Fehlberg Method (Embedded RK formula)

- 5^{th} order accuracy, variable time step, only

Generally, RK$(4+m)$ - m orders higher than 4 - requires additional $f^{\underline{n}}$ calls, but not more than .

Fehlberg discovered an RK5 with

$$k_1 = hf(y_n, x_n)$$
$$k_2 = hf(y_n + b_{21}k_1, \; x_n + a_2 h)$$
$$k_3 = hf(y_n + b_{31}k_1 + b_{32}k_2, \; x_n + a_3 h)$$
$$\vdots$$

$y_{n+1} =$

$=$ RK5

that also yield a 4^{th} order method (RK4) with the

$y^*_{n+1} =$

$=$ RK4

The a_i's and b_{ij}'s are the same for both, but the c_i's and c_i^*'s are different.

The parameters commonly used nowadays are due to Cash and Karp (see e.g. Numerical Recipes)

We get an RK5 (5^{th} order method) <u>AND</u> an estimate of the error.

$\Delta =$

with only 6 f^{n} calls. More efficient than RK4-based step halving/doubling.

As before, if stepsize produces
we want stepsize that produces

$\Delta \sim h^5 \rightarrow$

In Numerical Recipes notation

Practical implementation issues regarding

- currently an
- different for
- different components may differ
- not general

better: want solution

set with - relative error tolerance

but what if y oscillates (and $y=0$ sometimes)?

set

In NR code

$i = 1, \dots n \rightarrow$

- if $|y(i)| \simeq 0$,

$h_{new} =$

use largest relative error $\frac{\Delta_1}{\Delta_0}$ over all components

use largest $\frac{y_{err}(i)}{\epsilon \, y_{scale}(i)}$ to control step size

Alternatively, may want to limit global error
(error at final integration point),

global error limit $\simeq$

$\therefore$ want (target error at a step should scale with h)

choose $\Delta_0 =$ as in NR's yscale

But then our formula needs to

change (since global error $\sim \mathcal{O}(h^4)$)

to

Note that
and are not too different.

NR takes a pragmatic, general, and conservative approach (not increasing or decreasing h too much per step).

When increasing step size ($\Delta_1 < \Delta_0$)

use $h_0 =$

but $h_0 \leq$ (upper limit on stepsize increase)

When decreasing step size $(\Delta_1 > \Delta_0)$

use $h_0 =$

but $h_0 \geq$

lower limit on stepsize decrease

Also check that h does not become too small, ie, check that

These aspects of the implementation are not rigourous, but are experience-based, practical steps to produce robust code that will solve most problems.

User must supply function that returns

and initialize $\vec{y}$.

Part of Project 2 deals with using NR code to solve a physics problem.

Appendix - Details on global error

Δ_1 - error estimate at each step $\sim \Theta(h^5)$

$$\Delta_1 = C h^5$$

Δ_0 - desired local error : $\Delta_0 = \epsilon h \frac{dy}{dx}$

Δ_1^g - approx. global error :

$$\Delta_1^g = N \Delta_1 \sim \frac{\Delta_1}{h} \sim \Theta(h^4) = D h^4$$

Δ_0^g - desired global error :

$$\Delta_0^g \doteq \epsilon h \frac{dy}{dx} N \sim \epsilon h \frac{dy}{dx} \frac{1}{h} = \epsilon \frac{dy}{dx}$$

$$\begin{aligned} h^4 D &= \Delta_1^g \\ h_{new}^4 D &= \Delta_0^g \end{aligned} \Rightarrow \frac{h^4}{h_{new}^4} = \frac{\Delta_1^g}{\Delta_0^g}$$

$$h_{new} = h\left(\frac{\Delta_0^g}{\Delta_1^g}\right)^{1/4} \sim h\left(\frac{\epsilon \, dy/dx}{\Delta_1/h}\right)^{1/4} = h\left(\frac{\epsilon h \, dy/dx}{\Delta_1}\right)^{1/4}$$

$$= h\left(\frac{\Delta_0}{\Delta_1}\right)^{1/4}$$