BRYN MAWR COLLEGE

Department of Physics

Computational Module 4 – Differentiation & Interpolation

Prerequisite modules: Module 0; Module 1
Estimated completion time: 3-6 hours
Learning objectives: understand the various numerical approximations to the derivative, become familiar with the approach to determining the optimal step size for such approximations, understand linear interpolation.

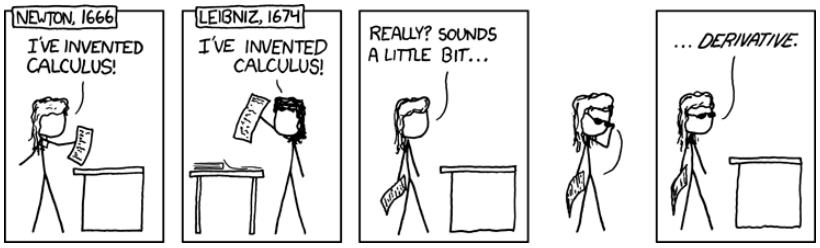

Image credit: xkcd.com

Numerical differentiation and its close relative, interpolation, are frequently-used tools in the analysis of scientific data. This module will investigate numerical differentiation and provide a short introduction to interpolation.[1]

4.1 Definition of the Derivative

Recall that the purpose of the derivative is to determine the rate of change of a function at a point: we do this by computing the rate of change in a small region containing the point, and then take the limit as the size of that region shrinks to zero, thus "collapsing" down to the point. The numerical approach to differentiation starts with the definition of the derivative:

[1] This Module adapted from *Computational Physics* by Mark Newman

$$\frac{df(x)}{dx} \equiv \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}. \tag{1}$$

Because a computer cannot actually take the limit as the size of the region, h, shrinks all the way to zero, an *approximate* derivative instead is computed from the same expression without taking the limit,

$$\frac{df(x)}{dx} \simeq \frac{f(x+h) - f(x)}{h}, \tag{2}$$

where h is very small, but not zero. This expression is known as the ***forward difference***, because it involves the function at points x and $x + h$. One can also define a ***backward difference*** as

$$\frac{df(x)}{dx} \simeq \frac{f(x) - f(x-h)}{h}, \tag{3}$$

which involves the function at x and $x - h$. These two methods of computing the derivative usually give roughly the same answer. They are instances of a category of techniques known as ***finite difference methods*** (introduced in Module 3), involving functions or sets of values evaluated at intervals that are very small but nonzero, as with h here. The critical issue involved in using either expression for a numerical computation is what value to use for h; that is, how closely spaced the points should be at which the function is computed. This issue will be considered below.

4.2 From Differences to the Derivative

There will be some error in the derivative calculation due to the fact that a computer cannot actually deal with an h that shrinks down to zero, and also because of inevitable rounding errors in numerical calculations. To see the effect of these potential errors, consider a Taylor expansion[2] of the function $f(x)$ in the small parameter h, giving

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2!}f''(x) + \frac{h^3}{3!}f'''(x) + \ldots, \tag{4}$$

where primes denote differentiation with respect to x. Briefly, this expression represents the value of a function at a specific point, $x+h$, in terms of its value at a nearby point x, using a power series in increasing powers of h. Note that going toward the right in the series the terms generally are of decreasing significance, since h is small and successive terms involve h to higher powers. Each successive term also involves one higher-order derivative (evaluated at x) than the preceding term. The second figure (an animated one) at the Wikipedia page on the Taylor series may help give a sense of how it approximates a function.

[2] If unfamiliar with this very powerful technique, see e.g., Chapter 1, Section 12 of *Mathematical Methods in the Physical Sciences* by Mary Boas.

