponding functions f_1 and f_{-1}, and finally determine their difference, $f_1 - f_{-1}$. To see how cancellation noise enters these computations in the first step, we replace the argument x_0 of these functions by $(x_0 + \delta)\ \ (1 + \varepsilon/2 \pm \varepsilon/\surd 12) \approx \mathrm{x}_0 + \delta + \varepsilon x_0/2 \pm \varepsilon x_0/\surd 12$ on the assumption that $x_0 >> \delta$. In the second stage we then compute, store, and subsequently read the functions $f_1 = f(x + \delta)$ and $f_{-1} = f(x - \delta)$ as

$$f_1 = f(x_0 + \delta + \varepsilon x_0/2 \pm \varepsilon x_0/\surd 12)$$
$$\approx f_0 \ + (\delta + \varepsilon x_0/2 \pm \varepsilon x_0/\surd 12) f_0^{\mathrm{I}} \qquad (14)$$

and

$$f_{-1} = f(x_0 - \delta + \varepsilon x_0/2 \pm \varepsilon x_0/\sqrt{12})$$
$$\approx f_0 + (-\delta + \varepsilon x_0/2 \pm \varepsilon x_0/\sqrt{12})f_0^{\mathrm{I}}, \quad (15)$$

where we approximate the function by the first two terms of its Taylor series, see (8) and (9), with f_0 and f_0^{I} denoting the mathematically correct (i.e. bias- and noise-free) values of the function and its first derivative at $x = x_0$, and where we use $\pm$ to indicate the *unknown*, *pseudo-random* sign of cancellation noise rather than as a *choice* between a *known* top or bottom sign. (This is also why, to avoid confusion, we wrote out (8) and (9) separately.) Finally, before they can be subtracted, these functions must be held or stored, at which point they will again be truncated. Inclusion of the corresponding errors will then lead to

$$f_1 \approx (1 + \varepsilon/2 \pm \varepsilon/\sqrt{12})f_0 + (\varepsilon x_0/2 \pm \varepsilon x_0/\sqrt{12}+ \delta)$$
$$(1 + \varepsilon/2 \pm \varepsilon/\sqrt{12})f_0^{\mathrm{I}} \approx (1 + \varepsilon/2 \pm \varepsilon/\sqrt{12})f_0$$
$$+ (\varepsilon x_0/2 \pm \varepsilon x_0/\sqrt{12} + \delta)f_0^{\mathrm{I}}, \quad (16)$$

and a corresponding expression for f_{-1}. Taking their difference and dividing by 2δ yields

$$f_{0\,\mathrm{calc}}^{\mathrm{I}} = (f_1 - f_{-1})/2\delta \approx f_0^{\mathrm{I}} \pm (\varepsilon\sqrt{12})\,|f_0|\sqrt{2}/(2\delta)$$
$$\pm (\varepsilon\sqrt{12})\,|x_0 f_0^{\mathrm{I}}|\sqrt{2}/(2\delta) = f_0^{\mathrm{I}}$$
$$\pm \varepsilon\,(|f_0| + |x_0 f_0^{\mathrm{I}}|)/(\delta\sqrt{24})$$
$$= 0{\cdot}2041241\varepsilon\,(|f_0|+|x_0 f_0^{\mathrm{I}}|)/\delta, \quad (17)$$

where the bias terms cancel, while the noise terms are added as standard deviations should, i.e. as the square root of the sum of their squares, so that their variances are added directly. We therefore end up with *two* distinct terms to describe cancellation noise, which originate from computing x and $f(x)$ respectively. Dependent on the nature of the function f_0 and on the value of x_0, one of the two error terms in $(|f_0|+|x_0 f_0^{\mathrm{I}}|)$ will often be dominant.

For $j = 5$, see (6), we likewise find

$$f_{0,\mathrm{calc}}^{\mathrm{I}} = f_0^{\mathrm{I}} \pm \frac{\left[\dfrac{(\varepsilon/\sqrt{12})(|f_0|+|x_0 f_0^{\mathrm{I}}|)}{\sqrt{1^2+8^2+8^2+1^2}}\right]}{12\delta}$$

$$= f_0^{\mathrm{I}} \pm 0{\cdot}2742836\varepsilon\frac{|f_0|+|x_0 f_0^{\mathrm{I}}|}{\delta}, \quad (18)$$

and, in general, $f_{0,\mathrm{calc}}^{\mathrm{I}} = f_0^{\mathrm{I}} \pm c_j(|f_0|+|x_0 f_0^{\mathrm{I}}|)/\delta$, so that

$$E_{\mathrm{canc}} = \frac{c_j(|f_0|+|x_0 f_0^{\mathrm{I}}|)}{\delta\,|f_0^{\mathrm{I}}|}, \quad (19)$$

where E again denotes an absolute value, and ε has been included in c_j. Some values of the coefficient c_j are listed in table 2, as calculated with $\varepsilon = 2^{-52}$ and the coefficients f_i in the equations of table 1, cf. (17) and (18).

At relatively large δ-values, the systematic error will dominate, in which case (13) describes a linear asymptote in a plot of pE_{syst} as a function of $p\delta$, with slope $j - 1$ and intercept $-\log(b_j|f_0^{\mathrm{J}}/f_0^{\mathrm{I}}|)$. Likewise, at much smaller δ-values, such a graph should show a linear asymptote with slope -1 and intercept $-\log[c_j\,(|f_0|+|x_0 f_0^{\mathrm{I}}|)/|f_0^{\mathrm{I}}|]$ according to (19).

5. Optimizing the step size δ

In general, we must consider both the systematic error caused by truncating the Taylor series, and the cancellation noise introduced by subtracting numbers of near-equal magnitude. Since the systematic error is unidirectional, while the cancellation error yields pseudo-random noise, we add these two independent types of error as the algebraic sum of their absolute errors, $E_{\mathrm{total}}|f_0^{\mathrm{I}}| \le E_{\mathrm{syst}}|f_0^{\mathrm{I}}| + E_{\mathrm{canc}}|f_0^{\mathrm{I}}|$, and then return to their relative errors,

$$E_{\mathrm{total}} \le E_{\mathrm{syst}} + E_{\mathrm{canc}}$$
$$= b_j\delta^{j-1}\frac{|f_0^{\mathrm{J}}|}{|f_0^{\mathrm{I}}|} + \frac{c_j}{\delta}\frac{|f_0|+|x_0 f_0^{\mathrm{I}}|}{|f_0^{\mathrm{I}}|}, \quad (20)$$

where we again use the absolute values to ensure that their logarithms and/or roots can always be computed, regardless of the signs of f_0, x_0, f_0^{I}, and f_0^{J}. After all, we are interested in reducing the absolute magnitude of the total error, which at $\delta = \delta_{\mathrm{opt}}$ follows from (20) as

$$\frac{\mathrm{d}E_{\mathrm{total}}}{\mathrm{d}\delta} = (j-1)\,b_j\delta_{\mathrm{opt}}^{j-2}\frac{|f_0^{\mathrm{J}}|}{|f_0^{\mathrm{I}}|}$$
$$-\frac{c_j}{\delta_{\mathrm{opt}}^2}\left(\frac{|f_0|+|x_0 f_0^{\mathrm{I}}|}{|f_0^{\mathrm{I}}|}\right) = 0, \quad (21)$$

so that

$$\delta_{\text{opt}} = \left(\frac{c_j}{(j-1)b_j}\right)^{1/j} \left(\frac{|f_0|+|x_0 f_0^{\text{I}}|}{|f_0^{\text{J}}|}\right)^{1/j}$$

$$= d_j \left(\frac{|f_0|+|x_0 f_0^{\text{I}}|}{|f_0^{\text{J}}|}\right)^{1/j}, \tag{22}$$

with

$$d_j = \left(\frac{c_j}{(j-1)b_j}\right)^{1/j}, \tag{23}$$

for which table 2 displays some values.

