Intro to Computational Math, Spring 2026
Section 8, March 31 (not due or graded).

TL;DR A **finite difference formula** approximates $f'(x)$ via a weighted combination of nearby function values:

$$f'(x) \approx \frac{1}{h} \sum_{k=-p}^{q} a_k f(x + kh).$$

The weights a_k must be independent of f and h. If the approximation becomes exact as $h \to 0$, the method is **convergent**.

Common two-node methods come from secant lines. The backward difference uses $\frac{f(x)-f(x-h)}{h}$, the forward difference uses $\frac{f(x+h)-f(x)}{h}$, and the central difference uses $\frac{f(x+h)-f(x-h)}{2h}$. More generally, for any two nodes $s \le x \le t$, we approximate $f'(x) \approx \frac{f(t)-f(s)}{t-s}$.

Higher-order methods via Lagrange interpolation. Given nodes $\{a+x_1, \ldots, a+x_n\}$, construct the Lagrange interpolation $L(x)$ through those points and approximate $f'(a) \approx L'(a)$. The weights on each $f(a + x_i)$ are simply $L_i'(0)$, where L_i are the Lagrange interpolants built on the node differences (i.e. interpolation through $x_1, ..., x_n$). This is what gives us, for example, the 4-node symmetric method:

$$f'(a) \approx \frac{1}{12h} f(a-2h) - \frac{2}{3h} f(a-h) + \frac{2}{3h} f(a+h) - \frac{1}{12h} f(a+2h).$$

Order of accuracy. The **truncation error** $\tau_f(h) = f'(0) - \frac{1}{h}\sum a_k f(kh)$ measures how well the method approximates the derivative. If $\tau_f(h) = O(h^m)$, the method has order of accuracy m. To find the best weights (and the order), Taylor expand each $f(kh)$ around 0, substitute in, and choose weights so that the lowest-order terms cancel. The forward difference is order 1; the central difference is order 2.

Stability and optimal node spacing. Smaller h reduces truncation error but *increases* round-off error due to subtractive cancellation. The total error behaves like

$$|f'(x) - \bar{\delta}(h)| \lesssim Kh^m + \frac{|f(x)|}{h}\varepsilon_{\texttt{mach}}.$$

Minimizing this gives an optimal step size of $h_{\texttt{opt}} \sim \varepsilon_{\texttt{mach}}^{1/(m+1)}$. In practice, for a first-order method this is around $\sqrt{\varepsilon_{\texttt{mach}}} \approx 10^{-8}$, and for a second-order method around $\varepsilon_{\texttt{mach}}^{1/3} \approx 10^{-6}$.

Remark 1

I didn't have time to write up notes on Richardson extrapolation, but it may be useful to read up on how one may produce much higher convergent finite difference methods without increasing the number of nodes.

1 Finite Differences

Interpolation has many applications in data science, optimization, and numerical analysis. One particular use case for the latter in with finite differences. We will see a lot of these from now on, so it will be prudent to get used to their analysis and application. Below, we give the general form of a finite difference, but we will step back quickly to more comfortable and digestible examples. We will then explore how one could derive the coefficients used for optimal finite differences (gives a set of interpolates) and finally, we will analysis their convergence guarantees in the next chapter. In general, nodes need not be evenly spaced. Here we assume they are.

Definition 1.1

*A **finite difference formula** is a list of weights, values $a_{-p}, , ..., a_q$, such that*

$$f'(x) \approx \frac{1}{h} \sum_{k=-p}^{q} a_k f(x + kh).$$

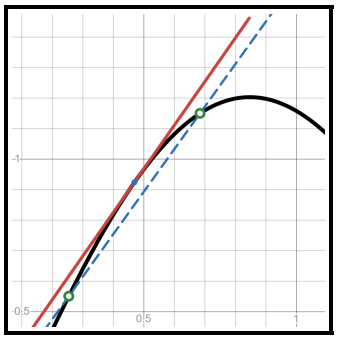
*The weights must be independent of function f and node width h. If the approximation becomes an equality as $h \to 0$, this formula is said to be **convergent**.*

The general idea above is that we approximate a derivative by taking some weight combination of points nearby. A specific example is that of approximating via a secant line.

