Multi-Step Methods for IVP

Math 131: Numerical Analysis

J.C. Meza

April 30, 2024

Section 1

Introduction

Summary - Single-Step Methods

- Started with a higher-order Taylor series method
- But we didn't like the requirement of higher-derivatives of f.
- Replaced derivatives with appropriate combinations of function evaluations (Runge-Kutta, etc.)
- Price we paid was that we need more function evaluations at each step (computational expense).

Question

Is it possible to use fewer function evaluations and still retain higher-order accuracy?

Motivation

- All the methods so far belong to a class of methods known as one-step methods, which is to say that all of the information used in the computation of the approximation at the next time step only used information from the immediately prior time step.
- However, you might ask yourself, can we use information from other previous steps to improve our approximation. This would be especially useful in the case when each of the function evaluations are computationally expensive.
- This leads us to proposing a set of methods that attempt to take advantage of all of this additional information already available to us.

Multi-step methods

Idea - Multi-step methods

- Maybe we can use past information, and in particular, we could try to use the past values of y_i that we have already computed.
- This leads to the following idea:

Idea

Use some number of past values of y_i (e.g. $y_i, y_{i-1}, y_{i-2}, \ldots$) to fit an ***interpolating polynomial to*** $f(t, y)$, which can then be used to derive a higher order method using techniques similar to the ones we used to derive the multi-stage (i.e. Explicit Trapezoid, Runge-Kutta, etc.) methods

Visually

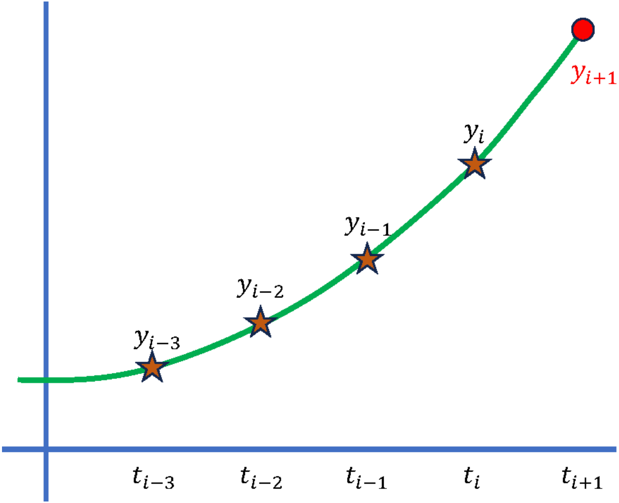

Figure 1: Multi-step method uses past computed values (stars)

Options

- There are many methods one could use to solve the IVP and we will give examples of several of the more popular multi-step methods including the
  - Adams-Bashforth (explicit) and
  - Adams-Moulton (implicit) methods.
- All methods are derived by using an appropriate interpolating polynomial.
- We will not derive all the methods here, but if you're interested we give a brief derivation for the general case in the supplemental section.

Example: Linear interpolating polynomial

As before let's first rewrite the IVP as an integral:

$$y(t_{i+1}) = y(t_i) + \int_{t_i}^{t_{i+1}} f(t, y(t))dt.$$

Using the current and immediate past timesteps as nodal points, we can write a first degree polynomial approximation to f by:

$$P_1(x) = f(t_{i-1}) + \frac{f(t_i) - f(t_{i-1})}{t_i - t_{i-1}}(t - t_{i-1})$$

(cont.)

Similarly to when we looked at quadrature methods, all we need to do is to replace f by $P_1(x)$ in the integral.

$$y(t_{i+1}) = y(t_i) + \int_{t_i}^{t_{i+1}} f(t_{i-1}) + \frac{f(t_i) - f(t_{i-1})}{t_i - t_{i-1}}(t - t_{i-1})\, dt.$$

Integrating the polynomial we get:

$$y(t_{i+1}) = y(t_i) + f(t_{i-1})(t_{i+1} - t_i) + \left[\frac{f(t_i) - f(t_{i-1})}{t_i - t_{i-1}}\right] \cdot \frac{(t - t_{i-1})^2}{2}\Bigg|_{t_i}^{t_{i+1}}.$$

To simplify the equations recall that: $h = (t_{i+1} - t_i)$ and $2h = (t_{i+1} - t_{i-1})$.