Finally, substituting (22) and (23) back into (20) yields

$$E_{\text{opt}} = \left(\frac{jc_j}{(j-1)d_j}\right)\left(\frac{|f_0|+|x_0 f_0^{\text{I}}|}{|f_0^{\text{I}}|}\right) \left(\frac{|f_0^{\text{J}}|}{|f_0|+|x_0 f_0^{\text{I}}|}\right)^{1/j}, \tag{24}$$

which shows that it is in principle possible to predict the accuracy of the resulting answer.

Incidentally, the two extrapolated asymptotes intersect at $\delta = (c_j/b_j)^{1/j}$, which differs slightly from d_j because the absolute magnitudes of the slopes of the two asymptotes are different.

6. A first test of the model

We can readily examine the applicability of the above relations with functions $f(x)$ that have known derivatives. Here we first take the simple power law $f(x) = x^{20}$, so that the required higher derivative f_0^{J} does not vanish, and is readily computed even for $J = 17$, as will occur when we test with $j = 15$. This is a non-trivial test, because a 20th order power law is a highly nonlinear function of x. Our preliminary tests will use expressions on the left-hand side of table 1 to compute values for f_0^{I} for a wide range of δ-values around their optimal value δ_{opt} where E_{total} is maximal. We will look specifically at the asymptotic behaviours, to test the validity of (13) and (19) semi-quantitatively.

Let the known, correct values of f_0^{I} be denoted by $f_{0,\,\text{ref}}^{\text{I}}$ so that we can calculate the logarithm of the absolute values of the resulting relative numerical errors, which we will display as a function of pδ, where p is the operator –log( ), as in pH = –log[$\mathbf{H}^+$].

The quantity pE indicates the number of *significant* decimals in the answer, e.g. pE = 6 indicates that the answer is good to ± 1 in 10^6 or has 6 significant decimals, with the sixth good to ± 1, while pE > 6·3 will assure that all six decimals are correct. The double-logarithmic representation of pE vs pδ is convenient in this case because both (13) and (19) are power laws in δ. A plot of pE vs. pδ will emphasize that we want to *maximize the accuracy* of the sought derivative. Alternatively we could plot log(E) vs. log(δ), as done in, e.g. figure 3.3 of ref. (10), which instead emphasizes the effort to *minimize the errors*. Both representations contain the very same information, and are fully equivalent.

Figures 1 through 5 were generated on an Excel spreadsheet with the auxiliary custom macro DerivScan, which is part of my freely downloadable, open-access MacroBundle.[11] It takes j, $F(x)$ and x_0 as its input, and generates a column of values of the derivative as a function of pδ in the region around $\delta = |x_0|\,\varepsilon^{1/j}$, which we will use as our first estimate of δ_{opt}. The spreadsheet can then be used to combine

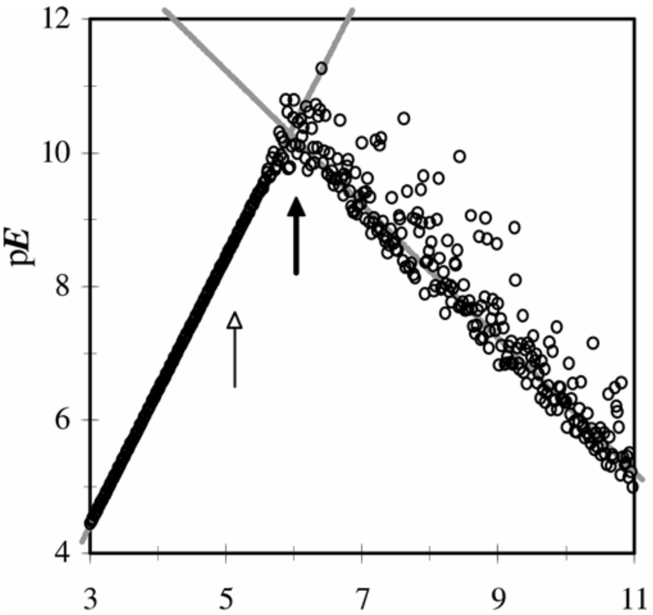

Figure 1. Plot of pE vs. pδ for $f(x) = x^{20}$, and its numerical analysis for $j = 3$ with $x_0 = 1{\cdot}234$ (open black circles). The gray straight line with slope +2 through the data is computed with (13), and the gray line with slope –1 with (19). Here and in figures 2 through 5, the location of the heavy vertical arrow identifies pδ_{opt} as calculated with (22) (here with the value p$\delta_{\text{opt}} = 6{\cdot}04$), while the light vertical arrow shows p$\delta = -\log|x_0| - (1/j)\log\varepsilon$ (here at p$\delta = 5{\cdot}13$).

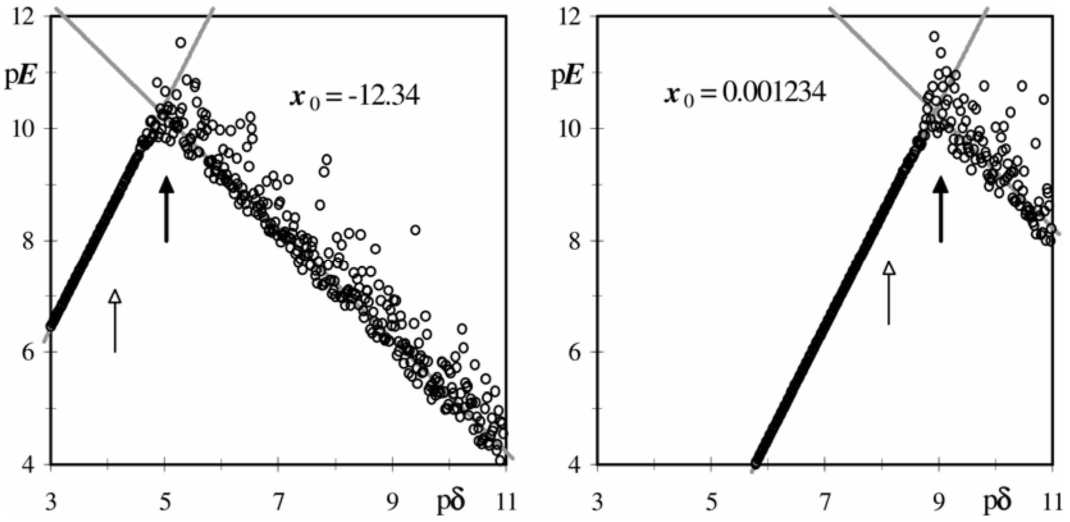

Figure 2. Plot of pE vs pδ for $f(x) = x^{20}$, and its numerical analysis for $j = 3$ with $x_0 = -12{\cdot}34$ (left panel) and $0{\cdot}001234$ (right panel). The gray straight lines with slope +2 are based on (13), those with slope −1 on (19). The heavy vertical arrows show (22) at p$\delta_{\text{opt}} = 5{\cdot}04$ for $x_0 = -12{\cdot}34$, and at p$\delta_{\text{opt}} = 9{\cdot}04$ for $x_0 = 0{\cdot}001234$; the light arrows display the corresponding values for pδ at p$\delta = 4{\cdot}13$ and p$\delta = 8{\cdot}13$ respectively. In figures 1 and 2, δ_{opt} is directly proportional to $|x_0\varepsilon^{1/j}|$, as expected from (22) for $f(x) = x^{20}$.