Rewriting Eq. (4) to resemble the forward difference (by isolating the f' term), we get

$$\frac{df(x)}{dx} = \frac{f(x+h) - f(x)}{h} - \frac{h}{2!}f''(x) - \frac{h^2}{3!}f'''(x) - \ldots \tag{5}$$

We see that the difference between the exact derivative on the left, and the approximate one given by the forward difference term on the right, is a series of terms in increasing powers of h. Clearly, the difference between the exact and forward-difference terms will shrink as h does. Furthermore, since h will be made very small, we can focus on just the term containing h to the first power, as the others will be small relative to it. (We're assuming that $f'' \neq 0$.) Thus, the error in approximating the derivative by the forward difference term and ignoring higher-order terms is roughly $\frac{1}{2}h|f''(x)|$. Since this is "of order" h (that is, its magnitude is comparable to h), it often is written as $\mathrm{O}(h)$. (Note that this is different from the O of the big-O notation introduced in Module 2!)

Now, another issue that can arise is that because of numerical rounding errors, discussed in Module 2, subtracting one number from another can lead to a difference between them with a large fractional error. For example, suppose $a = 1000000000000001.2345$ and $b = 100000000000000$, so then $a - b = 1.2345$. However, in Python a would be represented in the computer as 1000000000000001.2 (for a computer that provides 16 significant figures), so $a - b$ would be computed as 1.2. The fractional error between the computed difference and the actual difference then would be

$$\text{Fractional error} = \frac{|\text{Difference}|}{\text{Average}} = \frac{|1.2345 - 1.2|}{\frac{1}{2}(1.2345 + 1.2)} \approx 0.03, \tag{6}$$

or 3%, which is much greater than the fractional errors in a and b.

Breakpoint 1: What is the fractional error in representing $a = 1000000000000001.2345$ as the number 1000000000000001.2?

We conclude that the fractional error in the difference between two nearly equal numbers can be large compared to the errors in those numbers. In Eq. (5), this may happen in the first term on the right when h becomes very small, as we want it to be. Denote by ϵn the fractional error associated with a number n stored in Python (with $\epsilon \approx 10^{-16}$, typically); then the error in the forward difference calculation due to the rounding issue is, at worst, $2\epsilon f(x)/h$. (This conclusion assumes that the rounding errors in the two terms in the numerator of the first term on the right, $\epsilon f(x+h) \simeq \epsilon f(x)$ and $\epsilon f(x)$, go "in the same direction" rather than canceling out).

The error $\frac{1}{2}h|f''(x)|$ due to omitting terms in the Taylor expansion of Eq. (4), and the error $2\epsilon f(x)/h$ due to numerical rounding, combine to give a worst-case total error $E = \frac{1}{2}h|f''(x)| + 2\epsilon f(x)/h$. The value of h that minimizes this total error can be found by differentiating with respect to h and setting the result to zero; one obtains

$$h_{opt} = \sqrt{4\epsilon |f(x)/f''(x)|}. \qquad \text{Forward difference optimal step size} \tag{7}$$

For that minimizing value of h, the corresponding minimum error is

$$E_{min} = \sqrt{4\epsilon |f(x) f''(x)|}. \qquad \text{Forward difference minimum error} \tag{8}$$

Breakpoint 2: Derive these two results.

Inspecting the last two results, we see that if $f(x)$ and $f''(x)$ are roughly of order 1, then h should be chosen to be $h \approx \sqrt{\epsilon} \approx 10^{-8}$, and then the corresponding error E also is of order 10^{-8}. That is, the derivative calculated by the forward difference method will be accurate to only roughly 8 significant figures, even though the values of the function being differentiated are stored to 16 significant figures. The same analysis applies to the backward difference approximation.

While both the forward and backward differences have theoretical errors of leading order h, a better difference can be constructed from a kind of combination of the two, namely the ***central difference***, defined as

$$\frac{df(x)}{dx} \simeq \frac{f(x + h/2) - f(x - h/2)}{h}. \tag{9}$$

This difference compares the function at two points symmetrically located around x at $x \pm h/2$. Geometrically, we might expect this difference to be more accurate than the forward or backward differences, and in fact it is. By writing out the Taylor series for $f(x \pm h/2)$, one sees that the terms of order h cancel, leaving those of order h^2 (which multiply $f'''(x)$) as the largest error terms. One can again compute the total error in this case, obtaining $E = \frac{1}{24} h^2 |f'''(x)| + 2\epsilon f(x)/h$; minimizing it, the result is that the optimal step size and corresponding minimum error are

$$h_{opt} = \left(24\epsilon \left|f(x)/f'''(x)\right|\right)^{1/3} \approx \epsilon^{1/3} \approx 10^{-5}, \qquad \text{Central difference optimal step size} \tag{10}$$

$$E_{min} = \left(\frac{9}{8}\epsilon^2 |f(x)|^2 \left|f'''(x)\right|\right)^{1/3}. \qquad \text{Central difference minimum error} \tag{11}$$

