Linear Multistep Methods: Adams and BDF Methods

Varun Shankar

February 4, 2019

1 Introduction

Thus far, we have discussed polynomial interpolation and its application to generating FD and quadrature formulae. Now, we will apply these FD and quadrature formulae (in fairly simple form) to the problem of time integration. More formally, consider an initial value problem of the form

$$\frac{dy}{dt} = f(t, y), \tag{1}$$

$$y(t_0) = y_0. \tag{2}$$

Let the variable t here represent time. The goal is to solve this ODE using only samples of the function f. Our first step will be to divide up the time domain $[t_0, T]$ into a *grid*. We will never really form and store this 1D grid, but it is a conceptually useful tool. Let a typical discrete time level on this grid be t_n, where $t_n = t_0 + n\Delta t$, where n is the (integer) number of steps taken from the initial time t_0, and Δt is the spacing between discrete time levels on this grid; Δt is called the "timestep", and it is common to use the symbol k to represent this.

Now, consider solving the ODE over the interval $[t_n, t_{n+1}]$, where $t_{n+1} = t_n + \Delta t$. To do so, we merely need to integrate both sides of the ODE; this yields

$$\int_{t_n}^{t_{n+1}} \frac{dy}{dt} dt = \int_{t_n}^{t_{n+1}} f(t, y) dt. \tag{3}$$

We can apply the Fundamental Theorem of Calculus to evaluate the integral on the left hand side. This yields

$$y(t_{n+1}) - y(t_n) = \int_{t_n}^{t_{n+1}} f(t, y) dt. \tag{4}$$

Henceforth, we will use superscripts to indicate time levels. This is fairly common. Thus, we have $y^n = y(t_n)$. In this notation, the above equation becomes

$$y^{n+1} = y^n + \int_{t_n}^{t_{n+1}} f(t, y)dt. \tag{5}$$

Note that this is still exact; no approximations have been made thus far. Now, consider the situation where the solution is known at the current time t_n, and must be computed at t_{n+1}. The above equation seems tailor-made for this situation. If f was exactly known, we could easily find y^{n+1}; indeed, we could find y for all time using the techniques from an ODE class! The problem, of course, is that we may only have limited samples of f. The question then becomes: "how do we find an integral of $f(t, y)$ if only samples of f are given?". By now, you should know the answer: quadrature. Specifically, you would fit a polynomial interpolant to those samples of f, and then simply integrate the Lagrange basis!

In this chapter, we will discuss several schemes for approximating the time integral based on polynomial interpolation. As always, the schemes will vary in accuracy and computational cost based on how much information is available to us. For example, we could have samples of f at t_n alone; alternatively, we could have samples of f at multiple time levels t_n through t_{n-s}, where $0 < s < n$ is some integer. In other words, we could have f values at *multiple time-steps*; the resulting numerical time-integration schemes are called Linear Multistep Methods (LMM). We will also discuss an important class of methods that involves using FD formulae rather than quadrature to solve these ODEs; these schemes also fall under the LMM framework, despite their differing philosophy.

2 Two One-Step Methods

Before discussing multistep methods, we will go over 3 one-step methods that also use the polynomial interpolation-integration idea. At least one of them should be familiar to you.

2.1 Forward Euler

For some reason, most introductory textbooks and classes refer to this as "Euler's method". More properly, it is referred to as "Forward Euler". The key idea is to assume that f stays constant over the interval $[t_n, t_{n+1}]$. In other words, we assume the "polynomial" that we fit has $N = 0$. In this case, we get

$$y^{n+1} = y^n + \Delta t f(t_n, y^n) + O(\Delta t^2). \tag{6}$$

One step of Forward Euler incurs a second-order error; this error for one step of a method is called the *local truncation error* (LTE). However, over many steps, the LTE accumulates, reducing the accuracy to first-order *i.e.*, at some fixed time t, Forward Euler will have a global truncation error of $O(\Delta t)$.