(cont.)

$$
\begin{aligned}
y(t_{i+1}) &= y(t_i) + f(t_{i-1}) \cdot h + \left[\frac{f(t_i) - f(t_{i-1})}{2h}\right] \cdot \Big((2h)^2 - h^2\Big). \\
&= y(t_i) + hf(t_{i-1}) + \frac{3h}{2}\Big[f(t_i) - f(t_{i-1})\Big] \quad \text{combine } h \text{ in 2nd term} \\
&= y(t_i) + h\Big[f(t_{i-1}) + \frac{3}{2}(f(t_i) - f(t_{i-1}))\Big] \quad \text{factor out } h \\
&= y(t_i) + h\Big[\frac{3}{2}f(t_i) - \frac{1}{2}f(t_{i-1})\Big].
\end{aligned}
$$

Note: This method can be shown to be $O(h^2)$ accurate.

General Form

Can show that the general formula has form:

$$y(t_{i+1}) = y(t_i) + h \sum_{j=0}^{n} \beta_{nj} f(t_{i-j})$$

Example: Linear Interpolation, $n = 1$:

$$\beta_{n0} = 3/2, \beta_{n1} = -1/2.$$

Adams-Bashforth

Example: Choosing $n = 3$:

$$\beta_{n0} = 55/24; \quad \beta_{n1} = -59/24; \quad \beta_{n2} = 37/24; \quad \beta_{n3} = -9/24.$$

Adams-Bashforth fourth-order

$$\begin{aligned} y_0 = \alpha_0,\ y_1 = \alpha_1,\ y_2 = \alpha_2,\ y_3 = \alpha_3, \\ y_{i+1} = y_i + \frac{h}{24}\Big[55f(t_i, y_i) - 59f(t_{i-1}, y_{i-1}) + \\ 37f(t_{i-2}, y_{i-2}) - 9f(t_{i-3}, y_{i-3})\Big] \end{aligned} \tag{1}$$

Some Key Points

- Notice that Adams-Bashforth methods are explicit since we only use either the current or past (known) function values.
- Adams-Bashforth methods need to supply additional initial values. For example, in the case of an Adams-Bashforth fourth-order method, we need to have 4 initial values in total.
- These are usually computed through another explicit method, for example a Runge-Kutta method, which only require the one initial value.

Adams-Moulton

- We can easily derive implicit methods, by choosing the point at t_{i+1} as one of the interpolation points. The general philosophy still holds!
- This will of course require additional work to solve a nonlinear equation using some iterative method (fixed-point, Newton, etc.)
- As with other implicit methods, the advantage is that we may be able to take longer time steps, thereby reducing the computational cost.

Adams-Moulton

Adams-Moulton fourth-order

$$
\begin{aligned}
y_0 = \alpha_0,\ y_1 = \alpha_1,\ y_2 = \alpha_2,& \\
y_{i+1} = y_i + \frac{h}{24}\Big[9f(t_{i+1}, y_{i+1}) + 19f(t_i, y_i) -& \\
5f(t_{i-1}, y_{i-1}) + f(t_{i-2}, y_{i-2})\Big]&
\end{aligned}
\tag{2}
$$

Note that $f(t_{i+1}, y_{i+1})$ appears on both sides of the equation, which signals that this is an implicit method.

Predictor-Corrector Methods

- The fact that we have an option to use either an explicit method or an implicit method suggests the idea of using the two in combination to get the best of both worlds.
- General idea is to use an (explicit) Adams-Bashforth method to "predict" a value for $f(t_{i+1}, y_{i+1})$.
- Then use an Adams-Moulton method to "correct" this value.
- These types of methods are known as ***Predictor-Corrector methods.***

Example Algorithm

1. **Predict** using one of the Adams-Bashforth formulas:
   - $y_{i+1}^{(0)} = y_i + \frac{h}{24}\Big[55f(t_i, y_i) - 59f(t_{i-1}, y_{i-1})37f(t_{i-2}, y_{i-2}) - 9f(t_{i-3}, y_{i-3})\Big]$
2. **Evaluate** $f = f(t_{i+1}, y_{i+1}^{(0)})$
3. **Correct** using one of the Adams-Moulton formulas, $k = 0, 1, \ldots$:
   - $y_{i+1}^{(k+1)} = y_i + \frac{h}{24}\Big[9f(t_{i+1}, y_{i+1}^{(k)}) + 19f(t_i, y_i) - 5f(t_{i-1}, y_{i-1}) + f(t_{i-2}, y_{i-2})\Big]$
4. **If** $|y_{i+1}^{(k+1)} - y_{i+1}^{(k)}| < \tau$ **then** converged
5. **Evaluate** $f = f(t_{i+1}, y_{i+1}^{(k)})$