For the same numerical values used before, this error is around 10^{-10}, roughly 100 times smaller than for the forward or backward differences. Note that this smaller error is obtained with a *larger* value of h than before! (Thus, the computation can be done with fewer steps, and therefore faster.)

A complication arises when the central difference method is applied to a *discrete* function whose value is computed at regularly-spaced intervals, rather than to a *continuous* function (which doesn't exist in the numerical world): in that case, if h is the distance between neighboring points, then we don't have values at the halfway points $x \pm h/2$. The central difference method still can be used,

but only at the points $x \pm h$, which amounts to doubling h, leading to the expression

$$\frac{df(x)}{dx} \simeq \frac{f(x+h) - f(x-h)}{2h}. \tag{12}$$

A little analysis shows that in this case the central difference method will lead to a smaller error than the forward difference method if $h^2|f'''(x)| < h|f''(x)|$, or

$$h < \left|\frac{f''(x)}{f'''(x)}\right|. \qquad \text{Condition for central difference to beat forward difference} \tag{13}$$

4.3 Higher-order Approximations to the First Derivative

In our treatment of the first derivative of a function, we essentially connected neighboring points (in the case of the forward and backward differences), or points separated by an intermediate point (in the case of the central difference), with imaginary straight line segments whose slopes represented the local derivatives of the function – we were making a piecewise-*linear* (degree 1) approximation to the function being differentiated. By using polynomials of higher degree to interpolate between points, it is possible to get better approximations to the true derivative of the function.

The next polynomial to try in the modeling process is a quadratic (degree 2), so we imagine fitting the positions of three neighboring points in the function using the equation $y = ax^2 + bx + c$. The central point of the three is the one at which we'll end up finding the derivative; once we know how to do this at one point, we can extend it to every point in the function. If the central point is taken to be at $x = 0$, and the neighbors at $x = -h$ and $x = +h$, then by requiring the quadratic curve to pass through those three points we get the three equations

$$f(-h) = ah^2 - bh + c, \qquad f(0) = c, \qquad f(h) = ah^2 + bh + c. \tag{14}$$

We could now solve these equations for the three parameters, a, b, and c, to get the quadratic curve that fits the function at $x = 0$. However, all we really want is the derivative of that quadratic curve at $x = 0$, which is represented by the parameter b.

Breakpoint 3: Prove this claim that b is the derivative of the quadratic curve $y = ax^2 + bx + c$ at $x = 0$.

We immediately get b by subtracting the first of Eqs. (14) from the third, finding:

$$\frac{df}{dx} \simeq b = \frac{f(h) - f(-h)}{2h}. \tag{15}$$

Unfortunately, this is not a new formula – it's just the central difference expression with $h/2$

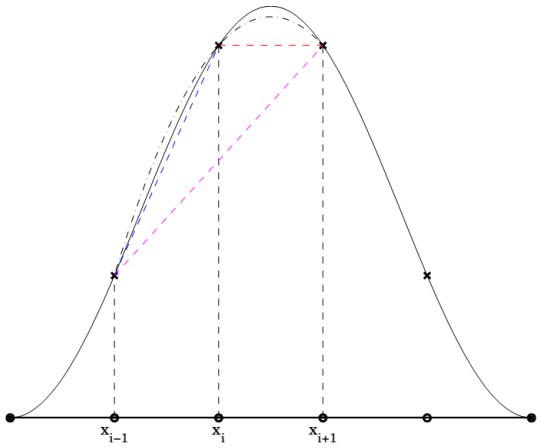

Figure 1: Comparison of first derivative methods. The slopes of the dark blue, red and magenta dashed lines represent the backward-, forward-, and central-difference derivative approximations, respectively, at the point x_i. The dash-dot curve shows the quadratic approximation; its slope at x_i gives the second-order derivative approximation. That slope looks the same as the slope of the central-difference line, supporting the claim made below Eq. (15).

