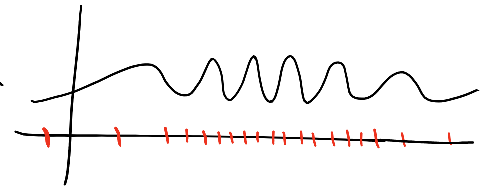
# Adaptive Runge-Kutta & Error Control

Thursday, November 5, 2020 10:21 PM

Two goals: (1) chose a wise spacing $h$, i.e., not uniform



(2) estimate error, and adapt so that error is bounded by our desired tolerance

This is quite similar to what we did for integration, ch. 4.6

- We'll use 2 methods, one high order, one lower, and use the high order method to estimate the error of the lower order method
- For efficiency, usually choose the low-order method to be embedded inside the higher order method.

Ex. of Embedded Formulas include Bogacki-Shampine (2nd/3rd order methods) and Runge-Kutta Fehlberg (RK45)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $0$ | | | | | | | |
| $\frac{1}{4}$ | $\frac{1}{4}$ | | | | | | |
| $\frac{3}{8}$ | $\frac{3}{32}$ | $\frac{9}{32}$ | | | | | |
| $\frac{12}{13}$ | $\frac{1932}{2197}$ | $-\frac{7200}{2197}$ | $\frac{7296}{2197}$ | | | | |
| $1$ | $\frac{439}{216}$ | $-8$ | $\frac{3680}{513}$ | $-\frac{845}{4104}$ | | | |
| $\frac{1}{2}$ | $-\frac{8}{27}$ | $2$ | $-\frac{3544}{2565}$ | $\frac{1859}{4104}$ | $-\frac{11}{40}$ | | |
| | $\frac{25}{216}$ | $0$ | $\frac{1408}{2565}$ | $\frac{2197}{4104}$ | $-\frac{1}{5}$ | | (L) |
| | $\frac{16}{135}$ | $0$ | $\frac{6656}{12825}$ | $\frac{28561}{56430}$ | $-\frac{9}{50}$ | $\frac{2}{55}$ | (H) |
| | $\frac{1}{360}$ | $0$ | $\frac{-128}{4275}$ | $\frac{-2197}{75240}$ | $\frac{1}{50}$ | $\frac{2}{55}$ | (H) - (L) |

4th order RK formula (5-stage) embedded within the 6-stage 5th order RK formula

So, (1) estimate error

let (L) be our low-order method (1-step method)

$$w_{i+1}^{L} = w_i^{L} + h\phi^{L}(t_i, w_i, h)$$

with local truncation error $\tau_{i+1}^{L}(h) = O(h^{n})$ for some $n \geq 1$

let (H) be our higher-order method

NOTE: the whole point of this lecture is that $h$ isn't constant/uniform, so we really ought to write $h_i$

$$w_{i+1}^{H} = w_i^{H} + h\phi^{H}(t_i, w_i, h)$$

with local truncation error $\tau_{i+1}^{H}(h) = O(h^{n+1})$

we're going to assume $\tau$ is a good proxy for global error

( cf Thm 6.2.1 in Driscoll + Braun, mentioned in notes on High Order Taylor Methods )

**Note**: we could work with the local error $y_{i+1} - (y_i + h \cdot \phi(t_i, y_i))$

instead of the local truncation error $\tau_i = \frac{1}{h}\left(y_{i+1} - (y_i + h\,\phi(t_i, y_i))\right)$

and so adjust $n \to n+1$ for Ⓛ

$n+1 \to n+2$ for Ⓗ.

Some textbooks do this, and it's a bit more common in modern practice. We'll stick w/ Burden and Faires

To estimate error, first note what the local truncation error measures:

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

(recall $y_i = y(t_i)$, $y$ is true soln)

and we're going to ignore the error made previously, so we assume $w_i = y_i$

$$\ldots = \frac{y_{i+1} - (\overbrace{w_i + h\,\phi(t_i, w_i)}^{w_{i+1}})}{h} = \frac{1}{h}\left(y_{i+1} - w_{i+1}\right)$$

of course we don't know $y_{i+1}$.

For Ⓛ in particular, $\tau_{i+1}^{L}(h) = \frac{1}{h}\left(y_{i+1} - w_{i+1}^{L}\right)$

Main idea: use $w_{i+1}^{H}$ as a proxy for $y_{i+1}$

so we estimate $\tau_{i+1}^{L}(h) \approx E := \frac{1}{h}\left(w_{i+1}^{H} - w_{i+1}^{L}\right)$

We have a target accuracy $\varepsilon$

1) if $|E| \leq \varepsilon$, great! we "accept" our stepsize $h$

and increment our counter $i$

2) if $|E| > \varepsilon$, then we "reject" our stepsize $h$, so need to estimate what a better stepsize is

Estimating a new stepsize

If we took a stepsize $qh$ (instead of $h$), and

if $|\tau_{i+1}^{L}(h)| \approx c \cdot h^n$ then we'd expect

$$\underbrace{|\tau_{i+1}^{L}(qh)|}\approx c\,q^n h^n \approx \underbrace{q^n |E| \overset{?}{<} \varepsilon}$$

want $< \varepsilon$ Set $q = \left(\frac{\varepsilon}{|E|}\right)^{1/n}$ ... or $\left(\frac{\varepsilon}{2|E|}\right)^{1/n}$ to be conservative.

... then repeat using stepsize $qh$

## Implementation Details

1. Use embedded RK formulas
2. use First Same As Last (FSAL)
3. 4th order is a good choice

} for efficiency } Dormand Prince 4th/5th order, used in ode45 in Matlab

4. our error estimate was for $w_{i+1}^{L}$, but why not actually use $w_{i+1}^{H}$ since we already computed it? This is commonly done in practice.

5. Check that $h$ never drops so small that $t_i + h = t_i$ in floating pt. If this happens, often due to a singularity in the solution (ex: $y(t) = \frac{1}{1-t}$). Code should exit with an error message.

6. If we accepted our step, that's great, but maybe $h$ was unnecessarily small? Can use that same equation for $q$, but now $q > 1$, and we use $qh$ as the timestep for the next interval. Usually limit stepsize to $\min(q, 4)h$ to reflect some distrust in our estimate (and next time interval may be different). Also sometimes require stepsize $< h_{max}$ (user supplied)

i.e. $h_{i+1} = q h_i$

7. If predicted $q$ is very small, like $< 0.1$, we also might be skeptical. Common practice is to take $q \leftarrow \max(q, 0.1)$

### Take-away

- Understand how we (heuristically) estimate the error

  numerics has a lot of heuristics that are informed by math, but ultimately need to be practical

- understand how we estimate a new stepsize $qh$ if we weren't happy with the error

- details less important... but you now have enough info to implement your own professional quality "ode45", which is a very nontrivial ODE/IVP solver!