IVP Higher-Order Methods

Math 131: Numerical Analysis

J.C. Meza

April 25, 2024

Section 1

Higher-order Taylor Methods

Higher-order Taylor Methods

- In a similar vein to what we did to derive Euler's method, we can derive higher-order methods by extending the Taylor series approximation to include the higher derivatives.
- This will yield methods that are more accurate, but at a cost of having to compute the higher derivatives.
- This is important to remember because in a real-world problem each function evaluation could cost hours of computer time.

Caution

While higher-order Taylor methods can be generated, they are not often used in practice, as the requirement for higher-order derivatives is rarely met in real-world problems.

Extending Euler's method to higher-order

Idea

If Euler's method used a Taylor series expansion truncated after the first derivative (i.e. $n = 1$), it is natural to try higher values of n.

Taylor's Series

Let's start by writing down the Taylor series expansion of $y(t)$ about the current time step, t_i.

$$y(t_{i+1}) = y(t_i) + hy'(t_i) + \frac{h^2}{2}y''(t_i) + ...$$
$$+\frac{h^{(n)}}{n!}y^{(n)}(t_i) + \frac{h^{(n+1)}}{(n+1)!}y^{(n+1)}(\xi_i), \quad \xi_i \in [t_i, t_{i+1}]$$

Using the statement of the IVP, we know that $y' = f(t, y)$, so we can replace the derivatives of y by the corresponding derivative of $f(t, y)$

(cont.)

Using $y' = f(t, y)$ leads to:

$$y(t_{i+1}) = y(t_i) + hf(t_i, y(t_i)) + \frac{h^2}{2} f'(t_i, y(t_i)) + ...$$
$$+ \frac{h^{(n)}}{n!} f^{(n-1)}(t_i, y(t_i)) + \frac{h^{(n+1)}}{(n+1)!} f^{(n)}(t_i, y(t_i)),\ \xi_i \in [t_i, t_{i+1}]$$

For convenience let's denote the function $T^{(n)}$ as:

$$T^{(n)} = f(t_i, y(t_i)) + \frac{h}{2} f'(t_i, y(t_i)) + ... + \frac{h^{(n-1)}}{n!} f^{(n-1)}(t_i, y(t_i)).$$

Example

$$
\begin{aligned}
T^{(1)} &= f(t_i, y(t_i)), \\
T^{(2)} &= f(t_i, y(t_i)) + \frac{h}{2} f'(t_i, y(t_i)), \\
T^{(3)} &= f(t_i, y(t_i)) + \frac{h}{2} f'(t_i, y(t_i)) + \frac{h^2}{3!} f^{(2)}(t_i, y(t_i)), \\
&\vdots
\end{aligned}
$$

Higher-order Taylor methods

- In a manner similar to Euler's method, this leads us to propose the following algorithm for higher-order Taylor methods:
- $$y_0 = \alpha$$
$$y_{i+1} = y_i + hT^{(n)}(t_i, y_i),\ i = 0, 1, \ldots, N-1.$$
- Clearly Euler's method is the special case of $n = 1$.

Remarks

- *By the definition of the **local truncation error**, we can see that the higher-order Taylor method will have a local truncation error of* $O(h^n)$, using the same argument that we used for Euler's method.
- For example, if we wanted to have a second order method, we would use $T^{(2)}$ in our difference equation:

$$T^{(2)} = f(t_i, y(t_i)) + \frac{h}{2} f'(t_i, y(t_i))$$

- Main disadvantage is the need for higher derivatives of $f(t, y)$.

Section 2

Runge-Kutta Methods

Runge-Kutta Methods

An alternative approach to higher-order Taylor leads to the well-known class of Runge-Kutta methods.

Idea

The idea for the Runge-Kutta methods is to have the high-order local truncation error of Taylor methods ***without the need to compute and evaluate the derivatives*** **of** $f(t, y)$. Ideally, we should only need function evaluations.

There are several ways to approach this. Your book shows you one way through matching coefficients to the higher-order Taylor method we just described. We'll take a different approach.