replaced by h, Eq. (12)! It will be no better than the original Eq. (9), and is likely to be less accurate, since it samples points farther away from the central point of interest. See Figure 1 for a comparison of different derivative methods.[3]

It is possible to get better derivative results by using yet higher-order polynomial approximations (i.e., polynomials of higher degree). The process echoes the one above: require that the polynomial pass through neighboring points to the central one (at $x = 0$), and solve for the parameter in the linear term, which represents the derivative at $x = 0$. In the case of a cubic polynomial ($y = ax^3+bx^2+cx+d$), four function sample points are needed (because of the unknown parameters a, b, c, d), at intervals of h symmetric about $x = 0$; namely, at $-\frac{3}{2}h$, $-\frac{1}{2}h$, $\frac{1}{2}h$, $\frac{3}{2}h$. These fall halfway between points at which the function is determined, as was the case for the linear approximation of the central difference method. For a quartic polynomial (with highest power x^4), five sample points are needed, at $-2h$, h, 0, h, $2h$. These correspond to actual function points, as in the quadratic case. Thus, we note a general pattern: for odd-order approximations (linear, cubic, etc.), the sample points are halfway between points at which the function is computed; for even-order approximations (quadratic, quartic, etc.), the sample points are at computed function points.

Following the earlier procedure, for the case of higher-order approximations one finds the coefficients for the various terms in the derivative shown in Table 1. In the first column of the table, the order of the polynomials is labeled "Degree": the linear fit has a degree of 1, the quadratic fit has a degree of 2, etc. The last column indicates the approximate scale of the error: note that odd- and even-order solutions come in pairs with roughly equal errors. Also note that the error decreases

[3]Figure from lecture notes on Scientific Programming for Atmospheric Science by M. Iskandarani, Rosenstiel School of Marine Atmospheric Science, U. of Miami. See item 4, Finite Difference Approximations, under Numerical Methods, at http://www.rsmas.miami.edu/personal/miskandarani/Courses/MSC321/.

Degree	$f(-2h)$	$f(-\frac{3}{2}h)$	$f(-h)$	$f(-\frac{1}{2}h)$	$f(0)$	$f(\frac{1}{2}h)$	$f(h)$	$f(\frac{3}{2}h)$	$f(2h)$	Error
1				-1		1				$O(h^2)$
2			$-1/2$				1/2			$O(h^2)$
3		1/24		$-27/24$		27/24		$-1/24$		$O(h^4)$
4	1/12		$-2/3$				2/3		$-1/12$	$O(h^4)$

Table 1: Coefficients of terms in higher-order approximations of the first derivative

significantly with higher-order approximations. The middle columns give the coefficients of the corresponding terms in the top row as they appear in the approximation sums (each sum must be divided by h). For example, in the cubic approximation the derivative at $x = 0$ is computed as

$$\frac{df}{dx} \simeq \frac{1}{h}\left[\frac{1}{24}f\left(-\tfrac{3}{2}h\right) - \frac{27}{24}f\left(-\tfrac{1}{2}h\right) + \frac{27}{24}f\left(\tfrac{1}{2}h\right) - \frac{1}{24}f\left(\tfrac{3}{2}h\right)\right]. \tag{16}$$

Note that the set of coefficients in the table is antisymmetric about the (empty) $f(0)$ column.

Breakpoint 4: Write out the expression for the quartic (degree 4) approximation to the derivative.