2.2 Backward Euler

We can still assume that f stays constant over the interval $[t_n, t_{n+1}]$, but that the value of f at the *end* of the interval is what matters. This gives us Backward Euler. In this case, we get

$$y^{n+1} = y^n + \Delta t f(t_{n+1}, y^{n+1}) + O(\Delta t^2). \tag{7}$$

To find y^{n+1}, we re-arrange the above equation, yielding

$$y^{n+1} - \Delta t f(t_{n+1}, y^{n+1}) = y^n + O(\Delta t^2). \tag{8}$$

If f is a linear function of y, then computing y^{n+1} requires solving a linear equation. For a system of ODEs where we now have a *vector* of unknown functions $\boldsymbol{y}$, we will end up solving a *system* of equations using the techniques you learned in 5610. It is also possible to have a nonlinear system of ODEs. In this scenario, we will need a nonlinear solver, like Newton's method or the Secant method. Again, like Forward Euler, the LTE is $O(\Delta t^2)$, and the global truncation error will be $O(\Delta t)$.

Why do all this if Forward Euler and Backward Euler have the same errors? Well, the issue is *stability*. While both Forward and Backward Euler using a single value of f to approximate the integral in (5), using the value of f at the *end* of the interval results in greater stability for the same value of Δt. We will discuss this in greater detail in part II of this discussion on multistep methods.

In general, a time integration method that uses unknown function values $f(t_{n+1}, y^{n+1})$ is known as an *implicit* method. An *explicit* method uses only known values of f; as a consequence, it is not necessary to solve for y^{n+1}. Computing y^{n+1} in an explicit method will only require an *update* of the y^n values.

We will now discuss several multistep methods that integrate higher-degree polynomials to obtain smaller LTEs and global errors. The explicit multistep methods based on this philosophy are called Adams-Bashforth methods, and the implicit ones are known as Adams-Moulton methods.

We can also generalize (5) to apply to a *system* of ODEs, and talk about solving for that system instead. Thus, we have

$$\boldsymbol{y}^{n+1} = \boldsymbol{y}^n + \int_{t_n}^{t_{n+1}} \boldsymbol{f}(t, \boldsymbol{y}) dt, \tag{9}$$

where $\boldsymbol{y} = [y_1, y_2, y_3, \ldots, y_M]^T$ is a *vector* of unknown functions $y_1 \ldots y_M$. Thus, (9) is really a set of M equations, written explicitly as

$$\begin{bmatrix} y_1^{n+1} \\ y_2^{n+1} \\ \vdots \\ y_M^{n+1} \end{bmatrix} = \begin{bmatrix} y_1^n \\ y_2^n \\ \vdots \\ y_M^n \end{bmatrix} + \begin{bmatrix} \int_{t_n}^{t_{n+1}} f_1(t, y_1) dt \\ \int_{t_n}^{t_{n+1}} f_2(t, y_2) dt \\ \vdots \\ \int_{t_n}^{t_{n+1}} f_M(t, y_M) dt \end{bmatrix}. \tag{10}$$

We will use the vector notation $\boldsymbol{y}$ to refer to these vectors of unknowns.

3 Adams-Bashforth (AB) Methods

The Adams-Bashforth methods are *explicit multistep methods*. This means that they use information from the current and previous time-steps to compute the solution at t_{n+1}. These are typically abbreviated by ABs, where s is the order of the method. For example, AB2 is the second-order Adams-Bashforth method. Let us derive AB2 now.

3.1 Second-order Adams-Bashforth (AB2)

The idea is to obtain a method that has an LTE of $k + 1$; in other words, to obtain AB2, we'll look for an LTE of $O(\delta t^3)$. Recall that our stated strategy is to interpolate f with a Lagrange interpolating polynomial (in *time*). Now, *linear* interpolants in time have an error of $O(\Delta t^2)$; when we integrate the linear interpolant, we will get an error of $O(\Delta t^3)$. Thus, our desired interpolant is a linear interpolant. For a *single* ODE, we may write this as

$$p(t) = f(t_{n-1}, y^{n-1})\ell_0(t) + f(t_n, y^n)\ell_1(t), \tag{11}$$

$$= f(t_{n-1}, y^{n-1})\frac{t - t_n}{t_{n-1} - t_n} + f(t_n, y^n)\frac{t - t_{n-1}}{t_n - t_{n-1}}. \tag{12}$$

We know that $t_n - t_{n-1} = \Delta t$, since this is simply the gap between two time levels. Using this, we may simplify the above to give

$$p(t) = \frac{t_n - t}{\Delta t} f(t_{n-1}, y^{n-1}) + \frac{t - t_{n-1}}{\Delta t} f(t_n, y^n). \tag{13}$$