Summary

- Both Adams-Bashforth and Adams-Moulton methods are higher-order methods for IVP that do not require additional derivatives. Only 2 function evaluations are required at each step.
- Error estimates are easy to generate.
- Step size and order of the method can be changed to reduce computational cost
- We have to store past values.
- Both methods require additional initial values. These are usually computed through another explicit method, for example a Runge-Kutta method.
- Some of the formulas may be numerically unstable (polynomial interpolation).

Summary of Methods Studied

Table 1: Comparison of Different Solution Methods for IVP

Method	Local Truncation Error	Explicit/Implicit	No. f evals
Euler	$O(h)$	E	1
Backward Euler	$O(h)$	I	1
Higher Order Taylor	$O(h^n)$	E	depends
Midpoint	$O(h^2)$	E	2
Runge-Kutta 2	$O(h^2)$	E	2
Runge-Kutta 4	$O(h^4)$	E	4
Adams-Bashforth	$O(h^4)$	E	1
Adams-Moulton	$O(h^4)$	I	1
Predictor-Corrector	$O(h^n)$	E/I	2+

Section 3

Demo

Demo: Basic SIR Model

- We demonstrate the use of a simple ODE/IVP solver by solving a problem of predicting the breakout of an epidemic using data from Merced County COVID cases taken from: USA Facts Merced County, California coronavirus cases and deaths
- To model an epidemic of an infectious disease, the usual approach is to use what is known as the SIR Model.
- This is one of the most basic (and hence easiest) models we can use. Most practical models take this as a starting point and enhance the model with additional equations.

SIR equations

The SIR equations are given by:

$$\begin{aligned}
\frac{dS}{dt} &= -\alpha S I \\
\frac{dI}{dt} &= \alpha S I - \gamma I \\
\frac{dR}{dt} &= \gamma I \\
N &= S + I + R
\end{aligned}$$

where S is the number of **susceptible** (healthy) individuals, I represents the number of **infected** individuals, R is the number of people who have **recovered** from the disease, and N is the total population.

SIR Parameters

- The demo we presented had 4 parameters you can play with:
  - initial population (N),
  - the number of days to run the simulation for, and
  - the 2 parameters that represent the rates between susceptible and infected (α) and between infected and recovered (γ).
- The solver used comes from the deSolve package in R called ode. The default solver is "lsoda" (Petzold & Hindmarsh), but other choices are available.
- Calling the ode solver requires the initial conditions (init), the times at which to compute the solution (times, the function to evaluate the ode (sir in this case), and a list of parameters that the ode solver passes along to the ode function.

SIR Data

The original data taken from the site gave us the following plot:

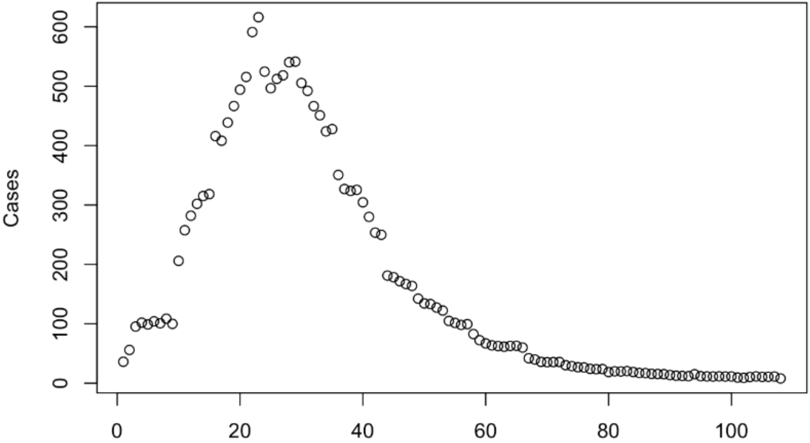

Figure 2: Merced County Covid Cases (weekly average)

Numerical Experiments

- In the demo, we played around with the parameters for the SIR model to match the data as best we could.
- According to the model, the basic Reproduction number $R_0 = [3.5 - 8.5]$.
- For comparison, for measles one of the more contagious diseases, $R_0 = 12 - 18$ while the normal flu has $R_0 \approx 1.28$.

