Higher-Order Taylor Methods

Sunday, October 25, 2020 3:27 PM

We'll discuss theory and give a prelude to Runge-Kutta methods

These higher-order Taylor methods are seldom used in practice

Recall our IVP $y' = f(t,y)$, $a \le t \le b$, $y(a) = y_0$ ← or α in book's notation

y_i is shorthand for $y(t_i)$

Our numerical scheme creates w_i to approximate y_i

Ex: Euler $w_0 = y_0$

$w_{i+1} = w_i + h f(t_i, w_i)$

We'll generalize to generic "one-step" methods, of which Euler and Runge-Kutta belong (and "high-order Taylor")

One-step method
$$w_0 = y_0$$
$$w_{i+1} = w_i + h\,\phi(t_i, w_i, h^*)$$

* Burden and Faires write $\phi(t_i, w_i)$, leaving the h implicit

and define the local truncation error τ

$$\tau_{i+1}(h) = \frac{y_{i+1} - (y_i + h\phi(t_i,y_i))}{h} = \frac{y_{i+1} - y_i}{h} - \phi(t_i, y_i)$$

i.e., how much does the true solution y_i fail to solve the difference eq'n

(a proxy for how much the approximate solution w_i fails to solve the differential equation)

Why divide by h?

$y(t)$ is true solution, and it's continuous (since y' exists)

Let's pick a bad numerical scheme: $\phi(t,y) = 0$

so $w_{i+1} = w_i + 0$ so $w_i = y_0 \ \forall i$. Obviously not good

But if we looked at

$$y_{i+1} - (y_i + h\phi(t,y)) = y_{i+1} - y_i$$

$$= 0 \qquad = y(t_i + h) - y(t_i)$$

$\longrightarrow 0$ as $h \to 0$

since $y(t)$ is continuous

... so a small error wouldn't mean much.

In fact, we'll want $\phi(t, y, h=0) = f(t,y)$: "Consistency"

More on this later

Why care about local truncation error?

For reasonable ODEs, it bounds global error!

Thm 6.2.1 Driscoll and Braun: global error of one-step methods

Consider a one-step method defined by $\phi(t, y, h)$ and

Suppose the local truncation error satisfies $\tau_{i+1}(h) \le C \cdot h^p \quad (\forall i)$

for $p > 0$, and that ϕ is uniformly Lipschitz in y w/ constant L,

and assume there is a unique solution $y(t)$ to the IVP.

Then the global error is bounded

$$|y(t_i) - w_i| \le \frac{C h^p}{L} \left(e^{L(t_i - a)} - 1 \right) = O(h^p)$$

proof: Similar to that of Thm 5.9 in Burden and Faires.

... so, small τ is good!

Recall, for Euler's Method, $\tau_{i+1}(h) = \frac{h}{2} y''(\xi) = O(h)$ if y'' bounded

Higher-Order Taylor Methods

Find better $\phi(t, y, h)$ so τ is $O(h^p)$ for $p \ge 2$

Similar to Runge-Kutta methods we'll see shortly, except high-order Taylor methods rely on knowledge of f' (whereas RK don't)

Euler:

$$y(t_{i+1}) = y(t_i) + h \cdot \underbrace{y'(t_i)}_{= f(t_i, y(t_i)) \text{ via ODE}} + \frac{h^2}{2!} y''(\xi)$$

Recall $t_{i+1} = t_i + h$

So... numerical method just ignores

$$w_{i+1} = w_i + h \cdot f(t_i, w_i) \quad \leftarrow \text{Euler's method}$$

Higher-Order:

Taylor method of order p "$T^{(p)}$"

$$w_0 = y_0$$
$$w_{i+1} = w_i + h \cdot T^{(p)}(t_i, w_i, h)$$

where $T^{(p)}(t_i, w_i, h) = f(t_i, w_i) + \frac{h}{2!} f'(t_i, w_i) + \dots + \frac{h^{p-1}}{p!} f^{(p-1)}(t_i, w_i)$

So (Thm 5.12) if $y \in C^{p+1}[a,b]$ ($\Rightarrow y^{(p+1)}$ is bounded via E.V.T)
then $T^{(p)}$ has $\tau_{i+1}(h) = O(h^p)$ ☆

...

we just need to know $f'(t, y(t))$ write $y(t)$ instead of y to remind us...

⚠ $f'(t, y(t))$ means $\frac{d}{dt} f(t, y(t))$ Total deriv.

NOT $\frac{\partial}{\partial t} f(t, y(t))$ Partial deriv.

$$f' = \frac{d}{dt} f(t, y(t))$$
$$= \frac{\partial}{\partial t} f(t, y(t)) + \frac{\partial}{\partial y} f(t, y(t)) \cdot \underbrace{\frac{d}{dt} y}_{\text{aka } y', \text{ and } y' = f}$$
$$= \frac{\partial}{\partial t} f(t,y) + \left(\frac{\partial}{\partial y} f(t,y)\right) \cdot f(t,y)$$

<u>Ex</u>: $y' = y - t^2 + 1$ so $f(t,y) = y - t^2 + 1$

$$f' = \frac{d}{dt} f = \frac{\partial}{\partial t} f + \left(\frac{\partial}{\partial y} f\right) \cdot f$$
$$= -2t + (1) \cdot (y - t^2 + 1)$$

How to find $f'' := \frac{d^2}{dt^2} f$? Just find $\frac{d}{dt} f'$

$$f'' = \frac{d}{dt}(-2t + y - t^2 + 1)$$
$$= \frac{\partial}{\partial t}(\dots) + \left(\frac{\partial}{\partial y}(\dots)\right) \cdot \overbrace{\frac{d}{dt} y}^{= f \ (\text{NOT } f')}$$
$$= (-2 - 2t) + (1) \cdot (y - t^2 + 1)$$
$$= -t^2 - 2t - 1 + y .$$

<u>Note</u>: as discussed in demo, we usually <u>interpolate</u> w_i using, e.g., Hermite interpolation

For <u>Euler</u>, just a simple <u>piecewise linear</u> interpolation is ok, because the w_i are inaccurate (or to make accurate, h is very small)

For <u>higher-order methods</u>, interpolation should also be high-order (or Hermite), so it's now more important not to just do <u>piecewise linear interp.</u>

i.e., plot(t, w, '-') is
plotting piecewise linear interp.