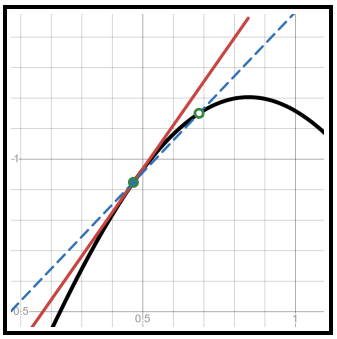
1.1 Common Examples and Derivations

Perhaps this simplest examples are that with only two nodes. As mentioned above, these correspond to approximating via a secant line. Doing so, we can either take nodes with one on a point of interest and the other behind, the other ahead, or have both nodes on either side. We note by inspection (for now) that the central difference method appears to be far more accurate. Here is a Desmos Graph that can illistrate these further. Go to "Finite Difference 3" to play around with general choices of nodes.

Backwards Difference



Central Difference



Forward Difference

Example 1.1

Here are the explicit formulae for the finite differences above

- ***backward difference:*** $f'(x) \approx \frac{f(x)-f(x-h)}{h}$
- ***central difference:*** $f'(x) \approx \frac{f(x+h)-f(x-h)}{2h}$
- ***forward difference:*** $f'(x) \approx \frac{f(x+h)-f(x)}{h}$

These formulas above directly come from just taking the secant line. For example, if x_1 represents the left node and x_2 represents the second node and we are doing a central difference, then $x_2 - x_1 = (x+h) - (x-h) = 2h$. In fact, this works for any choices of nodes when we consider a finite difference formula with only two nodes.

Proposition 1.2

Let $x \in \mathbb{R}$ be fixed and choose $s \leq x \leq t$ where $s \neq t$. Then we can approximate

$$f'(x) \approx \frac{f(t) - f(s)}{t - s}.$$

In particular if $s = x - k_1 h$ and $t = x + k_2 h$ then

$$f'(x) = \frac{f(x + k_2 h) - f(x - k_1 h)}{(k_1 + k_2)h}$$

We note that the line that passes through two points s and t where $s \neq t$ can always be written as

$$p(x) = f(s) + \frac{f(t) - f(s)}{t - s}(x - s).$$

Notably the slope of this line is precisely the value use the approximate the derivative in Proposition 1.2. In particular, we can say that $f'(x) \approx p'(x)$.

1.2 Generalizations and Higher Order Approximations

In this section with approximate derivatives by building up finite differences with more nodes. We will see in the next section that this implies a higher order of convergence as well. As before, when we had two points in our toolbox, we took the line that went through them. That is, we took the 1-degree interpolation.

For more than two points, we could try a few routes. We could take each line that goes through any pair of points and average them. We could find a line of best fit using least squares. We could also interpolate all point using Lagrange interpolation. It turns out the latter gives the best approximation. The intuition here is that the Lagrange Polynomial gives the "best" polynomial for interpolating points nearby, so it should approximate the slope up to a similar order. For example, if we use 4 points, the interpolating polynomial is of degree 3, and will match the degree 3 Taylor polynomial of the function, up to higher order terms. Thus, the derivative will also match up to (the derivative of) the higher order terms.

Definition 1.3 (Lagrange Polynomial)

Given a set of points $X = \{x_1, ..., x_n\}$ with function values $Y = \{y_1, ..., y_n\}$. Denote each

Lagrange Interpolant as

$$L_i(x) = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}, \quad i = 1, ..., n.$$

Then the Lagrange Interpolation is the full polynomial

$$L(x) = \sum_{i=1}^{n} y_i L_i(x).$$

Remark 1

Using the definition above, note that $L_i(x)$ is of degree $n-1$ and $L_i(x_j) = 0$ for $i \neq j$, and $L_i(x_i) = 1$. Therefore,

$$L(x_j) = \sum_{i=1}^{n} y_i L_i(x_j) = y_j L_j(x_j) = y_j$$

by noting that most of the terms in the sum evaluate to zero. Therefore, the polynomial is an interpolation.

