Lecture 30: Linear Multistep Methods: Truncation Error

5.2. Linear multistep methods.

One-step methods construct an approximate solution $x_{k+1} \approx x(t_{k+1})$ using only one previous approximation, x_k. This approach enjoys the virtue that the step size h can be changed at every iteration, if desired, thus providing a mechanism for error control. This flexibility comes at a price: For each order of accuracy in the truncation error, each step must perform at least one new evaluation of the derivative function $f(t, x)$. This might not sound particularly onerous, but in many practical problems, f evaluations are terribly time-consuming. A classic example is the N-body problem, which arises in models ranging from molecular dynamics to galactic evolution. Such models give rise to N coupled second order nonlinear differential equations, where the function f measures the forces between the N different particles. An evaluation of f requires $O(N^2)$ arithmetic operations to compute, costly indeed when N is in the millions. Every f evaluation counts.[†]

One could potentially improve this situation by re-using f evaluations from previous steps of the ODE integrator, rather than always requiring f to be evaluated at multiple new (t, x) values at each iteration (as is the case, for example, with higher order Runge–Kutta methods). Consider the method

$$x_{k+1} = x_k + h(\tfrac{3}{2} f(t_k, x_k) - \tfrac{1}{2} f(t_{k-1}, x_{k-1})),$$

where h is the step size, $h = t_{k+1} - t_k = t_k - t_{k-1}$. Here x_{k+1} is determined from the two previous values, x_k and x_{k-1}. Unlike Runge–Kutta methods, f is not evaluated at points between t_k and t_{k+1}. Rather, each iteration requires only one new f evaluation, since $f(t_{k-1}, x_{k-1})$ would have been computed already at the previous step. Hence this method has roughly the same computational requirements as Euler's method, though soon we will see that its truncation error is $O(h^2)$. The Heun and midpoint rules attained this same truncation error, but required *two* new f evaluations at each step.

Several drawbacks to this new scheme are evident: it is difficult to adjust the step size, and values for both x_0 and x_1 are needed before starting the method. The former concern can be addressed in practice through interpolation techniques. To handle the latter concern, initial data can be generated using a one-step method with small step size h. In some applications, including some problems in celestial mechanics, an asymptotic series expansion of the solution, accurate near $t \approx t_0$, can provide suitable initial data.

5.2.1. General linear multistep methods.

This section considers a general class of integrators known as *linear multistep methods*.[‡]

Definition. A general *m-step linear multistep method* has the form

$$\sum_{j=0}^{m} \alpha_j x_{k+j} = h \sum_{j=0}^{m} \beta_j f(t_{k+j}, x_{k+j}),$$

with $\alpha_m \neq 0$. If $\beta_m \neq 0$, then the formula for x_{k+m} involves x_{k+m} on the right hand side, so the method is *implicit*; otherwise, the method is *explicit*. A final convention requires $|\alpha_0| + |\beta_0| \neq 0$,

[†] A landmark improvement to this N^2 approach, the *fast multipole method*, was developed by Leslie Greengard and Vladimir Rokhlin in the late 1980s. Rokhlin is a CAAM alumnus (Ph.D.), and is now a professor at Yale.

[‡] These notes on linear multistep methods draw heavily from the excellent presentation in Endre Süli and David F. Mayers, *Numerical Analysis: An Introduction*, Cambridge University Press, 2003.

for if $\alpha_0 = \beta_0 = 0$, then we actually have an $m-1$ step method masquerading as a m-step method. As f is only evaluated at (t_j, x_j), we adopt the abbreviation

$$f_j = f(t_j, x_j).$$

Most Runge–Kutta methods, though one-step methods, *are not* multistep methods. Euler's method is an example of a one-step method that also fits this multistep template. Here are a few examples of linear multistep methods:

