The midpoint method is an example of a Runge-Kutta scheme

↳ use function calls to reduce truncation error

→ need to find where to evaluate f's, and associated weight

To obtain R-K scheme for $\frac{dy}{dt} = f(y,t)$

write $y(t+h) = y(t) + h y'(t) + \frac{h^2}{2} y''(t) + \frac{h^3}{3!} y'''(t) + \dots$

$$y'(t) = \frac{dy}{dt} = f(y,t)$$

$$\frac{d^2y}{dt^2} = \frac{\partial f}{\partial y}\frac{dy}{dt} + \frac{\partial f}{\partial t} = f_y f + f_t$$

$$\frac{d^3y}{dt} = \dots$$

$$y(t+h) = y + hf + \frac{h^2}{2}(f_t + f f_y) + \frac{h^3}{6}\left(f_{tt} + 2f f_{ty} + f^2 f_{yy} + f f_y^2 + f_t f_y\right) + \dots$$

Can also write

$$\star \quad y(t+h) = y(t) + \alpha_1 c_1 + \alpha_2 c_2 + \dots + \alpha_m c_m$$

where $c_1 = h f(y,t)$

$$c_2 = h f(y + \nu_{21} c_1, t + \nu_{21} h)$$

$$c_3 = h f(y + \nu_{31} c_1 + \nu_{32} c_2, t + \nu_{31} h + \nu_{32} h)$$

$$c_m = h f\left(y + \sum_{i=1}^{m-1} \nu_{mi} c_i,\ t + h \sum_{i=1}^{m-1} \nu_{mi}\right)$$

Example $m=2$ (2^{nd} order RK)

- keep terms of order h^2

$$y(t+h) = y + hf + \frac{h^2}{2}(f_t + f f_y) \qquad \text{(Taylor)}$$

★ $$y(t+h) = y + \alpha_1 c_1 + \alpha_2 c_2$$

$$c_1 = h f(y,t) \qquad [O(h)]$$

$$c_2 = h f(y + \nu_{21} c_1, t + \nu_{21} h) \qquad [\text{expand}]$$

$$\approx hf + h f_y \nu_{21} c_1 + h f_t \nu_{21} h$$

$$= hf + h^2 f f_y \nu_{21} + h^2 f_t \nu_{21}$$

$$= hf + h^2 \nu_{21}(f f_y + f_t)$$

so $$y(t+h) = y + \alpha_1 \underline{h}f + \alpha_2(\underline{h}f + h^2 \nu_{21}(f f_y + f_t))$$

$$= y + (\alpha_1 + \alpha_2) hf + \alpha_2 \nu_{21} h^2 (f f_y + f_t)$$

compare with Taylor series

$$\alpha_1 + \alpha_2 = 1, \qquad \alpha_2 \nu_{21} = 1/2$$

2 eq$^{\underline{n}}$s, 3 unknowns $(\alpha_1, \alpha_2, \nu_{21})$

In general: m eq$^{\underline{n}}$s and $m + \frac{m(m-1)}{2}$ parameters

- freedom to choose some parameters

Let's choose $\alpha_1 = 0 \rightarrow \alpha_2 = 1, \ \nu_{21} = 1/2$

then $y_{i+1} = y_i + c_2$

This is the Midpoint method

$$c_1 = hf$$
$$c_2 = hf(y + c_1/2, t + h/2)$$

or, e.g. choose $\alpha_1 = \frac{1}{2} \rightarrow \alpha_2 = \frac{1}{2}, \ \nu_{21} = 1$

$$y_{i+1} = y_i + \tfrac{1}{2} c_1 + \tfrac{1}{2} c_2$$

$$c_1 = hf(y_i, t_i)$$
$$c_2 = hf(y_i + c_1, t_i + h)$$

rewrite as

$$\tilde{y}_{i+1} = y_i + hf(y_i, t_i)$$ ← Euler "predictor" step

"Predictor Corrector Method"

$$y_{i+1} = y_i + \tfrac{1}{2}h\left[f(\tilde{y}_{i+1}, t_{i+1}) + f(y_i, t_i)\right]$$