Substituting this in place of the integral from (5), we get

$$y^{n+1} = y^n + \int_{t_n}^{t_{n+1}} f(t,y)dt, \tag{14}$$

$$= y^n + \int_{t_n}^{t_{n+1}} p(t)dt + O(\Delta t^3), \tag{15}$$

$$= y^n + \int_{t_n}^{t_{n+1}} \left[\frac{t_n - t}{\Delta t} f(t_{n-1}, y^{n-1}) + \frac{t - t_{n-1}}{\Delta t} f(t_n, y^n)\right] dt + O(\Delta t^3), \tag{16}$$

$$= y^n + \left[f(t_{n-1}, y^{n-1}) \left(-\frac{1}{2}\right) \frac{(t_n - t)^2}{\Delta t} + f(t_n, y^n) \frac{(t - t_{n-1})^2}{2\Delta t}\right]_{t_n}^{t_{n+1}} + O(\Delta t^3), \tag{17}$$

$$\implies y^{n+1} = y^n + \Delta t \left[\frac{3}{2} f(t_n, y^n) - \frac{1}{2} f(t_{n-1}, y^{n-1})\right] + O(\Delta t^3). \tag{18}$$

Thus, if we want y^{n+1}, we update y^n using the values of f at time t_n and t_{n-1}. By using polynomial interpolation to generate this formula, we automatically get the LTE; in this case, as desired, it is $O(\Delta t^3)$. This means that when the LTE accumulates over time, the global error will asymptotically tend to $O(\Delta t^2)$. This method is called the AB2 method, or "Adams-Bashforth method of Order 2". AB2 is a two-step method.

Note 1: For a system of ODEs, we have

$$\boldsymbol{y}^{n+1} = \boldsymbol{y}^n + \Delta t \left[\frac{3}{2} \boldsymbol{f}(t_n, \boldsymbol{y}^n) - \frac{1}{2} \boldsymbol{f}(t_{n-1}, \boldsymbol{y}^{n-1})\right] + O(\Delta t^3). \tag{19}$$

Note 2: At this point, you should be at least a little worried about AB2. If it is a two-step method, we always require two time levels. But, what if we are at t_0 and wish to advance our system to t_1? We do not have a t_{-1} available to us, in general. The practical solution to this is to use one-step of either forward or backward Euler; recall that for a single step of either Euler method, the error is exactly $O(\Delta t^2)$, which is the global error of AB2. By using a method with the same order of error, we avoid "polluting" the solution of our multistep method.

Note 3: There is something interesting going on here. Let us analyze the quantity $g = \frac{3}{2} f(t_n, y^n) - \frac{1}{2} f(t_{n-1}, y^{n-1})$ a little more carefully. What does this term signify? To figure out what is going on, we will Taylor expand the $f(t_{n-1}, y^{n-1})$ term in time. Using the shorthand notation $f(t_n, y^n) = f(t_n)$, we have

$$f(t_n - \Delta t) = f(t_n) - \Delta t f'(t_n) + \frac{\Delta t^2}{2!} f''(t_n) - \frac{\Delta t^3}{3!} f'''(t_n) + \dots \tag{20}$$

What is the quantity g then? Using the above Taylor expansion, we get

$$g = \frac{3}{2}f(t_n) - \frac{1}{2}f(t_n - \Delta t) \tag{21}$$

$$= \frac{3}{2}f(t_n) - \frac{1}{2}\left[f(t_n) - \Delta t f'(t_n) + \frac{\Delta t^2}{2!}f''(t_n) - \frac{\Delta t^3}{3!}f'''(t_n) + \ldots\right], \tag{22}$$

$$= f(t_n) + \frac{\Delta t}{2}f'(t_n) - \frac{\Delta t^2}{4}f''(t_n) + \ldots, \tag{23}$$

$$\implies g = f\left(t_n + \frac{\Delta t}{2}\right) + O(\Delta t^2), \tag{24}$$

$$\implies g = f\left(t_{n+\frac{1}{2}}\right) + O(\Delta t^2). \tag{25}$$

This is saying that g is simply a second-order approximation to $f\left(t_{n+\frac{1}{2}}\right)$; more specifically, it is a second-order extrapolation of $f\left(t_{n+\frac{1}{2}}\right)$ from f at t_n and t_{n-1}. This philosophy gives us an alternative way of generating the Adams-Bashforth methods: seek successively higher-order extrapolations from past time levels to approximate the function f at $t_{n+\frac{1}{2}}$. You will not find this rather useful fact in any textbook!