Lagrange interpolation will be the key to constructing finite difference formula. The idea is the fix a set of nodes, "centered" (not necessarily symmetrically) around a point. Then, we construct the interpolation of those points and the idea is that the derivative of the function should approximately match the derivative of the interpolation.

Theorem 1.4

Let $X = \{a + x_1, ..., a + x_n\}$ be a set of nodes (here x_i are allowed to be negative) to construct a finite difference method. Let $L_{\texttt{nodes}}(x)$ be the Lagrange interpolation of the nodes with their function values. Then

$$f'(a) \approx L'_{\texttt{nodes}}(a) = \sum_{i=1}^{n} f(a + x_i) L'_i(a)$$

Remark 2

One further simplification going forward is we can note that these finite difference methods should not depend on where we are evaluating. Therefore, we can without loss of generality let $a = 0$. Equivalently, we can image making the change of variables $x \mapsto x - a$ went interpolating.

The following corollary tells us exactly what we should expect from a finite difference method. The weights on the function values to approximate the derivative are independent of the function, the center, and only depends on the nodes. What isn't fully displayed by this corollary is that if the nodes are equally spaced, then the node width is also independent of the weights. This comes from algebraic manipulation of the Lagrangian interpolation and is left as an exercise to the reader.

Corollary 1.5

Let $a \in \mathbb{R}$ be a center for which we try to approximate a derivative and $X = \{a + x_1, ..., a + x_n\}$.

Let $L'_{\texttt{diff}}(x)$ interpolating the points $\{x_1, ..., x_n\}$. Then

$$f'(a) \approx L'_{\texttt{diff}}(0) = \sum_{i=1}^{n} f(a + x_i) L'_i(0).$$

That is, the ***weights*** *on $f(a + x_i)$ are simply the derivative of each Lagrange Interpolant (constructed just on the* ***differences****) at zero.*

The generality of the above statements can be quite difficult to digest, so it may be useful to see a particular example. **Note that the weights defined here will have an extra factor of $1/h$, so we just need to be careful!**

Example 1.2

Suppose we wish to find an approximation for $f'(a)$ by considering nodes

$$X = \{a - 2h, a - h, a + h, a + 2h\}.$$

We find the Lagrange interpolation to fit the data

$$\{-2h, -h, h, 2h\} \text{ and } \{f(a - 2h), f(a - h), f(a + h), f(a + 2h)\}.$$

The Lagrange Interpolants look like

$$\begin{aligned}
L_{-2}(x) &= \frac{(x - (-h))(x - h)(x - 2h)}{(-2h - (-h))(-2h - h)(-2h - 2h)} = -\frac{x^3 - 2hx^2 - h^2x + 2h^3}{12h^3} \\
L_{-1}(x) &= \frac{(x - (-2h))(x - h)(x - 2h)}{(-h - (-2h))(-h - h)(-h - 2h)} = \frac{x^3 - hx^2 - 4h^2x + 4h^3}{6h^3} \\
L_{+1}(x) &= \frac{(x - (-2h))(x - (-h))(x - 2h)}{(h - (-2h))(h - (-h))(h - 2h)} = -\frac{x^3 + hx^2 - 4h^2x - 4h^3}{6h^3} \\
L_{+2}(x) &= \frac{(x - (-2h))(x - (-h))(x - h)}{(2h - (-2h))(2h - (-h))(2h - h)} = \frac{x^3 + 2hx^2 - h^2x - 2h^3}{12h^3}
\end{aligned}$$

Using our corollary, noting that $f'(a) \approx L'(0) = \sum_{k \in \{-2, -1, 1, 2\}} f(a - kh) L'_k(0)$ which directly implies the weights on $f'(a + kh)$ is precisely $L'_k(0)$. Computing these