4.4 Higher-order Derivatives

To compute the second derivative of a function, one can imagine taking the output of the first derivative and feeding it back into the derivative function to get the second derivative. Applying this approach to abstract functions rather than to numerical data is a bit tricky, and will be postponed to a later module. An alternative approach is to start, as before, with essentially the definition of the second derivative (in central-difference form):

$$f''(x) = \frac{f'(x + h/2) - f'(x - h/2)}{h}. \tag{17}$$

We model the two derivatives with the central difference approximations about the points $x \pm h/2$:

$$f'(x + h/2) \simeq \frac{f'(x + h) - f'(x)}{h} \qquad \text{and} \qquad f'(x - h/2) \simeq \frac{f'(x) - f'(x - h)}{h}. \tag{18}$$

Substituting these into Eq. (17), and performing a little algebra, we end up with the following expression for $f''(x)$ in terms of $f(x)$

$$f''(x) \simeq \frac{f(x + h) - 2f(x) + f(x - h)}{h^2}. \tag{19}$$

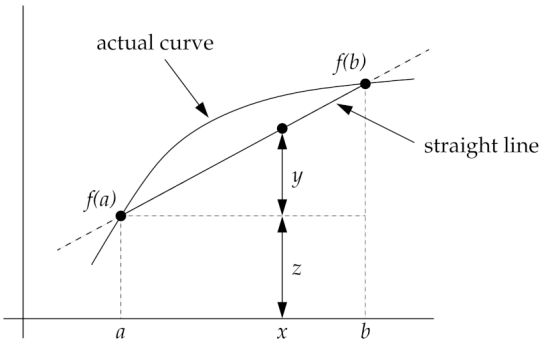

Figure 2: Linear interpolation of a function

The error in this expression can be found in the same manner as that used for the first derivative. The results are that the optimal value for h and the corresponding minimum error in $f''(x)$ are

$$h_{opt} = \left(48\epsilon \left|f(x)/f''''(x)\right|\right)^{1/4}, \qquad \text{Second derivative optimal step size} \tag{20}$$

$$E_{min} = \sqrt{\tfrac{4}{3}\epsilon \left|f(x)f''''(x)\right|}. \qquad \text{Second derivative minimum error} \tag{21}$$

As with the first derivative, there are higher-order approximations to the second derivative, but Eq. (19) will be sufficient for our purposes.

4.5 Interpolation

Interpolation – estimating the value of a function between two known points – uses similar mathematics to differentiation. If we know the value of a function $f(x)$ at $x = a, b$ (with $b > a$) then the simplest estimate for its value at an intermediate point is given by ***linear interpolation***, based on the approximation that the function changes linearly between the known points. See Figure 2.[4]

The slope of the assumed straight line connecting the two known values of the function would be $m = \dfrac{f(b) - f(a)}{b - a}$, and so at the intermediate point x, the estimate for $f(x)$ would be

$$f(x) = y + z = \frac{f(b) - f(a)}{b - a}(x - a) + f(a) = \frac{(b - x)f(a) + (x - a)f(b)}{b - a}. \tag{22}$$

This is the fundamental linear interpolation formula, and it may also be used for extrapolation beyond the interval (a, b), but not too far beyond.

One can estimate the error in this interpolation by writing $f(a)$ and $f(b)$ in terms of Taylor expansions around $f(x)$. Substitution of the two resulting equations into Eq. (22), followed by a

[4] From *Computational Physics* by Mark Newman

little algebraic manipulation, yields

$$f(x) = \frac{(b-x)f(a) + (x-a)f(b)}{b-a} + (a-x)(b-x)f''(x) + \ldots . \tag{23}$$

The first term on the right is just the linear interpolation formula, so the remaining terms represent the error. The leading-order error term, proportional to $f''(x)$, vanishes as the intermediate point x approaches endpoints a and b, as we'd expect (we're assuming we know the values of the function exactly at those points); thus, the error will be largest in the middle of the interval (assuming $f''(x)$ varies smoothly). At the midpoint, $x - a = b - x = \frac{1}{2}h$, so the leading-order error is of magnitude $\frac{1}{4}h^2|f''(x)|$, or $\mathrm{O}(h^2)$, just as for the central-difference formula for a first derivative (which interpolation resembles).

In contrast to the situation of the derivative approximated by the central-difference, rounding error is not a concern in linear interpolation, since the right-hand side of Eq. (22) involves a *sum* of terms rather than a difference.