3.2 Higher-order Adams-Bashforth Methods

To generate higher-order AB methods, we simply use more steps and higher-order polynomials. For instance, to generate AB3, we would fit a quadratic polynomial to f at t_n,t_{n-1} and t_{n-2}, integrate that polynomial over $[t_n, t_{n-1}]$, and add the result to y_n. In general, to obtain an order s AB method, we fit and integrate a polynomial of degree $s-1$ to s points in time (not including t_{n+1}). We can use this fact to come up with a very general formula for AB methods.

Let the Lagrange interpolating polynomial be

$$p(t) = \sum_{k=0}^{s-1} f(t_{n-k}, y_{n-k})\ell_k(t). \tag{26}$$

From our approximation of (5), we have

$$y^{n+1} - y^n \approx \int_{t_n}^{t_{n+1}} p(t)dt, \tag{27}$$

$$= \int_{t_n}^{t_{n+1}} \sum_{k=0}^{s-1} f(t_{n-k}, y_{n-k})\ell_k(t)dt, \tag{28}$$

$$= \sum_{k=0}^{s-1} f(t_{n-k}, y_{n-k}) \int_{t_n}^{t_{n+1}} \ell_k(t)dt. \tag{29}$$

Thus the s-order AB scheme can be concisely written as

$$y^{n+1} = y^n + \Delta t \sum_{k=0}^{s-1} b_k f(t_{n-k}, y_{n-k}), \tag{30}$$

where

$$b_k = \frac{1}{\Delta t} \int_{t_n}^{t_{n+1}} \ell_k(t)dt. \tag{31}$$

An AB method has to be at least order 1 to be consistent, much like the FD formulae we saw earlier. In your assignment, you will be asked to use the polynomial approach to generate higher-order AB methods.

4 Adams-Moulton (AM) Methods

AM methods are *implicit* methods; in other words, they use information at t_{n+1} to compute y^{n+1}. Let us derive AM2, the second-order Adams-Moulton method.

Again, like the AB method, we will use the Lagrange interpolating polynomial of degree 1, aka a linear interpolant. However, instead of fitting the interpolant to f at t_n and t_{n-1}, we will fit to f at t_n and t_{n+1}. Proceeding as for the AB2 case, we have

$$p(t) = f(t_{n+1}, y^{n+1})\ell_0(t) + f(t_n, y^n)\ell_1(t), \tag{32}$$

$$= f(t_{n+1}, y^{n+1})\frac{t - t_n}{t_{n+1} - t_n} + f(t_n, y^n)\frac{t - t_{n+1}}{t_n - t_{n+1}}. \tag{33}$$

We know that $t_{n+1} - t_n = \Delta t$. We simplify the above to give

$$p(t) = \frac{t_{n+1} - t}{\Delta t} f(t_{n+1}, y^{n+1}) + \frac{t - t_n}{\Delta t} f(t_n, y^n). \tag{34}$$

Substituting this in place of the integral from (5), integrating and simplifying, we get

$$y^{n+1} = y^n + \Delta t \left[\frac{1}{2} f(t_{n+1}, y^{n+1}) + \frac{1}{2} f(t_n, y^n)\right] + O(\Delta t^3). \tag{35}$$

Notice that the order 2 AM method only requires the use of *one* previous step for the same $O(\Delta t^3)$ LTE; again, the global error is $O(\Delta t^2)$. In general the s order AM method requires $s-1$ steps, despite using the same polynomial degree as an s order AB method. This is the benefit of going implicit. Of course, we now have a linear system to solve if we want to compute y^{n+1}.

Based on the general form of an AB method, we can now also write the general form of an AM method. We only need to adjust the indices so they go up to t_{n+1} on the interpolating polynomial.

$$y^{n+1} = y^n + \Delta t \sum_{k=0}^{s-1} b_k f(t_{n+1-k}, y^{n+1-k}), \tag{36}$$

$$b_k = \frac{1}{\Delta t} \int_{t_n}^{t_{n+1}} \ell_k(t) dt. \tag{37}$$

5 AB-AM Predictor-Corrector Methods

Though AB methods are commonly used, people rarely use AM methods by themselves. Instead, AM methods are used in conjunction with AB methods in a so-called *predictor-corrector pair*.