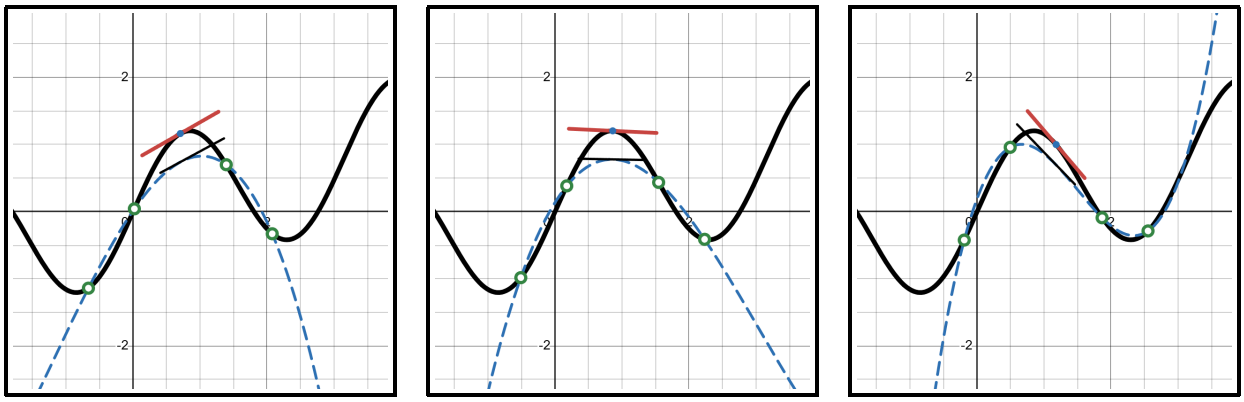
$$\begin{aligned}
L'_{-2}(0) &= -\frac{3x^2 - 4hx - h^2}{12h^3}\bigg|_0 = \frac{1}{12h} \\
L'_{-1}(0) &= \frac{3x^2 - 2hx - 4h^2x}{6h^3}\bigg|_0 = \frac{-2}{3h} \\
L'_{+1}(0) &= -\frac{3x^2 + 2hx - 4h^2}{6h^3}\bigg|_0 = \frac{2}{3h} \\
L'_{+2}(0) &= \frac{3x^2 + 4h - h^2}{12h^3}\bigg|_0 = \frac{-1}{12h}
\end{aligned}$$

Therefore, we can write out explicitly that for any differentiable function

$$f'(a) \approx \frac{1}{12h} f(a-2h) - \frac{2}{3h} f(a-h) + \frac{2}{3h} f(a+h) - \frac{1}{12h} f(a+2h)$$

which exactly matches the formula in the textbook

Below we show that the order 4 finite difference method constructed above is shocking accurate even for drastically large node width and realatively oscillatory functions.



Order 4 Finite Difference Methods for Various Centers

2 Convergence Guarantees for Finite Differences

So far, we have discussed ways of constructing finite difference methods. Now we can analyze their convergence guarantees.

So far, we have discussed ways of constructing finite difference methods. Now we can analyze their convergence guarantees.

2.1 Order of Accuracy

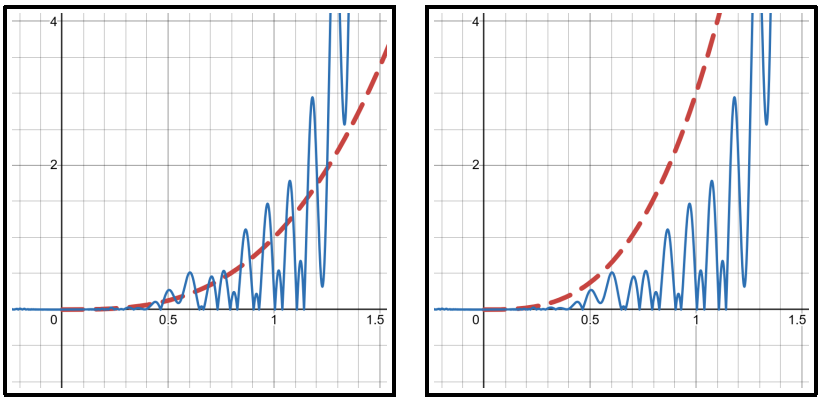
This section is quick, and should just be meant as a review of order of accuracy.