Euler's method:	$x_{k+1} - x_k = hf_k$	$\alpha_0 = -1,\ \alpha_1 = 1,\ \beta_0 = 1,\ \beta_1 = 0.$
Trapezoid rule:	$x_{k+1} - x_k = \frac{h}{2}(f_k + f_{k+1})$	$\alpha_0 = -1,\ \alpha_1 = 1,\ \beta_0 = \frac{1}{2},\ \beta_1 = \frac{1}{2}.$
Adams–Bashforth:	$x_{k+2} - x_{k+1} = \frac{h}{2}(3f_{k+1} - f_k)$	$\alpha_0 = 0,\ \alpha_1 = -1,\ \alpha_2 = 1,$ $\beta_0 = -\frac{1}{2},\ \beta_1 = \frac{3}{2},\ \beta_2 = 0.$

The 'Adams–Bashforth' method presented above is the 2-step example of a broader class of *Adams–Bashforth* formulas. The 4-step Adams–Bashforth method takes the form

$$x_{k+4} = x_{k+3} + \frac{h}{24}\Big(55f_{k+3} - 59f_{k+2} + 37f_{k+1} - 9f_k\Big).$$

Here $\alpha_0 = \alpha_1 = \alpha_2 = 0$, $\alpha_3 = -1$, $\alpha_4 = 1$; $\beta_0 = -\frac{9}{24}$, $\beta_1 = \frac{37}{24}$, $\beta_2 = -\frac{59}{24}$, $\beta_3 = \frac{55}{24}$, and $\beta_4 = 0$.

The *Adams–Moulton* methods are a parallel class of *implicit* formulas. The 3-step version of this method is

$$x_{k+3} = x_{k+2} + \frac{h}{24}\Big(9f_{k+3} + 19f_{k+2} - 5f_{k+1} + f_k\Big).$$

Here $\alpha_0 = \alpha_1 = 0$, $\alpha_2 = -1$, $\alpha_3 = 1$; $\beta_0 = \frac{1}{24}$, $\beta_1 = -\frac{5}{24}$, $\beta_2 = \frac{19}{24}$, and $\beta_3 = \frac{9}{24}$.

5.2.2. Truncation error for multistep methods.

Recall that the truncation error of one-step methods of the form $x_{k+1} = x_k + h\Phi(t_k, x_k; h)$ was given by

$$T_k = \frac{x(t_{k+1}) - x(t_k)}{h} - \Phi(t_k, x_k; h).$$

With general linear multistep methods is associated an analogous formula, based on substituting the exact solution $x(t_k)$ for the approximation x_k, and rearranging terms:

$$T_k = \frac{\sum_{j=0}^m \Big[\alpha_j\, x(t_{k+j}) - h\,\beta_j\, f(t_{k+j}, x(t_{k+j}))\Big]}{h\sum_{j=0}^m \beta_j}.$$

(The $\sum_{j=0}^m \beta_j$ term in the denominator is a normalization term; if it were absent, then multiplying the entire multistep formula by a constant would alter the truncation error, but the not the iterates x_j.) In order to get a simple form of the truncation error, we turn to Taylor series:

$$\begin{aligned}
x(t_{k+1}) = \quad x(t_k + h) \quad &= \quad x(t_k) + \quad hx'(t_k) + \quad \tfrac{h^2}{2!}x''(t_k) + \quad \tfrac{h^3}{3!}x'''(t_k) + \quad \tfrac{h^4}{4!}x^{(4)}(t_k) + \cdots \\
x(t_{k+2}) = \quad x(t_k + 2h) \quad &= \quad x(t_k) + \quad 2hx'(t_k) + \quad \tfrac{2^2h^2}{2!}x''(t_k) + \quad \tfrac{2^3h^3}{3!}x'''(t_k) + \quad \tfrac{2^4h^4}{4!}x^{(4)}(t_k) + \cdots \\
x(t_{k+3}) = \quad x(t_k + 3h) \quad &= \quad x(t_k) + \quad 3hx'(t_k) + \quad \tfrac{3^2h^2}{2!}x''(t_k) + \quad \tfrac{3^3h^3}{3!}x'''(t_k) + \quad \tfrac{3^4h^4}{4!}x^{(4)}(t_k) + \cdots \\
&\ \ \vdots \\
x(t_{k+m}) = x(t_k + mh) \quad &= \quad x(t_k) + \quad mhx'(t_k) + \quad \tfrac{m^2h^2}{2!}x''(t_k) + \quad \tfrac{m^3h^3}{3!}x'''(t_k) + \quad \tfrac{m^4h^4}{4!}x^{(4)}(t_k) + \cdots
\end{aligned}$$

