Numerical solution of ordinary differential equations: multistep methods

Recall that by integrating both sides of the ODE $u'(t) = f\big(t, u(t)\big)$ from $t = t_n$ to $t = t_{n+1}$, we got

$$u(t_{n+1}) - u(t_n) = \int_{t_n}^{t_{n+1}} f\big(t, u(t)\big)\, dt$$

and used this to derive several numerical methods in the form of

$$u_{n+1} = u_n + \big\{\text{approximation of area under the } F(t) \equiv f\big(t, u(t)\big) \text{ curve between } t = t_n \text{ and } t = t_{n+1}\big\}$$

or

$$(\dagger) \qquad u_{n+1} = u_n + \underbrace{\int_{t_n}^{t_n+h} \{\text{approximation of the } F(t) \text{ curve}\}\, dt}_{(\star)}$$

Approximating $F(t)$ by a straight line gave the Euler, backward Euler, midpoint, trapezoidal and Heun methods. Approximating $F(t)$ by a parabola through the three points $(t_n, F(t_n)), (t_n + \frac{h}{2}, F(t_n + \frac{h}{2}))$, $(t_n + h, F(t_n + h))$ and then integrating this quadratic interpolating polynomial (i.e. Simpson's rule) formed the basis of the 4-stage Runge–Kutta method.

So far, all the methods we have considered are *one-step* methods in that the computation of u_{n+1} depends only on knowledge of the most recently computed point u_n and not on any of the points before that (u_{n-1}, u_{n-2} etc.).

Adams–Bashforth methods

Approximate $F(t) \equiv f\big(t, u(t)\big)$ by a polynomial of degree k (with $k+1$ coefficients) fitted using the $k+1$ previously computed points, up to and including (t_n, u_n); that is, the polynomial should go through the points

$$\left\{\begin{array}{c} \big(t_n,\ f(t_n, u_n)\big) \\ \big(t_{n-1},\ f(t_{n-1}, u_{n-1})\big) \\ \vdots \\ \big(t_{n-k},\ f(t_{n-k}, u_{n-k})\big) \end{array}\right. \quad \text{or} \quad \left\{\begin{array}{c} \big(t_n,\ f_n\big) \\ \big(t_{n-1},\ f_{n-1}\big) \\ \vdots \\ \big(t_{n-k},\ f_{n-k}\big) \end{array}\right. \quad \text{where we have used the shorthand } f_j \equiv f(t_j,\, u_j)$$

For example, using a degree-1 polynomial (straight line) $At + B$, we require

$$\left\{\begin{array}{l} At_n + B = f_n \\ At_{n-1} + B = f_{n-1} \end{array}\right.$$

Solve these two simultaneous equations for A, B in terms of t_n, t_{n-1}, f_n and f_{n-1}, which are all already known; then compute $\int_{t_n}^{t_n+h} At + B\, dt$ and plug this in for $(\star)$ in the formula $(\dagger)$. This yields

$$u_{n+1} = u_n + \frac{1}{2}h\,\big(3f_n - f_{n-1}\big),$$

which is a 2-step method, because the computation of u_{n+1} depends on knowledge of two previously computed points, u_n and u_{n-1}.
By fitting a degree-2 (quadratic) polynomial we obtain a 3-step Adams–Bashforth method:

$$u_{n+1} = u_n + \frac{h}{12}\big(23f_n - 16f_{n-1} + 5f_{n-2}\big)$$

By fitting a degree-3 (cubic) polynomial we obtain a 4-step Adams–Bashforth method:

$$u_{n+1} = u_n + \frac{h}{24}\big(55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3}\big)$$

Adams–Moulton methods

Approximate $F(t) \equiv f\big(t, u(t)\big)$ by a polynomial of degree k (with $k+1$ coefficients) fitted using k previously computed points and 1 future (unknown) point; that is, the polynomial should go through the points

$$\begin{cases} \big(t_{n+1},\ f(t_{n+1}, u_{n+1})\big) \\ \big(t_n,\ f(t_n, u_n)\big) \\ \big(t_{n-1},\ f(t_{n-1}, u_{n-1})\big) \\ \qquad\vdots \\ \big(t_{n-k},\ f(t_{n-k}, u_{n-k})\big) \end{cases} \qquad \text{or} \qquad \begin{cases} \big(t_{n+1},\ f_{n+1}\big) \\ \big(t_n,\ f_n\big) \\ \big(t_{n-1},\ f_{n-1}\big) \\ \qquad\vdots \\ \big(t_{n-k+1},\ f_{n-k+1}\big) \end{cases}$$

For example, using a degree-1 polynomial $At+B$, we require

$$\begin{cases} At_{n+1} + B = f_{n+1} \\ At_n + B = f_n \end{cases}$$

Solve these two simultaneous equations for A, B in terms of t_n, t_{n+1}, f_n, which are already known, and $f_{n+1} \equiv f(t_{n+1}, y_{n+1})$, which is not known; then compute $\int_{t_n}^{t_n+h} At + B\,dt$ and plug this in for $(\star)$ in the formula $(\dagger)$ to get

$$u_{n+1} = u_n + \frac{1}{2}h\,\big(f_n + f_{n+1}\big), \qquad \text{where } f_{n+1} = f(t_n + h,\ u_{n+1}),$$

which is actually a one-step method.
By fitting a degree-2 (quadratic) polynomial we obtain a 2-step Adams–Moulton method:

$$u_{n+1} = u_n + \frac{h}{12}\,\big(5f_{n+1} + 8f_n - f_{n-1}\big)$$

By fitting a degree-3 (cubic) polynomial we obtain a 3-step Adams–Moulton method:

$$u_{n+1} = u_n + \frac{h}{24}\,\big(9f_{n+1} + 19f_n - 5f_{n-1} + f_{n-2}\big)$$

Note that the usage of one future (unknown) point to fit the polynomial will always result in a u_{n+1} on the right-hand side of the formula; so Adams–Moulton methods are all *implicit* methods.

Adams predictor–corrector methods

In the same way as we converted the implicit Crank–Nicolson (trapezoidal) method into an explicit predictor–corrector method, namely the Heun method, we can convert an Adams–Moulton method into an explicit predictor–corrector method by replacing the "u_{n+1}" in the right-hand side of the formula by an approximation that comes from an explicit method.

For instance, we can use an Adams–Bashforth method as the "predictor" to make an "intermediate" approximation of u_{n+1}, and then substitute this intermediate approximation for "u_{n+1}" in the right-hand side of an Adams–Moulton formula ("corrector").

Initializing multistep methods

With *multistep* methods, such as most of the Adams methods above, to compute u_{n+1} we need information about more than just one previously computed value. Therefore, to initialize a multistep algorithm, we also need more than just the initial condition (t_0, u_0).

Usually, we compute a few points in addition to the initial u_0 by using a one-step method, and use these points to get the multistep method started. For example, with a 3-step method we need to compute two points in addition to u_0 before entering the main time-stepping loop.

Note that which one-step method you choose to start off from can affect the overall order of the algorithm.