Definition 2.1 (informal)

*We say a convergence rate is of "**Big-O**" complexity if the error goes to zero faster than that complexity.*

When we see that a rate is of $O(h^3)$, what this should mean is that as $h \to 0$, the error converges to zero faster than $O(h^3)$ would, up to universal constants. That is to say, the rate of convergence need to be less than $f(h) = h^3$ everywhere, or even at all, but just that there needs to be some positive constant such that $[\text{error}(h)] \leq Ch^3$.

When we say a function is of the order of another, we typically write $f(x) = O(g(x))$. Sometimes we may see "little-o" notation as in $f(x) = o(g(x))$. The subtle difference is just related to the tightness of these bounds. That is, the analog of big-O to little-o is the same as greater than *or equal to*, or strictly greater than.



Display of $f(x) = x^3$ and $f(x) = 10x^3$ versus some error

Example 2.1

Here are some examples

- $h^n = O(h^m)$ *iff* $n \geq m$
- $h^n = o(h^m)$ *iff* $n > m$
- $|sin(h) - h| = o(h^2)$
- $e^{-1/h} = o(x^m)$ *for all* m
- $h^2 \ln(1/h) = o(h)$

There are more formal ways of describing these notations with limits, but the idea remains the same.

2.2 Truncations Error

With error in mind, we can define the truncation error for a specific finite difference method. Going forward, for simplicity, we shall only consider methods with equally spaced nodes. Furthermore, we consider only approximating derivatives at 0.

Definition 2.2 (Truncation error of a finite-difference formula)

For a finite difference method with weights $a_{-p}, ..., a_q$, *the* ***truncation error*** *is*

$$\tau_f(h) = f'(0) - \frac{1}{h} \sum_{k=-p}^{k} a_k f(kh).$$

If $\tau_f(h) \to 0$ *as* $h \to 0$, *the method is said to be convergent.*

Note that simply by construction all finite difference methods constructed from above will be convergent. The intuition is that we are interpolating with some polynomial that perfectly fits through the nodes. As the nodes get closer, this polynomial gets closer to fitting through our center,

and in conjugation with the tools of Taylor series, we can show that the derivative must converge as well.

Regardless, some methods outperform others. Some converge faster, and some converge slower. We really just care about the rate of convergence, which typically ignores constants. Therefore, big (or small)-O notation is the perfect setting.

Definition 2.3

If the truncation error of a method $\tau_f(h) = O(h^m)$ for a positive integer m, then we say the ***order of accuracy*** *is m.*

We can then compute the order of accuracy explicitly. The standard way to do so is to simply take a Taylor expansion and see what cancels. We will see shortly that we can similarly construct these finite difference methods to be with the weights that will cause the highest terms in a Taylor expansion to cancel. In this framework, we can both construct methods and determine their order of convergence at the same time!

Lemma 2.4

The finite difference method

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

has order 1. That is $\tau_f(h) = O(h)$.

The finite difference method

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

has order 2. That is $\tau_f(h) = O(h^2)$.

Proof. The proofs for both of these come from just looking at the Taylor expansion. See examples 5.5.1 and 5.5.2 in the textbook. □

We can, however constructively show that the best choice of weights for the central difference method *must* be $\frac{1}{2}$ and $-\frac{1}{2}$ respectively.

Example 2.2

Suppose we wish to approximate $f'(x)$ with nodes $f(x-h)$, $f(x)$ and $f(x+h)$. What values of $a, b, c \in \mathbb{R}$ maximize the order of accuracy?
Without loss of generality, lets take $x = 0$, so we wish to maximize the order of accuracy of

$$\frac{af(-h) + bf(0) + cf(h)}{h}.$$

Taylor expanding around $x = 0$ we get

$$f(-h) = f(0) + f'(0)(-h) + \frac{f''(0)}{2}(-h)^2 + \frac{f'''(0)}{6}(-h)^3 + O(h^4)$$
$$f(0) = f(0) + f'(0)(0) + \frac{f''(0)}{2}0^2 + \frac{f'''(0)}{6}0^3 + O(h^4)$$
$$f(h) = f(0) + f'(0)(h) + \frac{f''(0)}{2}(h)^2 + \frac{f'''(0)}{6}(h)^3 + O(h^4)$$