and also

$$\begin{aligned}
f(t_{k+1},x(t_{k+1})) = x'(t_k+h) &= x'(t_k) + hx''(t_k) + \tfrac{h^2}{2!}x'''(t_k) + \tfrac{h^3}{3!}x^{(4)}(t_k) + \cdots \\
f(t_{k+2},x(t_{k+2})) = x'(t_k+2h) &= x'(t_k) + 2hx''(t_k) + \tfrac{2^2h^2}{2!}x'''(t_k) + \tfrac{2^3h^3}{3!}x^{(4)}(t_k) + \cdots \\
f(t_{k+3},x(t_{k+3})) = x'(t_k+3h) &= x'(t_k) + 3hx''(t_k) + \tfrac{3^2h^2}{2!}x'''(t_k) + \tfrac{3^3h^3}{3!}x^{(4)}(t_k) + \cdots \\
&\vdots \\
f(t_{k+m},x(t_{k+m})) = x'(t_k+mh) &= x'(t_k) + mhx''(t_k) + \tfrac{m^2h^2}{2!}x'''(t_k) + \tfrac{m^3h^3}{3!}x^{(4)}(t_k) + \cdots.
\end{aligned}$$

Substituting these expansions into the expression for T_k (eventually) yields a convenient formula:

$$\begin{aligned}
\Big(\sum_{j=0}^m \beta_j\Big) T_k &= \frac{\sum_{j=0}^m \Big[\alpha_j\, x(t_{k+j}) - h\,\beta_j\, f(t_{k+j},x(t_{k+j}))\Big]}{h} \\
&= h^{-1}\Big[\sum_{j=0}^m \alpha_j\Big] x(t_k) + \sum_{\ell=0}^\infty h^\ell \Big[\sum_{j=0}^m \Big(\alpha_j \frac{1}{(\ell+1)!} j^{\ell+1} - \beta_j \frac{1}{\ell!} j^\ell\Big) x^{(\ell+1)}(t_k)\Big] \\
&= \frac{1}{h}\Big[\sum_{j=0}^m \alpha_j\Big] x(t_k) + \Big[\sum_{j=0}^m j\alpha_j - \sum_{j=0}^m \beta_j\Big] x'(t_k) \\
&\quad + h\Big[\sum_{j=0}^m \frac{j^2}{2}\alpha_j - \sum_{j=0}^m j\beta_j\Big] x''(t_k) \\
&\quad + h^2\Big[\sum_{j=0}^m \frac{j^3}{6}\alpha_j - \sum_{j=0}^m \frac{j^2}{2}\beta_j\Big] x'''(t_k) \\
&\quad + h^3\Big[\sum_{j=0}^m \frac{j^4}{24}\alpha_j - \sum_{j=0}^m \frac{j^3}{6}\beta_j\Big] x^{(4)}(t_k) + \cdots.
\end{aligned}$$

In particular, the coefficient of the h^ℓ term is simply

$$\sum_{j=0}^m \frac{j^{\ell+1}}{(\ell+1)!}\alpha_j - \sum_{j=0}^m \frac{j^\ell}{\ell!}\beta_j$$

for all nonnegative integers ℓ.

Definition. A linear multistep method is *consistent* if $T_k \to 0$ as $h \to 0$.

A condition for consistency is obvious from the formula for T_k:

Theorem. An m-step linear multistep method of the form

$$\sum_{j=0}^m \alpha_j x_{k+j} = h\sum_{j=0}^m \beta_j f_{k+j}$$

is consistent if and only if

$$\sum_{j=0}^{m} \alpha_j = 0 \qquad \text{and} \qquad \sum_{j=0}^{m} j\alpha_j = \sum_{j=0}^{m} \beta_j.$$

Definition. A linear multistep method is *order-p accurate* if $T_k = O(h^p)$ as $h \to 0$.

