3 Extrapolation methods

The idea of the extrapolation methods is as follows: If it is possible to express the error of some numerical approximation as a power series of some parameter (typically a stepsize h), this information can be used to systematically cancel the lowest order error terms, and thereby obtain a higher order approximation.

Let us start with an example:

Example 3.1. *Consider the central difference formula for approximating the derivative of a function f at some point x_0,*

$$\frac{f(x_0+h)-f(x_0-h)}{2h} \approx f'(x_0).$$

By Taylor expansion of $f(x_0+h)$ and $f(x_0-h)$ around x_0 the the error can be expressed as a power series in h:

$$\frac{f(x_0+h)-f(x_0-h)}{2h} = \frac{1}{2h}\sum_{p=0}^{\infty}\left(1-(-1)^p\right)\frac{h^p}{p!}f^{(p)}(x_0) = f'(x_0)+\sum_{k=1}^{\infty}\frac{h^{(2k)}}{(2k+1)!}f^{(2k+1)}(x_0).$$

Assume we want to compute some quantity Q with some algorithm $F(h)$ where h is a method dependent parameter. Further, assume that we have an error expansion given by

$$F(h) = Q + C_1h^2 + C_2h^4 + C_3h^6 + \cdots = Q + \sum_{k=1}^{\infty} C_kh^{2k}. \tag{7}$$

in which the constants C_k depends on the problem, *but not on h*. In the example above

$$F(h) = \frac{f(x_0+h)-f(x_0-h)}{2h}, \qquad Q = f'(x_0) \qquad \text{and} \qquad C_k = \frac{f^{(2k+1)}(x_0)}{(2k+1)!}. \tag{8}$$

The idea is to compute $F(h)$ for different values of h, and use this information to systematic eliminate the error terms, and thereby obtain a higher order methods. We can use half the stepsize to get

$$F\left(\tfrac{h}{2}\right) = Q + C_1\left(\tfrac{h}{2}\right)^2 + C_2\left(\tfrac{h}{2}\right)^4 + C_3\left(\tfrac{h}{2}\right)^6 + \cdots \tag{9}$$

By subtracting (7) from 4 times this equation, and divide the whole thing by 3 we get

$$\frac{4F\left(\frac{h}{2}\right)-F(h)}{3} = Q + C_2^1h^4 + C_3^1h^6 + \cdots$$

where $C_k^1 = (4^{-k+1}-1)/3\cdot C_k$. This can be done more general: Assume

$$\begin{aligned} T_{j-1,k} &= Q + D_kh^{2k} + D_{k+1}h^{2(k+1)} + D_{k+2}h^{2(k+2)} + \cdots \\ T_{j,k} &= Q + D_k\left(\tfrac{h}{2}\right)^{2k} + D_{k+1}\left(\tfrac{h}{2}\right)^{2(k+1)} + D_{k+2}\left(\tfrac{h}{2}\right)^{2(k+2)} + \cdots \end{aligned}$$

for some constants D_k, independent of h. And by the same procedure as above, the h^{2k}-term can be elminated such that

$$T_{j,k+1} = \frac{4^kT_{j,k}-T_{j-1,k}}{4^k-1} = Q + D_{k+1}^1h^{2(k+1)} + D_{k+2}^1h^{2(k+2)} + \cdots. \tag{10}$$

This gives a systematic way of constructing higher order schemes from a lower order one,

$$T_{j,1} = F\left(\frac{h}{2^{j-1}}\right), \qquad j = 1, 2, \ldots \tag{11a}$$

$$T_{j,k+1} = \frac{4^k T_{j,k} - T_{j-1,k}}{4^k - 1}, \qquad k = 1, 2, \ldots, j-1. \tag{11b}$$

resulting in a table

$$\begin{array}{lllll} F(h) = T_{1,1} & & & & \\ F(h/2) = T_{2,1} & T_{2,2} & & & \\ F(h/4) = T_{3,1} & T_{3,2} & T_{3,3} & & \\ F(h/8) = T_{4,1} & T_{4,2} & T_{4,3} & T_{4,4} & \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{array}$$

Example 3.2. *Let $f(x) = \sin(x)$ and use (7) and extrapolation to find an approximation to $f'(x_0)$ for $x_0 = 0.5$. Starting with $h = 0.1$, the first four rows in the table will be*

0.876120655431924			
0.877216948194290	0.877582379115078		
0.877491149896850	0.877582550464370	0.877582561887655	
0.877559708356366	0.877582561176204	0.877582561890327	0.877582561890369

with errors $E_{i,i} = Q - T_{i,i}$:

$$E_{1,1} = 1.46 \cdot 10^{-3}, \qquad E_{2,2} = 1.83 \cdot 10^{-7}, \qquad E_{3,3} = 2.72 \cdot 10^{-12}, \qquad E_{4,4} = 3.55 \cdot 10^{-15}.$$

This has been computed by the matlab code `extrapolation.m`.

Other examples satisfying (7) and for which the algorithm (11) are applicable are:

- The trapezoidal rule for integrals $\int_a^b f(x)dx$

$$T(h) = h\left(\frac{1}{2}f(x_0) + \sum_{j=1}^{N-1} f(x_i) + \frac{1}{2}f(x_N)\right)$$

  where $h = (b-a)/n$ for some n, and $x_i = a + ih$, $i = 0, \cdots, n$. This algorithm is better known as *Romberg integration*.

- The solution of an ODE by the implicit midpoint rule from t_0 to t_{end}

$$k_1 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right), \qquad y_{n+1} = y_n + hk_1.$$

  where $F(h) = y_N$, the numerical solution at t_{end} using the stepsize $h = (t_{end} - t_0)/N$, and $Q = y(t_{end})$. For the extrapolation strategy to work in this case, the nonlinear equations has to be solved sufficiently accurate.

The strategy proposed here can be modified to other situation where the error can be expressed as power series of h, e.g. if

$$F(h) = Q + C_1 h + C_2 h^2 + C_3 h^3 + C_4 h^4 + \ldots$$

It is also possible to use other sequences of stepsizes h.

But in all cases, the constants C_k depends on some derivatives of the underlying problem, so it will only work if this problem is sufficiently differentiable.