With some rearrangement our approximation $\frac{af(-h)+bf(0)+cf(h)}{h}$ equals

$$\frac{(a+b+c)f(0) + (-a+0b+c)f'(0)h + \frac{1}{2}(a+0b+c)f''(0)h^2 + \frac{1}{6}(-a+0b+c)f'''(0)h^3 + O(h^4)}{h}.$$

Notably, we want no zero order terms (i.e. $f(0)$), we want a single first order term (i.e. $f'(0)$) and to vanish as many subsequent terms as possible. We only have three variables to play with though so we aim for

$$\frac{a+b+c}{h} = 0$$
$$-a+c = 1$$
$$(a+c)h = 0$$

which gives us $(a, b, c) = (-1/2, 0, 1/2)$ as expected. Furthermore, this tells us that

$$\frac{f(h) - f(-h)}{2h} = \frac{f'''(0)}{6}h^2 + O(h^3)$$

after dividing by h, and since that h^2 term remained, the finite difference method is $O(h^2)$.

2.3 Stability, Round off Error, and Optimal Node Spacing

Since the finite difference methods are all of order $O(h^m)$ it would be expected that smaller h means better approximation. However, this is not always the case. At least, when dealing with computation math on computers, we always have to worry about round off errors and stability. In fact, since we are taking differences of numbers that are typically not zero (i.e. $f(x+h)$ and $f(x)$), this is a recipe for bad subtraction.

Example 2.3

Suppose the method we are using is the simple case of forward difference

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

We wish to analyze the total error, from both truncation (the approximation of the finite difference method) and the round off error (from the approximation the compute makes). That

is, suppose

$$\delta(h) = \frac{f(x+h) - f(x)}{h}$$

is the true approximation of the derivative. However, we can estimate exact how off the compute will be. First we do this nasty subtraction $f(x+h) - f(x)$ which has condition number

$$\kappa(h) = \frac{\max\{|f(x+h)|, |f(x)|\}}{|f(x+h) - f(x)|}.$$

Since the relative error is of size $\kappa(h)\varepsilon_{\texttt{mach}}$ we can compute the numerical value calculated by the compute to be

$$\begin{aligned}
\bar{\delta}(h) &= \delta(h)(1 + \text{ relative error}) \\
&= \delta(h)(1 + \kappa(h)\varepsilon_{\texttt{mach}}) \\
&= \delta(h) + \frac{f(x+h) - f(x)}{h} \frac{\max\{|f(x+h)|, |f(x)|\}}{|f(x+h) - f(x)|}\varepsilon_{\texttt{mach}} \\
&\sim \frac{|f(x)|}{h}\varepsilon_{\texttt{mach}}
\end{aligned}$$

Finally, utilizing the triangle inequality, we get that error between the true derivative and the one the compute spits out is

$$\begin{aligned}
|f'(x) - \bar{\delta}(h)| &\leq \underbrace{|f'(x) - \delta(h)|}_{\text{truncation error}} + \underbrace{|\delta(h) - \bar{\delta}(h)|}_{\text{round off error}} \\
&\sim |\tau_f(h)| + \frac{|f(x)|}{h}\varepsilon_{\texttt{mach}}
\end{aligned}$$

So, this true difference has this "push-pull" factor as $h \to 0$. The truncation error goes down, but the round off error goes up. Therefore, there is some ideal "middle ground". In particular we solve

$$\min_h |\tau_f(h)| + \frac{|f(x)|}{h}\varepsilon_{\texttt{mach}}$$

If $\tau_f(h) = O(h^m)$ we can say (with some hand waviness) that $\tau_f(h) = Kh^m$ and therefore, this problem is solved via some basic calculus. We take the derivative of