The idea is simple: take an AB method and an AM method of the same order. Use the AB method to predict y^{n+1}, and call the predicted value $\tilde{y}^{n+1}$; now, use $\tilde{y}^{n+1}$ within the AM method. This way, the AM method is no longer fully implicit, and no solve is required, yet, one hopes, we have a better approximation to y^{n+1} than if we had used AB alone. To demonstrate, consider the AB2-AM2 predictor-corrector pair.

1. First predict y^{n+1} with AB2. This gives us

$$\tilde{y}^{n+1} = y^n + \Delta t \left[\frac{3}{2} f(t_n, y^n) - \frac{1}{2} f(t_{n-1}, y^{n-1})\right] + O(\Delta t^3). \tag{38}$$

2. Next, correct with AM2. This gives

$$y^{n+1} = y^n + \Delta t \left[\frac{1}{2} f(t_{n+1}, \tilde{y}^{n+1}) + \frac{1}{2} f(t_n, y^n)\right] + O(\Delta t^3). \tag{39}$$

An AB-AM predictor-corrector pair can also be used to dynamically alter the time-step size Δt. Basically, we check if $\alpha|y^{n+1} - \tilde{y}^{n+1}| \leq \epsilon$, where ϵ is some

tolerance, and α is some real-valued constant. If we are less than the tolerance, we increase Δt or do nothing; if we are greater than the tolerance, we decrease Δt. This allows us to use large time-steps when possible, and small time-steps when necessary.

6 Backward Differentiation Formula (BDF) Methods

The BDF methods follow a different philosophy from the Adams methods. For this, we return to the differential form of the IVP

$$\frac{dy}{dt} = f(t, y), \tag{40}$$

$$y(t_0) = y_0. \tag{41}$$

Now, we approximate $\frac{dy}{dt}$ directly over the interval $[t_{n+1-s}, t_{n+1}]$, where s is the order of the BDF method. We evaluate f only at t_{n+1}. For illustration, consider BDF2, the second-order Backward Differentiation Formula.

Since we have $s = 2$, we must approximate the time derivative over the interval $[t_{n-1}, t_{n+1}]$. To do so, we fit the Lagrange interpolating polynomial to these points and differentiate it. The polynomial is

$$p(t) = \frac{(t-t_n)(t-t_{n+1})}{(t_{n-1}-t_n)(t_{n-1}-t_{n+1})}y^{n-1} + \frac{(t-t_{n-1})(t-t_{n+1})}{(t_n-t_{n-1})(t_n-t_{n+1})}y^n + \frac{(t-t_{n-1})(t-t_n)}{(t_{n+1}-t_{n-1})(t_{n+1}-t_n)}y^{n+1}. \tag{42}$$

We know that $t_{n-1}-t_n = -\Delta t$, $t_{n-1}-t_{n+1} = -2\Delta t$, $t_n-t_{n-1} = \Delta t$, $t_n-t_{n+1} = -\Delta t$, $t_{n+1}-t_{n-1} = 2\Delta t$ and $t_{n+1}-t_n = \Delta t$. We can plug this in to simplify the above expression. This yields

$$p(t) = \frac{(t-t_n)(t-t_{n+1})}{2\Delta t^2}y^{n-1} - \frac{(t-t_{n-1})(t-t_{n+1})}{\Delta t^2}y^n + \frac{(t-t_{n-1})(t-t_n)}{2\Delta t^2}y^{n+1}. \tag{43}$$

Differentiating this expression with respect to t gives

$$\frac{d}{dt}p(t) = \frac{2t-t_n-t_{n+1}}{2\Delta t^2}y^{n-1} + \frac{2t-t_{n-1}-t_{n+1}}{-\Delta t^2}y^n + \frac{2t-t_{n-1}-t_n}{2\Delta t^2}y^{n+1}. \tag{44}$$

