Example 121. $\boxed{\text{Python}}$ Let us see how the forward and central difference compare in practice.

```
>>> def forward_difference(f, x, h):
        return (f(x+h)-f(x))/h
```

```
>>> def central_difference(f, x, h):
        return (f(x+h)-f(x-h))/(2*h)
```

We apply these to $f(x)=2^x$ at $x=1$. In that case, the exact derivative is $f'(1)=2\ln(2)\approx 1.386$.

```
>>> def f(x):
        return 2**x
```

```
>>> [forward_difference(f, 1, 10**-n) for n in range(5)]
```

```
[2.0, 1.4354692507258626, 1.3911100113437769, 1.3867749251610384, 1.3863424075299946]
```

```
>>> [central_difference(f, 1, 10**-n) for n in range(5)]
```

```
[1.5, 1.3874047099948572, 1.3863054619682957, 1.3862944721280135, 1.3862943622289237]
```

It is probably easier to see what happens to the error if we subtract the true value from these approximations:

```
>>> from math import log
```

```
>>> [forward_difference(f, 1, 10**-n) - 2*log(2) for n in range(12)]
```

```
[0.6137056388801094, 0.04917488960597205, 0.004815650223886303, 0.00048056404114782403,
4.80464101040301e-05, 4.804564444071957e-06, 4.80467807983942e-07, 4.703673917028084e-
08, 7.068710283775204e-09, 2.7352223619381277e-07, 1.161700655893938e-06,
1.8925269049896443e-05]
```

```
>>> [central_difference(f, 1, 10**-n) - 2*log(2) for n in range(12)]
```

```
[0.11370563888010943, 0.001110348874966638, 1.1100848405165564e-05,
1.1100812291608975e-07, 1.1090330875873633e-09, 1.879385536085465e-11,
7.43052286367174e-11, -7.028508886008922e-10, 7.068710283775204e-09,
1.6249993373129712e-07, 1.161700655893938e-06, 7.823038803644877e-06]
```

For the forward difference, we can see how the error decreases roughly by $1/10$ initially, as expected. Likewise, for the central difference, the error decreases roughly by $1/10^2$ (order 2) initially. However, in both cases, the errors end up increasing after a while (and never gets close to machine precision). We discuss this surprising issue in the next section.

Note how, for the forward difference, our best approximation has error $7.07\cdot 10^{-9}$ while, for the central difference, our best approximation has error $1.88\cdot 10^{-11}$. While the latter is an improvement, either is worryingly large.

The errors in numerical differentiation

In practice, we always get two kinds of errors: the **theoretical error** as well as a **rounding error** (due to the fact that we have to round all involved quantities to, say, double precision). In the past, we have been able to mostly ignore the rounding error. However, we cannot afford to do so in the case of numerical differentiation. The reason for the trouble is that our finite difference approximations always subtract nearly equal quantities (that's in the nature of differentiation!) which can lead to a devastating loss of precision.

Comment. The theoretical error is often also called **truncation error**. Here, truncation is meant in the sense of, for instance, having a function $f(x)$ and truncating its Taylor series to get an approximation of $f(x)$.

Example 122. Analyze the overall error in using the approximation $f'(x) \approx \frac{1}{h}[f(x+h) - f(x)]$.

Solution. As we worked out in Example 114, the theoretical error is $\frac{1}{h}[f(x+h) - f(x)] = f'(x) + \frac{h}{2} f''(\xi)$.

In practice, all quantities including $f(x+h)$ and $f(x)$ are slightly rounded and only accurate to within some ε (the machine precision). In the sequel, we assume double precision, in which case $\varepsilon \approx 2^{-52} \approx 2.2 \cdot 10^{-16}$. Our approximation therefore is (ε_1 and ε_2 are the rounding errors for $f(x+h)$ and $f(x)$ respectively) roughly:

$$\begin{aligned}\frac{1}{h}[(f(x+h)+\varepsilon_1) - (f(x)+\varepsilon_2)] &= \frac{1}{h}[f(x) - f(x-h)] + \frac{\varepsilon_1 - \varepsilon_2}{h}\\ &= f'(x) + \underbrace{\frac{1}{2} f''(\xi) h}_{\text{theoretical error}} + \underbrace{\frac{\varepsilon_1 - \varepsilon_2}{h}}_{\text{rounding error}}\end{aligned}$$

Writing $M = \frac{1}{2}|f''(\xi)|$ (note that $M \approx \frac{1}{2}|f''(x)|$ if h is small) and observing that $\left|\frac{\varepsilon_1 - \varepsilon_2}{h}\right| \leqslant \frac{2\varepsilon}{h}$, we see that the overall error is bounded by

$$E(h) = Mh + \frac{2\varepsilon}{h}.$$

Plot the function $E(h)$ to see that this bound becomes bad as h gets too small (that's because of the rounding error $2\varepsilon/h$). In particular, we can see that there will be a "best" choice for h which minimizes this bound for the error. To find this best value $h^\star$, we compute $E'(h) = M - \frac{2\varepsilon}{h^2} = 0$ and solve for h. We find

$$h^* = \sqrt{\frac{2\varepsilon}{M}} \approx M^{-1/2} \cdot 2.1 \cdot 10^{-8}.$$

The corresponding bound for the error is $E(h^*) \approx Mh^* + \frac{2\varepsilon}{h^*} \approx M^{1/2} \cdot 4.2 \cdot 10^{-8}$.