Solution

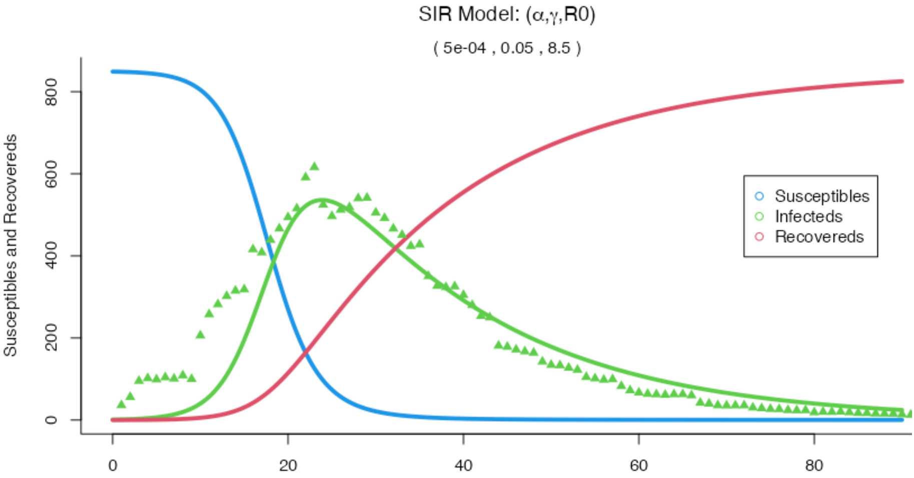

Figure 3: SIR demo using Merced County data

Supplemental Materials

Definition

Let's first start with some notation and a definition. As before, let's assume that we have equally spaced time steps such that $t_i = a + ih, i = 0, 1, N.$

Definition: *an s-step linear multistep method for solving the IVP has a difference equation of the form:*

$$\sum_{j=0}^{s} \alpha_j y_{i+1-j} = h \sum_{j=0}^{s} \beta_j f_{i+1-j}, \tag{3}$$

where we let $f_{i+1-j} = f(t_{i+1-j}, y_{i+1-j})$.

Note - without loss of generality, we can assume that $\alpha_0 = 1$ since we can rescale all of the equations.

Explicit methods

- It will also be useful to distinguish between cases that need the function value $f(t_{i+1}, y_{i+1})$ at the next time step to compute y_{i+1}.
- In particular, if $b_0 = 0$ the method is called an **explicit (open) method** and we can write Equation 3 as:

$$y_{i+1} = -\sum_{j=1}^{s} \alpha_j y_{i+1-j} + h \sum_{j=1}^{s} \beta_j f_{i+1-j}$$

- Notice that we can recover Euler's method from the explicit form of this equation by setting $\beta_0 = 0, s = 1, \alpha_1 = -1, \beta_1 = 1$.

Implicit methods

The second case is if we let $b_0 \neq 0$ and the method is then called **implicit (closed)** as y_{i+1} appears on both sides of Equation 3 so it is only implicitly defined.

$$y_{i+1} - h\beta_0 f_{i+1} = -\sum_{j=1}^{s} \alpha_j y_{i+1-j} + h \sum_{j=1}^{s} \beta_j f_{i+1-j}$$

Note

In general implicit methods are more accurate than explicit methods and we can get by with larger time steps. The disadvantage is that we need to solve a system of linear (or nonlinear) equations at each time step. Additionally, the solution may not be unique (or even exist).

Derivation of multi-step methods

Advanced: This derivation closely follows the proof in Burden and Faires, pages 304-305. There are other similar proofs.

Our first step is to write:

$$
\begin{aligned}
y(t_{i+1}) - y(t_i) &= \int_{t_i}^{t_{i+1}} y'(t)dt \\
&= \int_{t_i}^{t_{i+1}} f(t, y(t))dt
\end{aligned}
$$

(cont.)

Let $y_i \approx y(t_i)$, and rearrange the equation to give us an expression for the approximation at the next time step t_{i+1}

$$y(t_{i+1}) \approx y_i + \int_{t_i}^{t_{i+1}} f(t, y(t))dt \tag{4}$$

This should remind us of a similar problem we studied earlier, namely the numerical approximation for an integral, i.e. quadrature.