Theorem. An m-step linear multistep method is order-p accurate if and only if it is consistent and

$$\sum_{j=0}^{m} \frac{j^{\ell+1}}{(\ell+1)!}\alpha_j = \sum_{j=0}^{m} \frac{j^{\ell}}{\ell!}\beta_j$$

for all $\ell = 1, \ldots, p-1$.

We now compute the order of accuracy of some multistep methods discussed earlier.

Example (Euler's method). $\alpha_0 = -1$, $\alpha_1 = 1$, $\beta_0 = 1$, $\beta_1 = 0$.
Clearly $\alpha_0 + \alpha_1 = -1 + 1 = 0$ and $(0\alpha_0 + 1\alpha_1) - (\beta_0 + \beta_1) = 0$. When we analyzed this algorithm as a one-step method, we saw it had $T_k = O(h)$. We expect the same result from this multistep analysis. Indeed,

$$\left(\tfrac{1}{2}0^2\alpha_0 + \tfrac{1}{2}1^2\alpha_1\right) - \left(0\beta_0 + 1\beta_1\right) = \tfrac{1}{2} \neq 0.$$

Thus, $T_k = O(h)$.

Example (Trapezoid rule). $\alpha_0 = -1$, $\alpha_1 = 1$, $\beta_0 = \frac{1}{2}$, $\beta_1 = \frac{1}{2}$.
Again, consistency is easy to verify: $\alpha_0 + \alpha_1 = -1 + 1 = 0$ and $(0\alpha_0 + 1\alpha_1) - (\beta_0 + \beta_1) = 1 - 1 = 0$.
Furthermore,

$$\left(\tfrac{1}{2}0^2\alpha_0 + \tfrac{1}{2}1^2\alpha_1\right) - (0\beta_0 + 1\beta_1) = \frac{1}{2} - \frac{1}{2} = 0,$$

so $T_k = O(h^2)$, but

$$\left(\tfrac{1}{6}0^3\alpha_0 + \tfrac{1}{6}1^3\alpha_1\right) - \left(\tfrac{1}{2}0^2\beta_0 + \tfrac{1}{2}1^2\beta_1\right) = \frac{1}{6} - \frac{1}{4} \neq 0,$$

so the trapezoid rule is not third order accurate.

Example (2-step Adams–Bashforth). $\alpha_0 = 0$, $\alpha_1 = -1$, $\alpha_2 = 1$, $\beta_0 = -\frac{1}{2}$, $\beta_1 = \frac{3}{2}$, $\beta_2 = 0$.
We can now verify that this method presented earlier in this lecture lives up to its name. Consistency follows: $\alpha_0 + \alpha_1 + \alpha_2 = 0 - 1 + 1 = 0$; $(0\alpha_0 + 1\alpha_1 + 2\alpha_2) - (\beta_0 + \beta_1) = 1 - 1 = 0$. The second order condition is also satisfied,

$$\left(\tfrac{1}{2}0^2\alpha_0 + \tfrac{1}{2}1^2\alpha_1 + \tfrac{1}{2}2^2\alpha_2\right) - \left(0\beta_0 + 1\beta_1\right) = \tfrac{3}{2} - \tfrac{3}{2} = 0,$$

but not the third order,

$$\left(\tfrac{1}{6}0^3\alpha_0 + \tfrac{1}{6}1^3\alpha_1 + \tfrac{1}{6}2^3\alpha_2\right) - \left(\tfrac{1}{2}0^2\beta_0 + \tfrac{1}{2}1^2\beta_1\right) = \tfrac{7}{6} - \tfrac{3}{4} \neq 0.$$

Example (4-step Adams–Bashforth). $\alpha_0 = \alpha_1 = \alpha_2 = 0$, $\alpha_3 = -1$, $\alpha_4 = 1$; $\beta_0 = \frac{9}{24}$, $\beta_1 = \frac{37}{24}$, $\beta_2 = -\frac{59}{24}$, $\beta_3 = \frac{55}{24}$, $\beta_4 = 0$.
Consistency holds, since $\sum \alpha_j = -1 + 1 = 0$ and

$$\sum_{j=0}^{4} j\alpha_j - \sum_{j=0}^{4} \beta_j = \left(3(-1) + 4(1)\right) - \left(\tfrac{9}{24} + \tfrac{37}{24} - \tfrac{47}{24} + \tfrac{55}{24}\right) = 1 - 1 = 0.$$