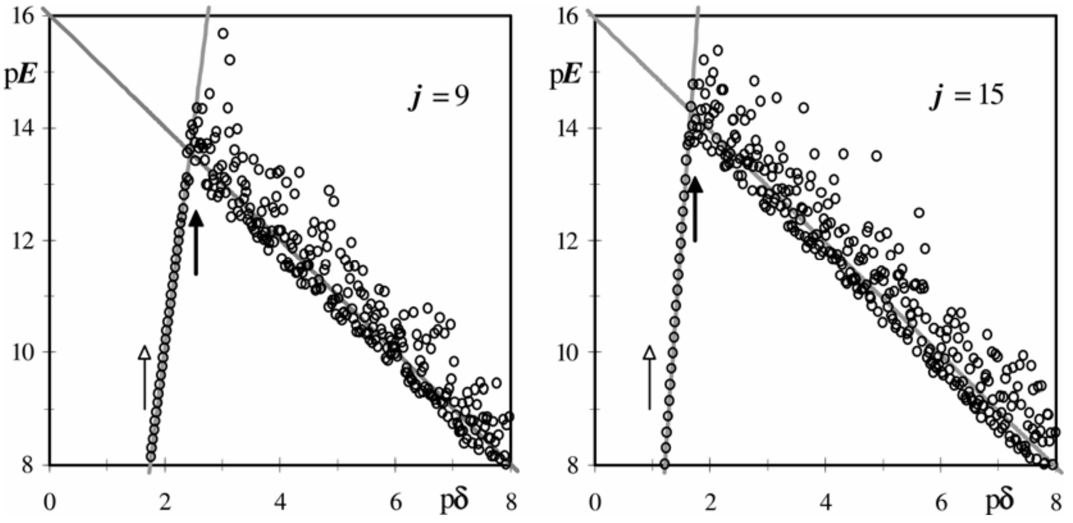

Figure 3. Plot of pE vs pδ for $f(x) = x^{20}$ with $x_0 = 1{\cdot}234$, for $j = 9$ (left panel) and 15 (right panel). The gray straight lines with slope +2 are based on (13), those with slope −1 on (19), and the vertical arrows are drawn at p$\delta_{\text{opt}} = 2{\cdot}54$ and p$\delta = 1{\cdot}65$ for $j = 9$, and at p$\delta_{\text{opt}} = 1{\cdot}74$ and p$\delta = 0{\cdot}95$ at $j = 15$. Note that the pE and pδ scales are both shifted by 3 units with respect to those in figures 1 and 2, and that δ_{opt} is again proportional to $|x_0\varepsilon^{1/j}|$.

the derivative with its reference value to yield the *pE*-values; instead, we could hard-code this into the macro which then, however, would become function-specific.

Figure 1 illustrates the result so obtained for the function $f(x) = x^{20}$ for $x_0 = 1{\cdot}234$, by varying the magnitude of δ over eight decades, from $\delta = 10^{-3}$ to $\delta = 10^{-11}$, so that pδ runs from 3 to 11. In figure 2 we show equivalent results for two different values of x_0, of different sign and magnitude, and in figure 3 for two larger values of j. These graphs are in quantitative agreement with the model: the asymptotes fit the data, and the model values of δ_{opt} and E_{opt} are indeed located near the maximum value of E.

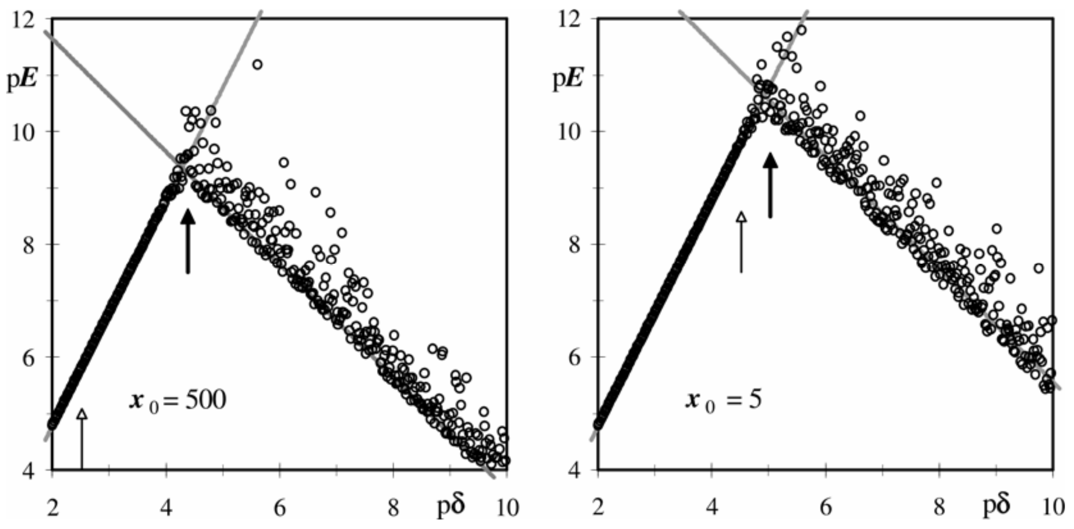

Figure 4. Plot of pE vs pδ for $f(x) = e^x$, and its numerical analysis for $j = 3$ with $x_0 = 500$ (left panel) and 5 (right panel). The gray straight lines are the model asymptotes, and the heavy vertical arrows are positioned at p$\delta_{\text{opt}} = 4{\cdot}39$ and $5{\cdot}03$ respectively, while the light arrows calculated at p$\delta = |x_0|\,\varepsilon^{1/j}$ at 2·52 and 4·52 differ by log(100) = 2.

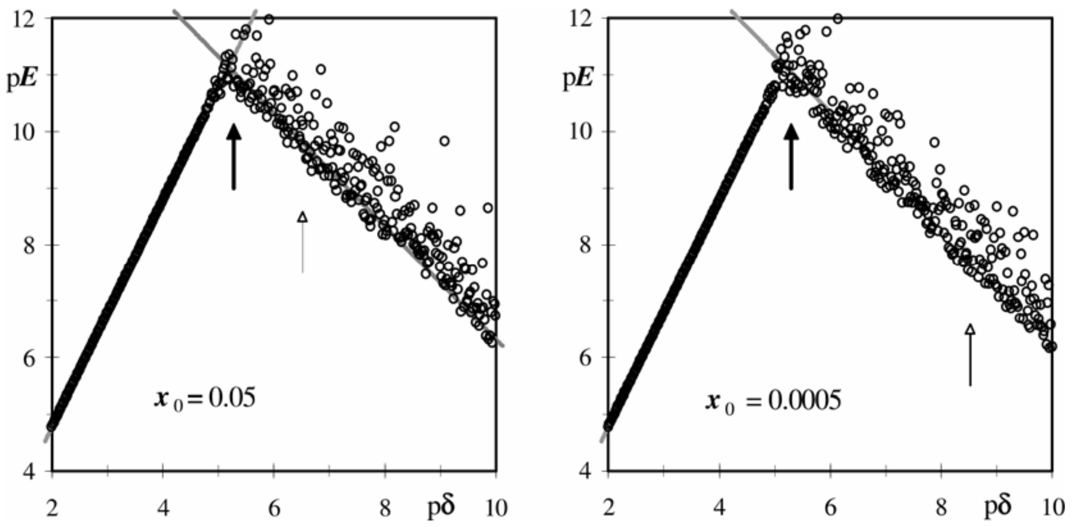

Figure 5. Plot of pE vs pδ for $f(x) = e^x$, and its numerical analysis for $j = 3$ with $x_0 = 0{\cdot}05$ (left panel) and 0·0005 (right panel). The heavy vertical arrows at p$\delta_{\text{opt}} = 5{\cdot}28$ and 5·29 are nearly identical, and roughly coincide with the observed maxima in pE, while the light arrows at p$\delta = 6{\cdot}52$ and 8·52 are clearly far off the mark.