In the earlier case, we replaced the function by a polynomial, for which it will be easier to compute the integral. In that case, we used a Lagrange interpolating polynomial.

Here, it will be more convenient to use a ***Newton backward-difference polynomial*** because we can more easily incorporate previously calculated values.

(cont.)

As reminder we can write the $m-1$ degree interpolating polynomial as (ref: equation 3.13, p. 130 textbook):

$$P_{m-1}(t) = \sum_{k=0}^{m-1} (-1)^k \binom{-s}{k} \nabla^k f(t). \tag{5}$$

We can then use this polynomial (along with the remainder term) as an approximation to $f(t, y)$

$$f(t, y) = P_{m-1}(t) + \frac{1}{m!} f^{(m)}(\xi_i, y(\xi_i))(t - t_i)(t - t_{i-1}) \ldots (t - t_{i+1-m}) \tag{6}$$

where $\xi_i \in (t_{i+1-m}, t_i)$.

(cont.)

Substituting Equation 5 and Equation 6 into Equation 4, and taking the integral of both sides, yields:

$$\int_{t_i}^{t_{i+1}} f(t,y(t))dt = \int_{t_i}^{t_{i+1}} \sum_{k=0}^{m-1} (-1)^k \binom{-s}{k} \nabla^k f(t_i, y(t_i))dt$$
$$+ \int_{t_i}^{t_{i+1}} \frac{1}{m!} f^{(m)}(\xi_i, y(\xi_i))(t-t_i)(t-t_{i-1}) \dots (t-t_{i+1-m})dt. \tag{7}$$

The integral is easier to solve by using the variable substitution:

$$t = t_i + sh$$
$$dt = hds$$

in Equation 7

(cont.)

Substituting yields:

$$\int_{t_i}^{t_{i+1}} f(t, y(t))dt = h \left[\sum_{k=0}^{m-1} \nabla^k f(t_i, y(t_i)) \, (-1)^k \int_0^1 \binom{-s}{k} ds \right] + \frac{h^{m+1}}{m!} \int_0^1 (s)(s+1) \ldots (s+m-1) f^{(m)}(\xi_i, y(\xi_i)) ds$$

The integrals involving the binomial function are easily computed (see Table 5.12 (textbook)). ***Note:*** *The header in Table 5.12 incorrectly states the second column. It should read* $(-1)^k \int_0^1 \binom{-s}{k} ds$

Also, recall that $\nabla p_n = p_n - p_{n-1}, n \geq 1$ and
$\nabla^k p_n = \nabla(\nabla^{k-1} p_n), k \geq 2$ (see p. 130, textbook)

(cont.)

Substituting the computed values of the integrals simplifies the equation to:

$$
\begin{aligned}
\int_{t_i}^{t_{i+1}} f(t, y(t))dt = h\Big[& f(t_i, y_i) + \frac{1}{2}\nabla f(t_i, y_i) \\
& + \frac{5}{12}\nabla^2 f(t_i, y_i) + \frac{3}{8}\nabla^3 f(t_i, y_i) + \dots \Big] \\
& + \frac{h^{m+1}}{m!}\int_0^1 (s)(s+1)\dots(s+m-1) f^{(m)}(\xi_i, y(\xi_i))ds
\end{aligned}
$$

(cont.)

The last step is to recognize that:

$$\begin{aligned}
\nabla^0 f(t_i, y_i) &= f(t_i, y_i) \\
\nabla^1 f(t_i, y_i) &= f(t_i, y_i) - f(t_{i-1}, y_{i-1}) \\
\nabla^2 f(t_i, y_i) &= \nabla(\nabla^1 f(t_i, y_i)) \\
\nabla^3 f(t_i, y_i) &= \nabla(\nabla^2 f(t_i, y_i))
\end{aligned}$$

to expand the backward difference terms in the integral, followed by collecting like terms to arrive at the Adams-Bashforth Four-Step ($m = 4$) Method:

$$y_0 = \alpha_0,\ y_1 = \alpha_1,\ y_2 = \alpha_2,\ y_3 = \alpha_3,$$

$$y_{i+1} = y_i + \frac{h}{24}\Big[55 f(t_i, y_i) - 59 f(t_{i-1}, y_{i-1}) \\ + 37 f(t_{i-2}, y_{i-2}) - 9 f(t_{i-3}, y_{i-3})\Big]$$