Breakpoint 5: Suppose you want to do an extrapolation from the linear fit between the points $x = a$ and $x = b$ to a nearby point $x = c$ ($> b$). How would Eq. (23) appear in this case?

If the value of the function is known at more than two points, then one can use higher-order polynomial approximations for more accurate interpolation. However, there is a danger in using polynomials of too high order, as they wiggle a lot and therefore may not represent the function well in between the known points. Instead, one could use lower-order polynomials connecting small groups of adjacent points of the function, but polynomials defined in adjacent regions tend not to match up well to each other where they meet, so the interpolation may have a cusp at such points. A better approach is to fit polynomials to regions of the function so that they both pass through the function points and have the same derivative where they meet. Such interpolating curves are called ***splines***. The most common ones use cubic polynomials, and so are called ***cubic splines***. We will not discuss them here, but you are likely to encounter them if you use a program like Excel or Mathematica to fit data. (The `scipy` package also has spline-fitting functions.)

Recap

- Several numerical approximations to the analytical derivative of a function based on linear approximations to that function exist: the forward-, backward-, and central-difference methods, with the latter generally being more accurate than the others.
- More-accurate derivative approximations can be obtained by using higher-order polynomial approximations to the function.

- An approach very similar to that used to derive the forward-, backward-, and central-difference methods can be employed to obtain an approximate expression for the second derivative in terms of the original function evaluated at three different points.
- Linear interpolation can be used to approximate values of a function between sample points or beyond, but not too far beyond, the range limits of the sample points.

Exercises

Exercise #1

Perform the Taylor expansions of $f(x + h/2)$ and $f(x - h/2)$ analytically, take their difference, and show that the leading-order error term is O(h^2).

Exercise #2

(a) In Python create an array of data by evaluating the function $f(x) = 1 + x + 3x^3$ at 100 equally-spaced points in the range $-2 \leq x \leq 2$. (b) Write out pseudocode for computing the derivative of the data using the central difference method. (It should involve a loop over the x values. You will have to think carefully about the starting and ending values of the loop.) (c) Now write actual code and run it, plotting $f(x)$ and its derivative. (d) Determine an analytic (exact) formula for the derivative and make a second plot of both the exact and central-difference results. (Shoe your data as points and the analytical derivative as a solid curve.) (e) Make a third plot showing the *difference* between the analytic and central-difference methods. Discuss all of your results.

Exercise #3

(a) Write new code to use the forward and backward difference methods to compute the derivatives of the data of Exercise #3. (b) Plot these two derivatives along with the data from the analytical formula. (c) Plot the *difference* between the results of the central and forward methods, and also plot the difference between the results of the central and backward methods. Discuss all of the results.

Exercise #4

Code up the cubic approximation to the derivative and apply it to the function data from Exercise #2. Discuss your results.

Exercise #5

Code up Eq. (19) and apply it to the function data used in Exercise #2. Discuss your results.

NEW Mastery Exercise

In the previous Exercises, derivatives were computed by acting on *arrays of data*. (This is the *only* way to do differentiation in older programming languages.) A more sophisticated way to do numerical differentiation, possible in modern programming languages like Python, works by operating with *functions*. This approach to the derivative will be explored in this Exercise.

(a) Create the following Python function to implement $f(x)$ of Exercise #2:

```
def f(x):
    return 1 + x + 3 * x**3
```

Also generate the same array of data points you created in Exercise #2, at which you will compute the derivative of `f(x)`.

(b) Now, create a Python function that will compute the central-difference derivative of another function passed as an argument, and pass `f` as defined in (a). The syntax to do this is simply `central_deriv(f, xlist, h)`. (Of course, you can name your derivative function whatever you want.) Note that you don't pass any arguments of `f` – you pass just its *name*. The function call also passes the array of x values (labeled `xlist`) and the small-interval `h` used in Eq. (9). Inside your derivative function, you would evaluate and return the central-difference derivative of `f` just as shown in Eq. (9). Plot your derivative results and the analytical derivative on the same graph.