The coefficients of h, h^2, and h^3 in the expansion for T_k all vanish:

$$\sum_{j=0}^{4} \tfrac{1}{2} j^2 \alpha_j - \sum_{j=0}^{4} j\beta_j = \Big(\tfrac{3^2}{2}(-1) + \tfrac{4^2}{2}(1)\Big) - \Big(0(-\tfrac{9}{24}) + 1(\tfrac{37}{24}) + 2(-\tfrac{59}{24}) + 3(\tfrac{55}{24})\Big) = \tfrac{7}{2} - \tfrac{84}{24} = 0;$$

$$\sum_{j=0}^{4} \tfrac{1}{6} j^3 \alpha_j - \sum_{j=0}^{4} \tfrac{1}{2} j^2 \beta_j = \Big(\tfrac{3^3}{6}(-1) + \tfrac{4^3}{6}(1)\Big) - \Big(\tfrac{1^2}{2}(\tfrac{37}{24}) + \tfrac{2^2}{2}(-\tfrac{59}{24}) + \tfrac{3^2}{2}(\tfrac{55}{24})\Big) = \tfrac{37}{6} - \frac{148}{24} = 0;$$

$$\sum_{j=0}^{4} \tfrac{1}{24} j^4 \alpha_j - \sum_{j=0}^{4} \tfrac{1}{6} j^3 \beta_j = \Big(\tfrac{3^4}{24}(-1) + \tfrac{4^4}{24}(1)\Big) - \Big(\tfrac{1^3}{6}(\tfrac{37}{24}) + \tfrac{2^3}{6}(-\tfrac{59}{24}) + \tfrac{3^3}{6}(\tfrac{55}{24})\Big) = \tfrac{175}{24} - \tfrac{1050}{144} = 0.$$

However, the $O(h^4)$ term is not eliminated:

$$\sum_{j=0}^{4} \tfrac{1}{120} j^5 \alpha_j - \sum_{j=0}^{4} \tfrac{1}{24} j^4 \beta_j = \Big(\tfrac{3^5}{120}(-1) + \tfrac{4^5}{120}(1)\Big) - \Big(\tfrac{1^4}{24}(\tfrac{37}{24}) + \tfrac{2^4}{24}(-\tfrac{59}{24}) + \tfrac{3^4}{24}(\tfrac{55}{24})\Big) = \tfrac{1267}{120} - \tfrac{887}{144} \neq 0.$$

A similar computation establishes fourth-order accuracy for the 4-step Adams–Moulton formula

$$x_{k+3} = x_{k+2} + \tfrac{1}{24} h\Big(9f_{k+3} + 19f_{k+2} - 5f_{k+1} + f_k\Big).$$

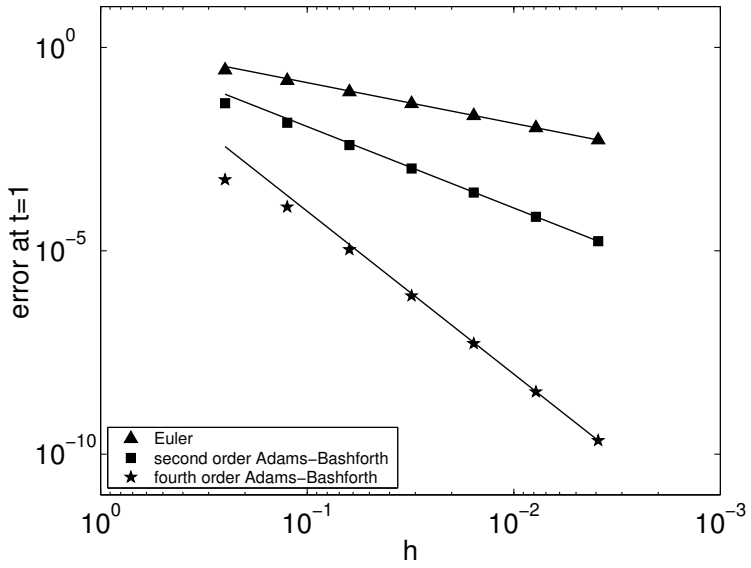
Computational verification. In the last lecture, we proved that *consistency implies convergence* for one-step methods. Essentially, provided the differential equation is sufficiently well-behaved (in the sense of Picard's Theorem), then the numerical solution produced by a consistent one-step method on the fixed interval $[t_0, t_{\text{final}}]$ will converge to the true solution as $h \to 0$. Of course, this is a key property that we hope is shared by multistep methods.