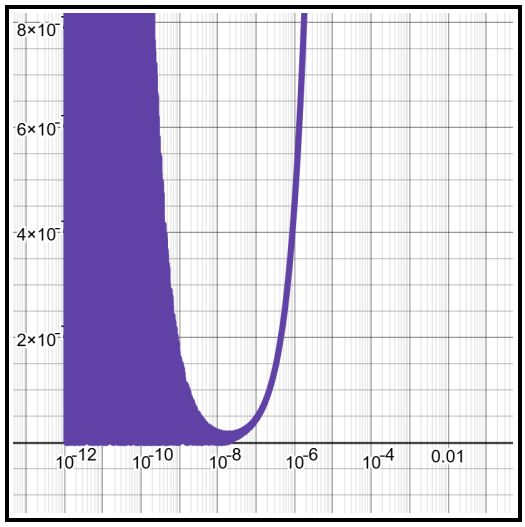
$$Kh^m + |f(x)|\varepsilon_{\texttt{mach}}h^{-1}$$

and set it to zero and solve. Therefore

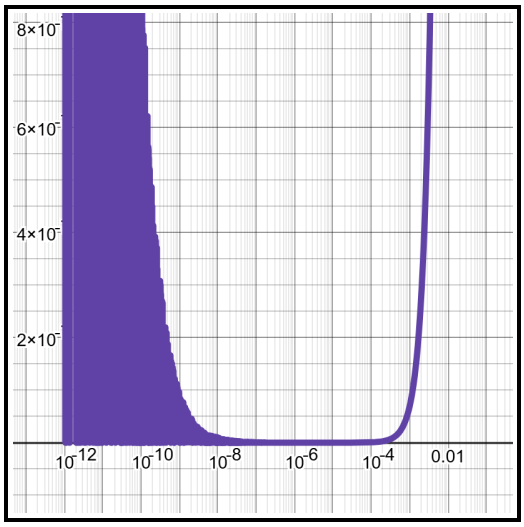
$$h_{\texttt{opt}} \sim \varepsilon_{\texttt{mach}}^{\frac{1}{m+1}}$$

The really interesting aspect of the above problem is we can actually see this in practice! If we take some arbitrary function, in this example it is just $f(x) = \sin(x)$. We know the derivative $f'(x_0) = \cos(x_0)$. We can then, in the computer, plot the error it calculates between the true value and its approximation via some finite difference method as a function of h.

We can then see that for a first order method, the h that minimizes the true error is around $10^{-8} \approx \sqrt{\varepsilon_{\texttt{mach}}}$ and the error that minimizes the second order method is around $10^{-6} \approx \sqrt[3]{\varepsilon_{\texttt{mach}}}$. A lot of noise gets introduced, but we can still get a visual of what is happening.



Error of $|\cos(x_0) - \frac{f(x_0+h)-f(x_0)}{h}|$



Error of $|\cos(x_0) - \frac{f(x_0+h)-f(x_0-h)}{2h}|$

3 Examples and Practice

Problem 3.1

As seen earlier `(i.e., Section #2)`*, consider*

$$f(x) = \log(\exp(x) + \exp(-x)) \ .$$

Evaluating this function directly can be unstable. This time, let's evaluate its derivatives.

a) *The derivative of f is given by*

$$f'(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$$

Is f' reasonably well-conditioned at $x = 1.23$? You can answer this by directly finding condition number of f' at $x = 1.23$.

b) *Consider numerically approximating $f'(1.23)$ for a given stepsize $h > 0$ via the formula*

$$\frac{f(1.23 + h) - f(1.23)}{h}.$$

If implemented in standard floating point arithmetic, what value of h would you recommend using to minimize the difference from the true answer $f'(1.23)$?

c) *What values of $a, b, c \in \mathbb{R}$ maximize the order of accuracy for approximating $f'(0)$ below?*

$$\frac{af(-h) + bf(0) + cf(2h)}{h}$$

Solution:

a) Yes, the only issue could be in the numerator and $e^x \approx 3.42$ and $e^{-x} \approx 0.29$. These are not close together and so the subtraction is stable.

b) This method is a first-order method, so we choose $h \approx 10^{-8}$. Implementing this on a computer gives

$$f'(1.23) \approx \frac{f(1.23+h}{-} f(1.23)h = 0.365927543822$$

up to fifteen digits. The most accurate answer for the derivative I can calculate up to fifteen digits is $f'(1.23) \approx 0.365927551699$.

c) Here is the more interesting question. We can solve this in two ways. One involves an interpolation and taking derivatives, the other involves Taylor expansion. Pick your poison.