In order to see the effect of the term $(|f_0|+|x_0f_0^{\text{I}}|)$ in (19) and (22) we use the exponential function $f(x) = e^x$, where $f_0^{\text{I}} = f_0$, so that merely changing x_0 from $x_0 >> 1$ to $x_0 << 1$ can serve as our test. In figures 4 and 5 we illustrate that our model prediction indeed applies.

For $x_0 >> 1$, both the location of the right-hand asymptote and the value of pδ_{opt} change with x_0 by the predicted amount, $(1/j)\log(x_0)$, whereas for $x_0 << 1$ we find that both are essentially constant. This supports the presence of two different stages at which cancellation errors can occur in central differencing, each with its own characteristic response.

The above examples illustrate the following: (i) The curves indeed exhibit two linear asymptotes, with a rather narrow transition region where pE is maximal. (ii) The asymptotes at large δ-values (and correspondingly small values of pδ) show very little noise, consistent with systematic errors, and are quantitatively described by (13). (iii) The asymptotes at small δ (i.e. at large pδ) are noisy, as compatible with cancellation noise, and are approximately described by (19). (iv) The cancellation noise in these graphs appears to be asymmetrical, but this is mostly an artifact of plotting pE, the logarithm of the absolute value of relative errors, as illustrated in the insert of figure 1. This is a well-known feature of noise after a logarithmic data transformation, further accentuated here by taking absolute values. (v) The value of δ_{opt} corresponding to the curve maximum is well-described by (22), and its shift with x_0 follows the predicted behaviour quantitatively, including a linear shift of pδ_{opt} with $(1/j)$ p$x_0 = -(1/j)\log|x_0|$ when $|f_0| << |x_0 f_0^{\text{I}}|$, and independence of x_0 when $|f_0| >> |x_0 f_0^{\text{I}}|$ for $f(x) = e^x$, see figures 4 and 5.

7. A first approximation

So far we have illustrated some preliminary tests of our model, by using known values of f_0^{I} and f_0^{J}. While this is useful to verify the applicability of the model used, it is the inverse of the practical problem at hand, which is to determine the first derivative f_0^{I} when it and the higher derivative f_0^{J} are not already known. As stated so far, the problem would appear to be circular: for an accurate determination of the first derivative we need to find δ_{opt}, which in turn requires estimates of both derivatives f_0^{I} and f_0^{J}, see (22). We therefore look for an approximation to break this circle.

The constants b_i, c_i, d_i, and ε in (22) are all dimensionless, while both δ and $[(|f_0| + |x_0 f_0^{\text{I}}|)/|f_0^{\text{J}}|]^{1/j}$ have the dimension of x. Moreover, d_i is approximately equal in magnitude to $\varepsilon^{1/j}$, see table 1. A crude, first-order approximation for δ is therefore $\delta \approx |x_0|\varepsilon^{1/j}$, see ref. 5. Anything beyond that will depend on the particular function f_0 involved. Unfortunately, as can readily be seen from the positions of the light vertical arrows in figures 1 through 5, the approximation $\delta \approx |x_0|\,\varepsilon^{1/j}$ is often way off the mark, whereas the heavy arrows defined by (22) for δ_{opt} trace the position of the maximum in pE quite well. On the other hand, as we will see below, even such a crude approximation usually suffices to get us started.

Consider figure 1, where (22) gives the apparently correct value for pδ of about 6, while $\delta \approx |x_0|\,\varepsilon^{1/j}$ yields a value of about 5. Because the slope of the left-most asymptote in figure 1 (i.e. for the systematic error) is 2, the resulting decrease in pE is also about 2, from approximately 10 to around 8. But keep that in perspective: it means that even $\delta \approx |x_0|\varepsilon^{1/j}$ gets the first derivative right to about 8 decimal places! That is certainly good enough for a first approximation.

At higher j-values, as the slope of the asymptote increases, the corresponding errors in pδ can likewise grow, see figure 3. But in that case the maximum in pE is quite high, near 13 or 14, so that we can tolerate a substantial error in pδ and still find a workable approximation for f_0^{I}. In figure 3b, e.g. the estimate based on $\delta \approx |x_0|\varepsilon^{1/j}$ yields a pδ-value that is about 0·69 too small, and its effect is amplified fourteen times to about 10 by the slope 14 of the term pE_{syst}. This brings the value of pE down from its lofty maximum of about 14 to a mere 4, but that still means that we can estimate f_0^{J} to about four significant decimals, certainly accurate enough for use in computing δ_{opt}, which only depends on the 1/14th power of f_0^{J}.

8. Higher-order derivatives

In an algorithm designed to determine f_0^{I}, its value is of course unknown, as is that of f_0^{J}. For an accurate determination of the first derivative we need to find δ_{opt}, which in turn requires estimates of both derivatives f_0^{I} and f_0^{J}, see (22). We therefore briefly extend the above treatment to such derivatives, confining our discussion to their smallest-footprint central difference formulas, where *footprint* defines the x-range over which the function must be sampled, such as from $x = x_0 - 2\delta$ to $x_0 + 2\delta$ when we use (6). Keeping the footprint small extends the range of applicability of central differencing when the function and/or its derivatives have discontinuities and singularities.

Proceeding as before, we find the equations shown in table 3, which include explicit expressions for their dominant systematic error terms. Note that the shortest central differencing formula for f_0^{III} involves $j = 5$ equidistant samples, while that for f_0^{V} requires $j = 7$, etc. where, as before, f_0 is included in counting j but is not used in the equations shown.