Runge-Kutta Order 2

Let's first rewrite the IVP as an integral:

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

One simple idea is to now use our methods from numerical integration to evaluate the integral, for example the Trapezoid rule:

$$y_{i+1} = y_i + \frac{h}{2}(f(t_i, y_i) + f(t_{i+1}, y_{i+1})). \tag{2}$$

As with our discussion of Backward Euler, we can see that this is an implicit equation as y_{i+1} is both on the right and left hand sides of the equation and would (in general) require the solution of a nonlinear equation.

Second Idea

But what if we could approximate the value of y_{i+1} on the right hand side of Equation 2 with an explicit method?

Since we only know one explicit method (Euler's method), let's try it out and see what happens.

Let's define:

$$Y = y_i + hf(t_i, y_i),$$

and substitute Y for y_{i+1} into Equation 2:

$$y_{i+1} = y_i + \frac{h}{2}(f(t_i, y_i) + f(t_{i+1}, Y)).$$

These 2 equations comprise the ***explicit trapezoidal method***.

Explicit Trapezoid Method

- The explicit trapezoidal method is also known as a two-stage Runge-Kutta method
- The second stage comes about because we have to evaluate the function at a second point, namely $f(t_{i+1}, Y)$.
- We can summarize this approach by the following steps:

Explicit Trapezoid (2-stage explicit RK)

$$\begin{aligned} Y &= y_i + hf(t_i, y_i) \\ y_{i+1} &= y_i + \frac{h}{2}(f(t_i, y_i) + f(t_{i+1}, Y)) \end{aligned} \tag{3}$$

It can be shown that this method is second order accurate.

Explicit Midpoint

A similar process but using the midpoint rule instead of the trapezoid rule to compute the integral in Equation 1 will also work and yields the following two-stage algorithm, known as the ***explicit midpoint*** method:

Explicit Midpoint

$$\begin{aligned} Y &= y_i + \frac{h}{2} f(t_i, y_i) \\ y_{i+1} &= y_i + h f(t_{i+1/2}, Y) \end{aligned} \tag{4}$$

Programming Tip

Programming Tip

When coding the algorithms above, for example the ***explicit trapezoidal method***, the algorithm is usually stated in terms of the following:

$$
\begin{aligned}
k_1 &= f(t_i, y_i) \\
k_2 &= f(t_{i+1}, y_i + hk_1) \\
y_{i+1} &= y_i + \frac{h}{2}(k_1 + k_2)
\end{aligned}
$$

You should convince yourself that these are equivalent to Equation 3.

Runge Kutta Order 4

We can derive higher-order Runge-Kutta methods using similar techniques. One of the most popular is the 4th order method, which can be written as:

$$
\begin{aligned}
k_1 &= f(t_i, y_i), \\
k_2 &= f(t_{i+1/2}, y_i + \frac{h}{2} k_1), \\
k_3 &= f(t_{i+1/2}, y_i + \frac{h}{2} k_2), \\
k_4 &= f(t_{i+1}, y_i + h k_3), \\
y_{i+1} &= y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4).
\end{aligned}
$$

As the name implies, this method is $O(h^4)$.

Comparison of Runge Kutta methods

- Higher-order RK methods have higher accuracy but also cost more.
- For RK methods, it can be shown that the following relationship holds between the additional function evaluations and the local discretization error:

Table 1: Function Evals vs. Local Error

Function Evaluations	Local Discretization Error
2	$O(h^2)$
3	$O(h^3)$
4	$O(h^4)$
$5 \leq n \leq 7$	$O(h^{n-1})$
$8 \leq n \leq 9$	$O(h^{n-2})$
$10 \leq n$	$O(h^{n-3})$

Adaptive Steps

- Adaptive methods can be derived that adjust the time step to achieve a desired accuracy
- The methods use an estimate of the truncation error similar to the approach we saw in numerical integration
- One popular method (Runge-Kutta-Fehlberg) uses a combination of 2 RK methods of order 4 and 5 to adjust the time step to keep the discretization error below a desired tolerance

Section 3

Summary

Summary

- Higher-order Taylor methods can be derived, extending the accuracy of Euler's method, although these methods are rarely use in practice.
- Another approach is to use the same idea but replace higher derivatives of $f(t, y)$ with other approximations.
- Leads to multi-stage methods such as explicit Trapezoidal, explicit midpoint methods and the widely used and popular class of methods known as Runge-Kutta methods.
- Choosing a good method will depend on the problem, how smooth the functions are, and the desired accuracy.