OLD Mastery Exercise

In the previous Exercises, derivatives were computed by acting on *functions*. This is a sophisticated way to do numerical differentiation, made possible by modern programming languages. The more traditional way (and the *only* way, in older languages) is to operate directly on *arrays of data*. This approach to the derivative will be explored in this Exercise.

(a) Write a Python function to compute the forward-difference derivative of a set of values given as an array, and run it on the output of the sin() function evaluated at 100 equally-spaced points in the range $0 \leq x \leq 2\pi$. (The array of x values, as well as their sines, should be inputs to your derivative function.) Compute the forward-differences using a loop, with the separation between consecutive x values being the finite difference h. (Note: You will need to think about what happens at the endpoints of the interval.) Plot your resulting derivative data as points, and the true derivative as a line, all on one graph. How well do they match?

(b) Repeat part (a) but set h to twice the value used there. (You will have to adjust your loop indexing appropriately.) How does the computed derivative data compare with the true derivative in this case? Is it more or less accurate than in part (a)? Could you have set h to half of the value used in part (a), as we could have done in the prior Exercises? If not, why not? (In other words, why are the possible values of h here different from those of the previous Exercises?)

Breakpoint Answers

Breakpoint 1: The fractional error is $\dfrac{0.0345}{100000000000000} \simeq 3\times10^{-16}$ or $\sim 3\times10^{-14}\%$ – *very* small.

Breakpoint 2: Differentiating E with respect to h and setting equal to zero, get

$$0 = \tfrac{1}{2}|f''(x)| - 2\epsilon|f(x)|/h_{opt}^2, \tag{24}$$

so

$$h_{opt}^2 = \frac{2\epsilon|f(x)|}{\frac{1}{2}|f''(x)|} \rightarrow h_{opt} = \sqrt{4\epsilon\left|\frac{f(x)}{f''(x)}\right|}. \tag{25}$$

Substituting into E, find

$$E_{min} = \tfrac{1}{2}|f''(x)|\sqrt{4\epsilon\left|\frac{f(x)}{f''(x)}\right|} + \frac{2\epsilon|f(x)|}{\sqrt{4\epsilon\left|\frac{f(x)}{f''(x)}\right|}} = \sqrt{4\epsilon|f(x)f''(x)|}. \tag{26}$$

Breakpoint 3: The derivative of $y = ax^2 + bx + c$ is $y'(x) = 2ax + b$. Setting $x = 0$, this becomes $y'(0) = b$, as claimed.

Breakpoint 4: Using the row of Table 1 corresponding to degree 4, we find

$$\frac{df}{dx} \simeq \frac{1}{h}\left[\frac{1}{12}f(-2h) - \frac{2}{3}f(-h) + \frac{2}{3}f(h) - \frac{1}{12}f(2h)\right]. \tag{27}$$

Breakpoint 5: Substitute $x = c$ into Eq. (23) to get

$$f(x=c) = f(a) + \frac{f(b)-f(a)}{b-a}(c-a) = \frac{(b-c)f(a)+(c-a)f(b)}{b-a}. \tag{28}$$

Scientist Profile

Mary Kenneth Keller was born in 1914 in Cleveland Ohio. She entered the Sisters of Charity of the Blessed Virgin Mary in 1932 and then took her vows with the order in 1940. At DePaul University she received a B.S. degree in mathematics and an M.S. in mathematics and physics. As a graduate student, Keller also studied at Dartmouth, Purdue, and the University of Michigan. At Dartmouth, she broke the gender barrier and worked in the previously all-male computer center, and helped develop the BASIC programming language. BASIC made computers accessible to a larger portion of the population, allowing anyone to write custom software, not just mathematicians and scientists. In 1965, she received a Ph.D. degree in computer science from the University of Wisconsin. For her dissertation she used CDC FORTRAN 63 to construct algorithms that performed analytic differentiation on algebraic expression. After earning her PhD, she founded the computer science department at Clarke College in Iowa and proceeded to direct it for twenty years. She has a Computer Science Scholarship in her honor. Sister Mary Kenneth was passionate about providing access and information to everyone, not just people well- versed in computer science. She was very forward-thinking, envisioning a world where computers would be able to think for themselves.