The leading relative errors in table 3 are

$$E_{\text{syst}} = -bb_{\text{J}}\delta^2 \frac{|f_0^{\text{J+II}}|}{|f_0^{\text{J}}|}, \tag{25}$$

where bb_{J} is the absolute value of the coefficient of $f_0^{\text{J+II}}\,\delta^2$ in the leading systematic error term of f_0^{J} in

Table 3. The lowest-order compact central difference formulas for the first through seventeenth odd derivatives of the function $f(x)$ at $x = x_0$, together with their leading systematic error terms.

f^{J} Formula	Leading term of systematic error
$f^{\mathrm{I}} = (-f_{-1} + f_1)/(2\delta)$	$-2\,f^{\mathrm{III}}\delta^2/12$
$f^{\mathrm{III}} = (-f_{-2} + 2f_{-1} - 2f_1 + f_2)/(2\delta^3)$	$-3\,f^{\mathrm{V}}\delta^2/12$
$f^{\mathrm{V}} = (-f_{-3} + 4f_{-2} - 5f_{-1} + 5f_1 - 4f_2 + f_3)/(2\delta^5)$	$-4\,f^{\mathrm{VII}}\,\delta^2/12$
$f^{\mathrm{VII}} = (-f_{-4} + 6f_{-3} - 14f_{-2} + 14f_{-1} - 14f_1 + 14f_2 - 6f_3 + f_4)/(2\delta^7)$	$-5f^{\mathrm{IX}}\delta^2/12$
$f^{\mathrm{IX}} = (-f_{-5} + 8f_{-4} - 27f_{-3} + 48f_{-2} - 42f_{-1} + 42f_1 - 48f_2 + 27f_3 - 8f_4 + f_5)/(2\delta^9)$	$-6\,f^{\mathrm{XI}}\delta^2/12$
$f^{\mathrm{XI}} = (-f_{-6} + 10f_{-5} - 44f_{-4} + 110f_{-3} - 165f_{-2} + 132f_{-1} - 132f_1 + 165f_2 - 110f_3 + 44f_4 - 10f_5 + f_6)/(2\delta^{11})$	$-7\,f^{\mathrm{XIII}}\delta^2/12$
$f^{\mathrm{XIII}} = (-f_{-7} + 12f_{-6} - 65f_{-5} + 208f_{-4} - 429f_{-3} + 572f_{-2} - 429f_{-1} + 429f_1 - 572f_2 + 429f_3 - 208f_4$ $+ 65f_5 - 12f_6 + f_7)/(2\delta^{13})$	$-8\,f^{\mathrm{XV}}\delta^2/12$
$f^{\mathrm{XV}} = (-f_{-8} + 14f_{-7} - 90f_{-6} + 350f_{-5} - 910f_{-4} + 1638f_{-3} - 2002f_{-2} + 1430f_{-1} - 1430f_1 + 2002f_2$ $- 1638f_3 + 910f_4 - 350f_5 + 90f_6 - 14f_7 + f_8)/(2\delta^{15})$	$-9\,f^{\mathrm{XVII}}\delta^2/12$
$f^{\mathrm{XVII}} = (-f_{-9} + 16f_{-8} - 119f_{-7} + 544f_{-6} - 1700f_{-5} + 3808f_{-4} - 6188f_{-3} + 7072f_{-2} - 4862f_{-1}$ $+ 4862f_1 - 7072f_2 + 6188f_3 - 3808f_4 + 1700f_5 - 544f_6 + 119f_7 - 16f_8 + f_9)/(2\delta^{17})$	$-10\,f^{\mathrm{XIX}}\delta^2/12$

table 3, i.e. $bb_{\mathrm{I}} = 1/6$, $bb_{\mathrm{III}} = 1/4$, $bb_{\mathrm{V}} = 1/3$, and so on. For this set of formulas, $j = J + 2$ and $bb_{\mathrm{J}} = (j + 1)/24 = (J + 3)/24$. Some numerical values of $1/bb_{\mathrm{J}}$ are listed in table 4. They are fairly constant since bb_{J} increases by a mere 1/24 for every unit increase in J. On the other hand, the slope of $\mathrm{p}E_{\mathrm{syst}}$ vs $\mathrm{p}\delta$ is 2, increasing significantly the higher the order of the derivative.

The cancellation noise of these higher-order derivatives will now be illustrated for f_0^{III} and f_0^{V}. Because the leading error term in the expression for f_0^{III} is in f^{V}, we need to include more terms of the Taylor series (8) and (9) in the equivalents of (14) and (15). Excluding the bias terms (which always cancel in central differencing regardless of whether we consider truncation or rounding) we therefore have

$$\begin{aligned} f_1 = (1 \pm \varepsilon/\sqrt{12})\ \{&f_0 + (\delta \pm \varepsilon x_0/\sqrt{12})f_0^{\mathrm{I}} \\ &+ (\delta \pm \varepsilon x_0/\sqrt{12})^2 f_0^{\mathrm{II}}/2! \\ &+ (\delta \pm \varepsilon x_0/\sqrt{12})^3 f_0^{\mathrm{III}}/3! + \ldots\}, \end{aligned} \tag{26}$$

or, deleting all terms that contain products of δ and ε or higher-order terms in ε,

$$\begin{aligned} f_1 \approx f_0 &\pm \varepsilon f_0/\sqrt{12} \pm \varepsilon x_0 f_0^{\mathrm{I}}/\sqrt{12} \\ &+ \delta f_0^{\mathrm{I}} + \delta^2 f_0^{\mathrm{II}}/2! + \delta^3 f_0^{\mathrm{III}}/3! + \ldots \end{aligned} \tag{27}$$

and likewise

$$\begin{aligned} f_{-1} \approx f_0 &\pm \varepsilon f_0/\sqrt{12} \pm \varepsilon x_0 f_0^{\mathrm{I}}/\sqrt{12} \\ &- \delta f_0^{\mathrm{I}} + \delta^2 f_0^{\mathrm{II}}/2! - \delta^3 f_0^{\mathrm{III}}/3! + \ldots \end{aligned} \tag{28}$$

so that

$$\begin{aligned} f_1 - f_{-1} \approx &\pm \varepsilon|f_0|/\sqrt{6} \pm \varepsilon|x_0 f_0^{\mathrm{I}}|/\sqrt{6} + 2\delta f_0^{\mathrm{I}} \\ &+ 2\delta^3 f_0^{\mathrm{III}}/3! + 2\delta^5 f_0^{\mathrm{V}}/5! + \ldots \end{aligned} \tag{29}$$

where we again add the variances of the cancellation noise terms. By including only the leading terms in systematic and cancellation errors we therefore find

$$f_0^{\mathrm{I}} \approx \frac{f_1 - f_{-1}}{2\delta} - \frac{\delta^2 f_0^{\mathrm{III}}}{3!} \pm \varepsilon \frac{|f_0| + |x_0 f_0^{\mathrm{I}}|}{2\delta\sqrt{6}}. \tag{30}$$

In a similar way we obtain

$$\begin{aligned} f_2 - f_{-2} \approx\ &\varepsilon|f_0|/\sqrt{6} \pm \varepsilon|x_0 f_0^{\mathrm{I}}|/\sqrt{6} + 4\delta f_0^{\mathrm{I}} \\ &+ 16\delta^3 f_0^{\mathrm{III}}/3! + 64\delta^5 f_0^{\mathrm{V}}/5! + \ldots \end{aligned} \tag{31}$$

$$\begin{aligned} &(f_2 - f_{-2}) - 2(f_1 - f_{-1}) \\ &\quad\approx \pm\varepsilon \frac{\sqrt{1^2 + 2^2}\,(|f_0| + |x_0 f_0^{\mathrm{I}}|)}{\sqrt{6}} \\ &\quad\quad + 2\delta^3 f_0^{\mathrm{III}} + \delta^5 f_0^{\mathrm{V}}/2 + \ldots \end{aligned} \tag{32}$$

so that

$$\begin{aligned} f^{\mathrm{III}} \approx &\frac{(f_2 - f_{-2}) - 2(f_1 - f_{-1})}{2\delta^3} \\ &- \frac{\delta^2 f_0^{\mathrm{V}}}{4} \pm \varepsilon \frac{|f_0| + |x_0 f_0^{\mathrm{I}}|}{2\delta^3\sqrt{6/5}}. \end{aligned} \tag{33}$$

For f_0^{V} we need (29) and (31) as well as

Table 4. Approximate numerical values of the coefficients bb_j, cc_j, and dd_j for the compact central difference expressions of the odd higher derivatives f_0^{J} listed in table 3 for $\varepsilon = 2^{-52} \approx 2{\cdot}220446 \times 10^{-16}$.