Section 4

Supplementary Materials

General form for Runge Kutta methods

The general form of the Runge-Kutta equations with s stages is given by:

$$y_{i+1} = y_i + h \sum_{j=1}^{s} b_j k_j, \tag{5}$$

where the k_i terms are given by:

$$\begin{aligned}
k_1 &= f(t_i, y_i), \\
k_2 &= f(t_i + hc_2, y_i + a_{21}k_1), \\
k_3 &= f(t_i + hc_3, y_i + a_{31}k_1 + a_{32}k_2), \\
&\vdots \\
k_s &= f(t_i + hc_s, y_i + a_{s1}k_1 + a_{s2}k_2 + ... + a_{s,s-1}k_{s-1}),
\end{aligned} \tag{6}$$

The coefficients, a_{ij}, b_i, c_i, are determined by the process we introduced earlier that compares partial derivatives.

(cont.)

In addition, for consistency, we ask that the coefficients satisfy the equations:

$$\sum_{j=1}^{i-1} a_{ij} = c_i, \quad i = 2, \dots, s$$

and

$$\sum_{j=1}^{s} b_j = 1,$$

An easy way to visualize the coefficients is to use the Butcher tableau:

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array}$$

where c, b are vectors of coefficients, and A is a matrix of coefficients, as defined in Equation 5 and Equation 6.

(cont.)

Using this tableau, we can, for example, summarize the formula for the Runge-Kutta method of order 2 (RK2) as:

$$\begin{array}{c|cc} 0 & 0 & \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \hline & 0 & 1 \end{array}$$

Substituting into Equation 6 yields the formulas:

$$\begin{aligned} k_1 &= f(t_i, y_i), \\ k_2 &= f(t_i + \frac{1}{2}h, y_i + \frac{1}{2}hk_1), \\ y_{i+1} &= y_i + hk_2. \end{aligned}$$

(cont.)

We can also derive a similar set of formulas for a Runge-Kutta method with 4 stages that will yield a method of $O(h^4)$. The corresponding tableau is given by.

$$
\begin{array}{c|cccc}
0 & 0 & & & \\
\frac{1}{2} & \frac{1}{2} & 0 & & \\
\frac{1}{2} & 0 & \frac{1}{2} & 0 & \\
1 & 0 & 0 & 1 & 0 \\
\hline
 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6}
\end{array}
$$

(cont.)

Again, substituting into Equation 6, yields the Runge-Kutta method of order 4:

$$
\begin{aligned}
k_1 &= f(t_i, y_i), \\
k_2 &= f(t_i + \frac{1}{2}h, y_i + \frac{1}{2}hk_1), \\
k_3 &= f(t_i + \frac{1}{2}h, y_i + \frac{1}{2}hk_2), \\
k_4 &= f(t_i + h, y_i + hk_3), \\
y_{i+1} &= y_i + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4).
\end{aligned}
$$

Recall that our goal was to replace the need for higher order derivatives while retaining the higher order local discretization error. The price we had to pay was in the form of extra function evaluations.

(cont.)

In the case of RK2, it was an additional 2 function evaluations. For RK4, it was 4 extra function evaluations. It can be shown that the following relationship holds between the additional function evaluations and the local discretization error holds for Runge-Kutta methods:

Table 2: Function Evals vs. Local Error

Function Evaluations	Local Discretization Error
2	$O(h^2)$
3	$O(h^3)$
4	$O(h^4)$
$5 \leq n \leq 7$	$O(h^{n-1})$
$8 \leq n \leq 9$	$O(h^{n-2})$
$10 \leq n$	$O(h^{n-3})$

Summary

- As you can see, when $n > 4$, there is no big advantage to increasing the order in terms of increasing the order of the local discretization error.
- As a consequence, the most popular of the Runge-Kutta methods is the one of order 4.