Whether this is true for general linear multistep methods is the subject of the next lecture. For now, we merely present some computational evidence that, for certain methods, the global error at t_{final} behaves in the same manner as the truncation error.

Consider the model problem $x'(t) = x(t)$ for $t \in [0, 1]$ with $x(0) = x_0 = 1$, which has the exact solution is $x(t) = \mathrm{e}^t$. We shall approximate this solution using Euler's method, the second-order Adams–Bashforth formula, and the fourth-order Adams–Bashforth formula. The latter two methods require data not only at $t = 0$, but also at several additional values, $t = h$, $t = 2h$, and $t = 3h$. For this simple experiment, we can use the value of the exact solution, $x_1 = x(t_1)$, $x_2 = x(t_2)$, and $x_3 = x(t_3)$.

The plot below reports the results of this exercise, showing the absolute error at t_{final} as a function of h. (Note the log-log axis here.) Near the data points, we have drawn lines indicating pure h, h^2, and h^4 convergence. For this particular problem, the global error shrinks at the same rate as the truncation error.[§] Will this be true in more general settings? That is the subject of the next lecture.

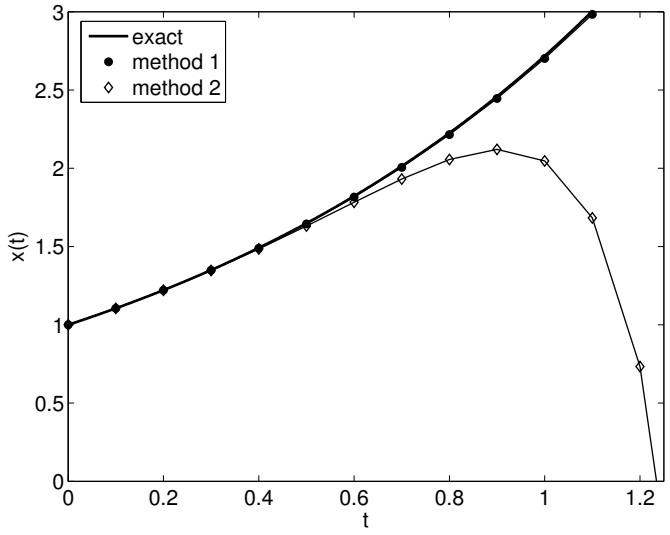
[§]MATLAB programming note: The horizontal axis of this `loglog` plot would, by default, run from smallest h to largest h, (i.e., 10^{-3} to 10^{0}). But since we are interested in convergence as $h \to 0$, it is intuitively appealing for h to get smaller as we read from left to right. This is accomplished via: `set(gca,'XDir','reverse')`.



We close by offering evidence that there is more to the analysis of linear multistep methods than truncation error. Here are two explicit methods that are both second order:

$$x_{k+2} - \tfrac{3}{2}x_{k+1} + \tfrac{1}{2}x_k = h(\tfrac{5}{4}f_{k+1} - \tfrac{3}{4}f_k)$$
$$x_{k+2} - 3x_{k+1} + 2x_k = h(\tfrac{1}{2}f_{k+1} - \tfrac{3}{2}f_k).$$

We apply these methods to the model problem $x'(t) = x(t)$ with $x(0) = 1$, with exact initial data $x_0 = 1$ and $x_1 = e^h$. The results of these two methods are shown below.



The first method tracks the exact solution $x(t) = e^t$ very nicely. The second method, however, shows a disturbing property: while it matches up quite well for the initial steps, it soon starts to fall far from the solution. Why does this second-order method do so poorly for such a simple problem? Does this reveal a general problem with linear multistep methods? If not, how do we identify such ill-mannered methods?