Comment. Note that our estimates match the values we observed in Example 121 fairly well. In that example, we had $f(x) = 2^x$ so that $f''(x) = (\ln 2)^2 2^x$ and, therefore, $M = \frac{1}{2}|f''(\xi)| \approx \frac{1}{2}|f''(1)| = (\ln 2)^2$. In particular, $M^{1/2} \approx \ln 2 \approx 0.6931$ and $E(h^*) \approx M^{1/2} \cdot 4.2 \cdot 10^{-8} \approx 2.9 \cdot 10^{-8}$.

Example 123. Analyze the overall error in approximating $f'(x) \approx \frac{1}{2h}[f(x+h) - f(x-h)]$.

Solution. The theoretical error is $\frac{1}{2h}[f(x+h) - f(x-h)] = f'(x) + \frac{h^2}{6} f'''(x) + O(h^3)$ (see Example 115).

In practice, proceeding as in the previous example, our approximation is roughly:

$$\begin{aligned}\frac{1}{2h}[(f(x+h)+\varepsilon_1) - (f(x-h)+\varepsilon_2)] &= \frac{1}{2h}[f(x+h) - f(x-h)] + \frac{\varepsilon_1 - \varepsilon_2}{2h}\\ &= f'(x) + \underbrace{\frac{1}{6} f'''(\xi) h^2}_{\text{theoretical error}} + \underbrace{\frac{\varepsilon_1 - \varepsilon_2}{2h}}_{\text{rounding error}}\end{aligned}$$

Note that $\left|\frac{\varepsilon_1 - \varepsilon_2}{2h}\right| \leqslant \frac{\varepsilon}{h}$. Writing $M = \frac{1}{6}|f'''(\xi)|$, the overall error is bounded by

$$E(h) = Mh^2 + \frac{\varepsilon}{h}.$$

Again, plotting the function $E(h)$ we see that this bound becomes bad as h gets too small and that there is a "best" value h^* which minimizes this bound for the error. We compute $E'(h) = 2Mh - \frac{\varepsilon}{h^2} = 0$ and solve for h. We find

$$h^* = \sqrt[3]{\frac{\varepsilon}{2M}} \approx M^{-1/3} \cdot 4.8 \cdot 10^{-6}.$$

The corresponding bound for the error is $E(h^*) \approx M(h^*)^2 + \frac{\varepsilon}{h^*} \approx M^{1/3} \cdot 6.9 \cdot 10^{-11}$.

Comment. Again, our estimates match the values we observed in Example 121 rather well.

Richardson extrapolation

Suppose that $A(h)$ is an approximation of a quantity A^* of order n. Our goal is to construct an approximation of A^* of higher order (by combining approximations $A(h)$ for different h).

If $A(h)$ approximates A^* to order n, then we often have $A(h) = A^* + Ch^n + O(h^{n+1})$ for some constant C. This is true, for instance, for all of our numerical differentiation examples.

In that case, we have $A(2h) = A^* + C(2h)^n + O(h^{n+1})$ as well as $2^n A(h) = 2^n A^* + C(2h)^n + O(h^{n+1})$.

Thus the difference is $2^n A(h) - A(2h) = (2^n - 1)A^* + O(h^{n+1})$.

Dividing both sides by $2^n - 1$, we therefore get an approximation of order higher than n.

(Richardson extrapolation) Starting with an approximation $A(h)$ of a quantity A^* of order n, its **Richardson extrapolation** is the approximation

$$R(h) := \frac{2^n A(h) - A(2h)}{2^n - 1}.$$

As we showed above, it typically is an approximation of higher order.

More generally, we get a Richardson extrapolation $R_\lambda(h) = \frac{\lambda^n A(h) - A(\lambda h)}{\lambda^n - 1}$ for any choice of $\lambda > 0$ (the choice $\lambda = 2$ is common but not mathematically special).

Comment. Note that, based on the values at h and $2h$, we are trying to get our hands on A^* which is the value at 0. Because 0 is outside of the interval $[h, 2h]$, this is an extrapolation.

Example 124. Suppose that $A\left(\frac{1}{2}\right) = \frac{3}{8}$ and $A\left(\frac{1}{3}\right) = \frac{5}{12}$ are approximations of order 3 of some quantity A^*. What is the approximation we obtain from using Richardson extrapolation?

Solution. Since $A(h)$ is an approximation of order 3, we expect $A(h) \approx A^* + Ch^3$ for some constant C.

Correspondingly, $A\left(\frac{1}{2}\right) \approx A^* + \frac{1}{8}C$ and $A\left(\frac{1}{3}\right) \approx A^* + \frac{1}{27}C$.

Hence, $27A\left(\frac{1}{3}\right) - 8A\left(\frac{1}{2}\right) \approx (27 - 8)A^* = 19A^*$. To get an approximation of A^*, we need to divide by 19.

The Richardson extrapolation is $\frac{27}{19}A\left(\frac{1}{3}\right) - \frac{8}{19}A\left(\frac{1}{2}\right) = \frac{27}{19}\frac{5}{12} - \frac{8}{19}\frac{3}{8} = \frac{33}{76}$.

Comment. What we did is equivalent to using the formula $R_\lambda(h) = \frac{\lambda^n A(h) - A(\lambda h)}{\lambda^n - 1}$ with $h = \frac{1}{3}$ and $\lambda = \frac{3}{2}$.

Comment. The numbers above are not random. Instead, $A(h) = \frac{1}{2} - \frac{h}{4}$ which is an order 3 approximation of $f(0) = \frac{1}{2}$ for $f(x) = \frac{1}{1 + e^x} = \frac{1}{2} - \frac{x}{4} + \frac{x^3}{48} + O(x^5)$. Indeed, $\frac{33}{76} \approx 0.434$ is slightly better than $A\left(\frac{1}{2}\right)$ and $A\left(\frac{1}{3}\right)$.