We now need to evaluate $\frac{d}{dt}p(t)$ at $t = t_{n+1}$. This gives

$$\left.\frac{d}{dt}p(t)\right|_{t=t_{n+1}} = \frac{2t_{n+1}-t_n}{2\Delta t^2}y^{n-1} + \frac{t_{n+1}-t_{n-1}}{-\Delta t^2}y^n + \frac{t_{n+1}-t_{n-1}-t_n}{2\Delta t^2}y^{n+1}, \tag{45}$$

$$= \frac{1}{2\Delta t}y^{n-1} - \frac{2}{\Delta t}y^n + \frac{3}{2\Delta t}y^{n+1}, \tag{46}$$

Setting this equal to $f(t_{n+1}, y^{n+1})$, we get

$$\frac{1}{2}y^{n-1} - 2y^n + \frac{3}{2}y^{n+1} = \Delta t f(t_{n+1}, y^{n+1}) + O(\Delta t^3), \tag{47}$$

$$\implies y^{n+1} = \frac{4}{3}y^n - \frac{1}{3}y^{n-1} + \frac{2}{3}\Delta t f(t_{n+1}, y^{n+1}) + O(\Delta t^3). \tag{48}$$

That last error term (the LTE) is a bit surprising. Didn't we differentiate a quadratic polynomial which has error $O(\Delta t^3)$? Yes, we did. Consequently, shouldn't the LTE be $O(\Delta t^2)$ for BDF2? No! In fact, you can show by Taylor expanding the y^{n+1} and y^{n-1} terms that we get a fortuitous $O(\Delta t^3)$ LTE. Thus, the global error here is $O(\Delta t^2)$. Note that despite being another implicit time-integration scheme, the BDF method looks very different from the AM method; rather than f appearing at multiple time levels, we have multiple y values. This is because of the differing philosophies between these two methods.

We can write down a fairly general formula for higher-order BDF methods as well. Keeping in mind that the interval of interest is $[t_{n-s+1}, t_{n+1}]$, we have

$$\frac{d}{dt}\sum_{k=0}^{s} \ell_{n+1-k}(t) y_{n+1-k}\bigg|_{t=t_{n+1}} = f(t_{n+1}, y^{n+1}), \tag{49}$$

$$\implies \sum_{k=0}^{s} \alpha_{n+1-k} y_{n+1-k} = f(t_{n+1}, y^{n+1}), \tag{50}$$

$$\alpha_{n+1-k} = \frac{d}{dt}\ell_{n+1-k}(t_{n+1}). \tag{51}$$

Note: Backward Euler is both BDF1 and AM1. As such, it can be used to start both BDF2 and AM2.

7 A Unified Formula for Multistep Methods

We can now generate a very general formula for all multistep methods for *systems of ODEs*. Before doing so, we summarize:

1. An order s AB method combines f values in $[t_{n+1-s}, t_n]$ to update y^n;
2. An order $s+1$ AM method combines f values in $[t_{n+1-s}, t_{n+1}]$ to *solve* for y^{n+1};
3. An order s BDF method combines y values in $[t_{n+1-s}, t_{n+1}]$, evaluates f at t_{n+1} alone, and solves for y^{n+1}.

Clearly, the AM and BDF philosophies are compatible! Why not combine multiple y values *and* multiple f values to solve for y^{n+1}? This is indeed possible. Thus, our general formula will encompass all the methods in this document, but

will account for possibly new multistep methods as well. Here it is:

$$\sum_{k=0}^{s} \alpha_k \boldsymbol{y}_{n+1-k} = \Delta t \sum_{k=0}^{s} \beta_k \boldsymbol{f}_{n+1-k}. \tag{52}$$

We can see how to recover AB, AM and BDF methods from this formula:

1. For AB methods, $\alpha_k = 0, k > 1$ and $\beta_0 = 0$.
2. For AM methods, $\alpha_k = 0, k > 1$, but $\beta_0 \neq 0$.
3. For BDF methods, $\beta_k = 0, k > 0$.

8 Looking ahead

We derived several multistep methods using integrals and derivatives of polynomial interpolants, and presented a general formula that covered all the cases discussed in this document. Using the polynomial framework, we were also able to derive estimates for local truncation errors (LTEs) for all these methods. We remarked on global truncation errors for every method.

However, we never discussed the *stability* of these methods, beyond pointing out that implicit methods may be more stable than explicit ones. In the next chapter, we will revisit these different methods and discuss their stability based on conditions on their coefficients; this will provide guidelines for choosing implicit methods.