1) For the first method, we begin by constructing the degree-2 Lagrange interpolant through the nodes $x_0 = -h,\ x_1 = 0,\ x_2 = 2h$ and differentiate at $x = 0$.

$$L_0(x) = \frac{(x-0)(x-2h)}{(-h-0)(-h-2h)} = \frac{x(x-2h)}{3h^2},$$
$$L_1(x) = \frac{(x+h)(x-2h)}{(0+h)(0-2h)} = \frac{(x+h)(x-2h)}{-2h^2},$$
$$L_2(x) = \frac{(x+h)(x-0)}{(2h+h)(2h-0)} = \frac{x(x+h)}{6h^2}.$$

Then we take the derivatives to get

$$L_0'(x) = \frac{(2x-2h)}{3h^2} \implies L_0'(0) = \frac{-2h}{3h^2} = -\frac{2}{3h},$$
$$L_1'(x) = \frac{(2x-h)}{-2h^2} \implies L_1'(0) = \frac{-h}{-2h^2} = \frac{1}{2h},$$
$$L_2'(x) = \frac{(2x+h)}{6h^2} \implies L_2'(0) = \frac{h}{6h^2} = \frac{1}{6h}.$$

The interpolants give $f'(0) \approx L_0'(0)f(-h) + L_1'(0)f(0) + L_2'(0)f(2h)$, so matching the form $\frac{1}{h}[af(-h) + bf(0) + cf(2h)]$:

$$\boxed{a = -\tfrac{2}{3}, \quad b = \tfrac{1}{2}, \quad c = \tfrac{1}{6}.}$$

Because the interpolation is the unique degree-2 polynomial through three nodes, differentiation introduces an $O(h^2)$ error, giving a *second-order* accurate approximation.

2) If we instead wish to Taylor expand about $x = 0$:

$$f(-h) = f(0) - hf'(0) + \tfrac{h^2}{2}f''(0) - \tfrac{h^3}{6}f'''(0) + \cdots,$$
$$f(0) = f(0),$$
$$f(2h) = f(0) + 2hf'(0) + 2h^2 f''(0) + \tfrac{4h^3}{3}f'''(0) + \cdots.$$

Forming the linear combination and dividing by h:

$$\begin{aligned}\frac{af(-h)+bf(0)+cf(2h)}{h} &= \frac{(a+b+c)}{h}f(0)\\ &\quad + (-a+2c)\,f'(0)\\ &\quad + \left(\frac{a}{2}+2c\right)h\,f''(0)\\ &\quad + \left(-\frac{a}{6}+\frac{4c}{3}\right)h^2\,f'''(0)+\cdots\end{aligned}$$

To maximize the accuracy, we require the coefficient of $f'(0)$ to equal 1 and eliminate as many lower- and higher-order terms as possible.

$$a+b+c=0 \tag{C1}$$
$$-a+2c=1 \tag{C2}$$
$$\frac{a}{2}+2c=0 \tag{C3}$$

We have three equations and three unknowns, so we can simply solve this system. From (C3): $a=-4c$. Substituting into (C2): $4c+2c=1 \Rightarrow c=\frac{1}{6}$, so $a=-\frac{2}{3}$. From (C1): $b=-a-c=\frac{2}{3}-\frac{1}{6}=\frac{1}{2}$.

$$\boxed{a=-\tfrac{2}{3},\quad b=\tfrac{1}{2},\quad c=\tfrac{1}{6}.}$$

The leading error term is then

$$\left(-\frac{a}{6}+\frac{4c}{3}\right)h^2f'''(0)=\left(\frac{1}{9}+\frac{2}{9}\right)h^2f'''(0)=\frac{h^2}{3}f'''(0),$$

confirming the formula is *second-order accurate*.