J	bb_j	cc_j	dd_j
3	3/12	$1{\cdot}68915054013559 \times 10^{-17}$	$5{\cdot}07854927301573 \times 10^{-4}$
5	4/12	$2{\cdot}44781168581343 \times 10^{-18}$	$3{\cdot}23047985786717 \times 10^{-3}$
7	5/12	$1{\cdot}86265625116248 \times 10^{-19}$	$8{\cdot}46645396177353 \times 10^{-3}$
9	6/12	$8{\cdot}70921574402188 \times 10^{-21}$	$1{\cdot}50093447046816 \times 10^{-2}$
11	7/12	$2{\cdot}75306104760735 \times 10^{-22}$	$2{\cdot}16957437368253 \times 10^{-2}$
13	8/12	$6{\cdot}27363817156281 \times 10^{-24}$	$2{\cdot}78508667460298 \times 10^{-2}$
15	9/12	$1{\cdot}07920867708565 \times 10^{-25}$	$3{\cdot}31895094524350 \times 10^{-2}$

$$f_3 - f_{-3} \approx \varepsilon|f_0|/\sqrt{6} \pm \varepsilon x_0 f_0^{\mathrm{I}}/\sqrt{6} + 6\delta f_0^{\mathrm{I}} + 54\,\delta^3 f_0^{\mathrm{III}}/3! + 486\delta^5 f_0^{\mathrm{V}}/5! + \ldots \quad (34)$$

which we combine with the expression for f_0^{V} in table 3 to

$$(f_3 - f_{-3}) - 4(f_2 - f_{-2}) + 5(f_1 - f_{-1}) \approx \pm\varepsilon \frac{\sqrt{1^2 + 4^2 + 5^2}\,(|f_0| + |x_0 f_0^{\mathrm{I}}|)}{\sqrt{6}} + 240\delta^5 f_0^{\mathrm{V}}/5! + 3360\delta^7 f_0^{\mathrm{V}}/7! + \ldots \quad (35)$$

hence

$$f_0^{\mathrm{V}} \approx [(f_3 - f_{-3}) - 4(f_2 - f_{-2}) + 5(f_1 - f_{-1})]/(2\delta^5) - \delta^2 f_0^{\mathrm{VII}}/3 \pm (\varepsilon\sqrt{7})(|f_0| + |x_0 f_0^{\mathrm{I}}|)/(2\delta^5), \quad (36)$$

and so on.

Note that the systematic and cancellation errors again behave quite differently. While the lower-order systematic error terms cancel, because they add and subtract *algebraically*, the terms of the cancellation noise add as their (always positive) *variances*, so that they do not cancel each other but, instead, always grow in magnitude. Consequently, the leading term of the cancellation noise remains that of the first order. As a result, in this group of lowest-footprint central difference higher-order derivatives the result becomes more accurate the higher the order or the derivative, compare figures 1 and 3. On the other hand, the footprint expands as we use higher-order derivatives, for two reasons: the number of samples j increases and, more importantly, their sample spacing δ_{opt} increases. Moreover, the pδ-region near the maximum in pE becomes narrower at higher j-values, so that it becomes more critical to get pδ right.

Table 4 lists some coefficients of the mathematical description of this group, where

$$E_{\mathrm{canc}} = \frac{cc_{\mathrm{J}}}{\delta^{\mathrm{J}}}\,\frac{|f_0| + |x_0 f_0^{\mathrm{I}}|}{|f_0^{\mathrm{J}}|}, \quad (37)$$

$$\delta_{\mathrm{opt}} = dd_{\mathrm{J}}\left(\frac{|f_0| + |x_0 f_0^{\mathrm{I}}|}{|f_0^{\mathrm{J+II}}|}\right)^{1/(\mathrm{J+II})}, \quad (38)$$

$$dd_{\mathrm{J}} = \left(\frac{cc_{\mathrm{J}}}{2bb_{\mathrm{J}}}\right)^{1/(\mathrm{J+II})}. \quad (39)$$

9. Implementation

With all component pieces in place we can now design the following algorithm, which is implemented in the Excel custom macro Deriv1 of the MacroBundle:[11] (i) read the number j of equidistant samples to be used, the function $f(x)$, and the value x_0 of x at which the derivative must be taken; (ii) estimate the equidistant x-spacing δ as $x_0\ \varepsilon^{1/j}$ where $\varepsilon = 2^{-52}$; (iii) sample the function $f(x)$ from $x = x_0 - (j+1)\delta/2$ to $x = x_0 + (j+1)\delta/2$ and use these samples to estimate both f_0^{I} and f_0^{J}; (iv) estimate δ_{opt} with (22), and finally (v) use δ_{opt} to compute an improved estimate of f_0^{I}.

This algorithm indeed yields quite impressive results for a number of relatively simple functions for which the correct answer is known, see table 5. In these examples, the pE-values are all larger than 9·5 for $j = 3$, >11 for $j = 5$, >12 for $j = 7$, and about 13 for $j = 9$; there is of course no guarantee that similarly good results are always obtained.

Especially at the higher pE-values, cancellation noise can contribute a considerable uncertainty in pE, which is why there appears to be no good reason

Table 5. The accuracy of the results for the first derivative $\mathrm{d}f(x)/\mathrm{d}x$ of a function $f(x)$ obtained with the Excel custom macro Deriv1 for a number of simple test functions for which the exact results are known. Shown here are the function $f(x)$, the value x_0 at which its first derivative $f^{\mathrm{I}}(x)$ is taken, the 'exact' value of the derivative as computed algebraically in Excel, and the pE-value of the result of the numerical calculation. (pE shows the number of significant decimals in the result,[6] using the analytical computation as its reference. It is both more compact and more informative than showing the actual results, which all agree in their first nine digits.) For the error function we have used our more accurate function[6,11] cErf, and for all Bessel functions Volpi's more accurate (but again double-precision) functions,[11] because the corresponding Excel functions are too inaccurate.

x_0	$f(x)$	$f^{\mathrm{I}}(x)$	pE at j =						
			3	**5**	**7**	**9**	**11**	**13**	**15**
1.234	$4-3x+2x^2-x^3$ = 1.46443110	$-3+4x-3x^2$ = -2.63226800	10.62	14.16	13.82	14.36	14.52	15.77	15.47
1.234	x^{20} = 67.0352439	$20x^{19}$ = 1086.47073	10.00	11.89	12.65	13.83	13.70	15.20	14.73
1.234	x^{-20} = 0.01491753	$-20x^{-21}$ = -0.24177514	9.87	11.69	12.62	13.05	13.31	13.74	13.65
1.234	$x^{1/20}$ = 1.01056850	$0.05x^{-19/20}$ = 0.04094686	10.47	12.20	13.06	13.85	12.84	13.07	14.31
1.234	$x^{-1/20}$ = 0.98954202	$-0.05x^{-21/20}$ = -0.04009490	10.35	11.40	12.59	13.10	12.95	13.60	13.28
1.234	ln(x) = 0.21026093	$1/x$ = 0.81037277	11.70	13.69	13.98	13.72	14.86	14.12	15.86
1.234	exp(x) = 3.43494186	exp(x) = 3.43494186	10.78	12.99	14.01	14.93	14.71	14.47	15.04
1.234	sin(x) = 0.94381821	cos(x) = 0.33046511	10.97	12.93	13.96	14.08	15.77	14.50	15.77
1.234	sinh(x) = 1.57190806	cosh(x) = 1.86303380	11.13	13.01	15.02	14.29	14.53	14.56	14.88
0.567	arcsin(x) = 0.60285925	$1/\surd(1-x^2)$ = 1.21400801	11.52	12.73	14.01	14.23	14.59		
1.234	arsinh(x) = 1.03755875	$1/\surd(1+x^2)$ = 0.62959660	10.86	12.98	13.45	14.01	15.45	14.50	14.16
1.234	tan(x) = 2.85602984	$1+\tan^2(x)$ = 9.15690644	10.38	13.51	12.78	13.79	13.33	14.37	13.61
1.234	tanh(x) = 0.84373566	$1-\tanh^2(x)$ = 0.28811013	10.69	12.36	13.18	13.32	14.04	14.14	13.81
1.234	arctan(x) = 0.88976245	$1/(1-x^2)$ = 0.39639188	11.47	13.05	13.45	13.77	14.95	14.90	14.51
0.567	artanh(x) = 0.64309026	$1/(1+x^2)$ = 1.47381546	11.76	12.89	13.02	13.89	14.00	14.02	
1.234	erf(x) = 0.91903942	$(2/\surd\pi)\exp(-x^2)$ = 0.24611072	10.25	12.10	12.62	13.36	13.07	13.52	13.78
1.234	$I_0(x)$ = 1.41848958	$I_1(x)$ = 0.742135021	11.39	12.58	13.20	13.85	15.13	14.38	14.32
1.234	$J_0(x)$ = 0.65404541	$-J_1(x)$ = -0.50677701	10.70	13.22	14.01	14.05	14.10	14.62	14.38
1.234	$K_0(x)$ = 0.30411714	$-K_1(x)$ = -0.41218265	10.45	12.75	12.86	13.60	14.83	14.29	14.33
1.234	$Y_0(x)$ = 0.24877388	$-Y_1(x)$ = 0.596023514	10.91	12.53	13.62	14.58	15.73	14.30	14.33

to use j-values larger than 9. In the interest of execution speed, the multiple-j output of Deriv1 (called by specifying j as 1) is therefore restricted to 3 (2) 9, although it only takes one line of the open-access code to change that to the full range $j = 3$ (2) 15. Comparison of these results with those in figures 9.2.8 in ref. 6 show that the present algorithm is superior to the use of $\delta = |x_0|\ \varepsilon^{1/j}$; only in a few cases did the latter occasionally produce higher pE-values, and never by more than can be accounted for by the vagaries of cancellation noise.

For most of the above evaluations we used the Excel-computed formulas for the first derivative as our reference. This does not always work: when we tried this with, e.g. the error function or the Bessel functions (which before 2007 were part of the Data Analysis Toolkit, and were fully incorporated in Excel 2007), it clearly showed that these are based on single-precision algorithms. In this case we have therefore used the more accurate double-precision cErf from my website, and the corresponding higher-accuracy (but still double-precision) Bessel functions in Volpi's free add-in Xnumbers.xla.[12]

In principle, (24) provides an estimate of the pE value of the result as $-\log E_{\mathrm{opt}}$ after correction for its imprecision due to cancellation noise. Still, it should be considered an *estimate*, and it will occasionally be overoptimistic, as is the case for the data in table

5 for tan(1·234) at higher j-values, because of the mere *proximity* of the singularity of tan(x) at $x = \pi/2 \approx 1{\cdot}57$; in this case, the footprint (also listed in the cell comment) may even reach across the singularity, thereby completely distorting the answer. In all but the most routine cases, the information provided in the cell comments should therefore be considered, because numerical differentiation cannot be put on automatic control.

10. Efforts to extend this approach

It is tempting to iterate, by using the value of δ_{opt} to recompute both f_0^{I} and f_0^{J}, and to use these for an improved value of δ_{opt} etc., but this does not appear to improve matters much, because we are now largely operating in the region of pseudo-random cancellation noise. In principle, the influence of this noise could be reduced by averaging, after repeating the calculation while varying δ over a narrow range of values around δ_{opt}. While averaging based on invasive sampling would be too slow, reconstructing the formula for $F(x)$ in VBA can speed up sampling by about two orders of magnitude, and can therefore allow substantial averaging. The hope was that, after such averaging, the pseudo-random cancellation noise would be small enough to allow Richardson extrapolation.[13,14] Such extrapolation was shown by Ridders[15] to work with δ-values much larger than δ_{opt}, in which case cancellation noise is negligible, see figures 1 through 5, but in that case the footprint is much increased.

Unfortunately, averaging does not appear to improve the accuracy much; this may well be because we are operating near the limit of Excel, and its lack of using subnormals sometimes shows. However, with Volpi's extended precision add-in Xnumbers.dll[11,12] we can bypass the above problem, and get full 15-decimal accuracy in Excel, and we have implemented this in the macro xDeriv1. In order to avoid Excel's accumulation noise, we had to forego the invasive sampling method used in Deriv1, and instead have used Excel's character string functions to reconstruct the spreadsheet formula for $F(x)$ in the macro; beyond that, there are no further complications. In fact, in xDeriv1 we would not really need the sophistication of the present algorithm, because even with $\delta = |x_0|\varepsilon^{1/j}$ and $j = 3$ we obtain pE-values in excess of 14 when using quadruple precision, except in those cases where Excel's functions are below par, see, e.g. sections 11·6 and 11·7 of ref 6.

The macro xDeriv1 uses the same approach as Deriv1, with j-values up to 15 and values for DigitsMax up to 200, yet even at $j = 15$ and DigitsMax = 200, xDeriv1 takes only a fraction of a second to execute on my computer with a 3·4 GHz Pentium-D processor. As can be seen in table 6, a comparison of the use of $\delta = |x_0|\varepsilon^{1/j}$ with δ_{opt} as given by (22) illustrates the superiority of the latter, especially at higher j-values.

A crude model for the maximum value p$E_{\max}$ of pE can be obtained by approximating the systematic error as p$E_{syst} \approx (j-1)$ pδ and the cancellation error as p$E_{\text{canc}} \approx$ DgtMax −1 pδ, see figure 9.2.5 in ref 6, so that their intersection yields p$\delta_{\text{opt}} \approx$ DgtMax/j and p$E_{\max} \approx$ DgtMax × $(j-1)/j$. The results in the rightmost column of table 3 all come to within 1·5 of this prediction, which disregards the presence of cancellation noise. This exercise is useful because the approximate expression for pδ_{opt} as DgtMax/j illustrates how the footprint of central differencing, $\pm(j-1)\,\delta_{\text{opt}}/2 \approx \pm 1/2(j-1) \times 10^{-\text{DgtMax}/j}$, increases sharply with j, from about $\pm 10^{-5}$ at $j = 3$ to about $\pm 0{\cdot}7$ at $j = 15$ for DgtMax = 15.

11. Discussion

As illustrated in figures 9.2.4 and 9.2.5 of ref 6, the presence of systematic errors and cancellation noise sets limits to the accuracy that can be achieved with central differencing in double-precision. Discounting the noise, those limits would be of the order of

Table 6. Results obtained for $f(x) = x^{20}$ at $x_0 = 1{\cdot}234$ with xDeriv1 for DigitsMax = 30, the Xnumbers default value, roughly corresponding with quadruple precision. Shown are two sets of results: one obtained by using $\delta = |x_0|\,\varepsilon^{1/j}$, the other with δ_{opt} as given by (22). The pE-values were also computed in Xnumbers, and displayed in the Immediate Window. When the results are instead written back onto the spreadsheet, they are all rounded to 15 decimals, and become indistinguishable.

	DgtMax = 30	
j	With δ pE	With δ_{opt} pE
3	18·24	19·00
5	20·51	23·04
7	20·57	24·43
9	19·98	25·40
11	19·19	25·90
13	18·39	26·40
15	17·71	26·66

11 significant decimals with $j = 3$, about 13 significant digits with $j = 5$, approximately 14 with $j = 7$, and 14·5 with $j = 9$, while the presence of noise tends to lower that estimate by about 1·5. The algorithm outlined above indeed succeeds in this respect, see table 5, by finding the relatively narrow range of δ-values over which the result can have such near-maximal accuracy. Quadruple precision would alleviate this problem, and could even confer about 15 significant digits to answers obtained with the simplest forms of forward and backward differencing, but at present such extended numberlength is only available in Excel with add-on software such as Leonardo Volpi's Xnumbers or John Beyers'XN.

There are few if any *input or output data* that require more than 15-decimal precision, but plenty of *data processing* procedures that do, not just differentiation but, much more importantly, linear algebra operations such as matrix inversion. With Xnumbers.dll, the user still can enter and extract (through message boxes, cell comments, and/or the Immediate Window) its higher-precision information, but its default input and output uses the standard double-precision spreadsheet format. In my opinion, this is a near-ideal solution. Excel could easily offer the same, at least up to quadruple precision, if VBA could access the extended-precision capabilities already available in many modern central processing units for personal computers, but Microsoft has not yet implemented this.

Because Xnumbers.dll runs as an Excel add-in, it can only operate in open-access software. Moreover, it cannot use any (often already available) dedicated hardware for quadruple precision, and therefore must take the more time-consuming route of software implementation. To make matters worse, it is often difficult to install Xnumbers.dll in Vista/Excel 2007, which appears to be much less welcoming of add-in software than previous versions of Excel; it should work if and when Microsoft has sorted out its registry problems. Until then, my advice is therefore to stay with Windows XP and Excel 2003 when spreadsheet accuracy is important enough to warrant use of Xnumbers.dll. The recent, quite substantial extension of Volpi's Xnumbers.xla by John Beyers to XN.xla and, for Vista/Excel 2007, XN.xlam can be used to overcome this handicap, see ref 12.

But forget what could have been, and undoubtedly will be in the near future, and back to the present. The macro Deriv1 has been tested preliminarily and found to work in Excel 2000, 2003, XP, and 2007, xDeriv1 likewise appears to work in Excel 2000, 2003, and XP, and corresponding macros Deriv2 and xDeriv2 for the second derivative are coming. Of course, there is no way to test general differentiation algorithms comprehensively, and there may well be cases where they do not work, e.g. when the initial estimate $\delta \approx |x_0|\varepsilon^{1/j}$ is too far off for convergence to δ_{opt}. Users are therefore advised to first try these macros with their particular functions $f(x)$ and their specific ranges of arguments x. As always, for any particular type of application, testing must be a precondition for trusting.

The algorithm described here is not Excel-specific, as illustrated by the similar results shown in table 6, which were obtained with Xnumbers, performing the entire computation in non-Excel software. The spreadsheet is used here merely as a convenient (and, to this author, familiar) vehicle for data input, output, and plotting. The open-access VBA code of Deriv1 can be translated readily into other computer languages, and readers are encouraged to do so. My Excel macros can be freely downloaded, used, modified, and shared, and are only copyrighted and provided under the GNU General Public License to protect them from unauthorized commercial exploitation.

The derivation has emphasized keeping the footprint of central differencing as small as possible, because this footprint defines how closely we can approach a discontinuity or singularity without having to switch to (for the same j-values generally less accurate) lateral differencing. For the determination of f^{I} the footprint is $\pm(j-1)\delta/2$, while that for f^{J} is $\pm(j+1)\delta/2$; for the final value the footprint is $\pm(j-1)\delta_{\text{opt}}/2$. The larger of the latter two quantities, $(j+1)\delta/2$ or $(j-1)\delta_{\text{opt}}/2$, is displayed by Deriv1 in the cell comment as the magnitude of the footprint. It specifies the minimum distance between x_0 and the x-value of the nearest discontinuity; for a singularity, a larger distance is advisable.

It is often stated that good results in, e.g. optimization routines, require the use of *analytical* derivatives. With the present approach, that argument may no longer be so compelling.

Acknowledgements

Helpful comments by Linde Wittmeyer-Koch of the University of Linköping and Mohammad Tajdari of

Bowdoin College, and by Andrew Becker of Microsoft on its implementation of the IEEE-754 protocol, are gratefully acknowledged.

References

1. Levenberg K 1944 *Quart. Appl. Math.* **2** 164
2. Marquardt D W 1963 *SIAM J. Appl. Math.* **11** 431
3. Abadie J 1978 In *Design and implementation of optimization software* (ed.) H J Greenberg (Sijthoff & Noordhoff) pp 335–363
4. Lasdon L S and Waren A D 1978 In *Design and implementation of optimization software* (ed.) H J Greenberg (Sijthoff & Noordhoff) pp 363–397
5. Press W H, Teukolsky S A, Vetterling W T and Flannery B P 2007 *Numerical recipes* (Cambridge University Press) 3rd edn, pp 229–232
6. de Levie R 2008 *Advanced Excel for scientific data analysis* (Oxford University Press) 2nd edn
7. IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Std 754-1985, reprinted in SIGPLAN Notices 22#2, 9–25 (1987), http://portalparts.acm.org/30000/24686/fm/frontmatter.pdf, accessed December 2008
8. Floating-point arithmetic may give inaccurate results in Excel, Microsoft's Knowledge Base 78113 at http://support.microsoft.com/kb/78113/en-us, last reviewed 26 February 2007, accessed December 2008
9. Kahan W How futile are mindless assessments of roundoff in floating-point computation?, http://www.cs.berkeley.edu/~wkahan/Mindless.pdf, 11 January 2006, accessed December 2008
10. Rice J R 1983 *Numerical methods, software, and analysis* (McGraw-Hill) p. 186
11. The MacroBundle and xMacroBundle are freely downloadable from my website http://www.bowdoin.edu/~rdelevie/excellaneous. Volpi's software packages BigMatrix and Xnumbers can also be downloaded from this website, as well as from his own website, ref. 12
12. Leonardo Volpi's website is found at http://digilander.libero.it/foxes/Software Download.htm. A substantial extension of Xnumbers.xla called XN.xla (or, for Excel 2007, XN.xlam) has recently been released by John Beyers, and is downloadable from steve@thetropicalevents.com
13. Richardson L F 1910 *Philos. Trans. Roy. Soc. London* **A210** 307
14. Richardson L F 1927 *Philos. Trans. Roy. Soc. London* **A226** 299
15. Ridders C J F 1982 *Adv. Engin. Software* **4** 75