Common algorithm based on $m = 4$

"the" RK method, "Classical RK"

$$y(t+h) = y(t) + \frac{1}{6}(c_1 + 2c_2 + 2c_3 + c_4)$$
$$c_1 = hf(y, t)$$
$$c_2 = hf(y + c_1/2, t + h/2)$$
$$c_3 = hf(y + c_2/2, t + h/2)$$
$$c_4 = hf(y + c_3, t + h)$$

As with RK2, choice of parameters is <u>not</u> unique

Adaptive Time Step

- Algorithm chooses time step to give desired accuracy at each iteration
- can significantly decrease run time

→ at each step, make sure y_{n+1} is good enough
→ h needs to be small enough, but not too small or finding sol$^{\underline{n}}$ will take too long

Simplest case: Euler with step halving

- want to step from t to $t+h$ in

one jump: $y^*_{n+1} = y_n + hf(y_n, t_n) = y_n + hf_n$
$\Delta t = h$

and

two jumps: $y_{n+1} = y_n + \frac{h}{2} f(y_n, t_n) + \frac{h}{2} f(y_{n+1/2}, t_{n+1/2})$
$\Delta t = h/2$

where $y_{n+1/2} = y_n + \frac{h}{2} f(y_n, t_n)$

$$y_{n+1} = y_{n+1/2} + \frac{h}{2} f(y_{n+1/2}, t_{n+1/2})$$

Error in y^*_{n+1} is $\simeq -\frac{h^2}{2} y''(t)$

Error in y_{n+1} is $\simeq -2\left(\frac{h}{2}\right)^2 \frac{y''(t)}{2} = -\frac{1}{2}\frac{h^2}{2} y''(t)$

so $\Delta \equiv y^*_{n+1} - y_{n+1} \approx -\frac{1}{2}\frac{h^2}{2} y''$

is a good estimate of the error.

Now stipulate that we want the error Δ to be smaller than a prescribed limit ϵ

If $|\Delta| < \epsilon$, accept y_{n+1} as the $f^{\underline{n}}$ value

$\rightarrow$ $y(t+h) = y_{n+1}$ (two half-steps)

or, we might get a slightly better estimate by setting $y(t+h) = y_{n+1} - \Delta$ since Δ is an estimate of the truncation error in y_{n+1}.

(This will reduce the truncation error per step to $\Theta(h^3)$.)

Should also consider making h bigger, either by doubling (simple) or by increasing by a factor involving $\left|\frac{\Delta}{\epsilon}\right|$

If $|\Delta| > \epsilon$, we need to decrease h and try again.

option ① $h \rightarrow \frac{h}{2}$, and then reuse $y_{n+1/2}$ as y^*_{n+1} when we retry

option ② decrease h in a way involving $\left|\frac{\Delta}{\epsilon}\right|$

Simple example

$$\frac{dy}{dx} = \cos(x)$$

Note: RHS does not depend on y.

$y(0) = 0$, on $x \in [0, 2\pi]$

$x = 0$, $y = 0$, $dx = 0.0001$, $tol = 0.0005$

```
do while ( x < 2π )

   yfull = y + dx cos(x)

   yhalf = y + 0.5dx ( cos(x) + cos(x+0.5dx))

   if( abs ( yfull - yhalf ) < tol ) then
       y = yhalf
       x = x + dx
       dx = dx * 2.0
   else
       dx = 0.5 * dx
   endif
enddo
```

(we will overshoot endpoint $x_f = 2\pi$)

The same approach can be applied to RK4

$y(x+2h) = y^*_{RK4} + (2h)^5 \cdot C + \mathcal{O}(h^6)$ full step of $\Delta x = 2h$

$y(x+2h) = y_{RK4} + 2\,(h)^5 \cdot C + \mathcal{O}(h^6)$ two steps each of $\Delta x = h$

$\left(C \sim \frac{d^5y}{dx^5} \right)$

$$\Delta \equiv y^*_{RK4} - y_{RK4} \longrightarrow \text{compare to } \epsilon$$

$$= (2 - 2^5)\, h^5 C + \mathcal{O}(h^6) = -30\, h^5 C + \mathcal{O}(h^6)$$

$$\Delta \propto h^5$$

$$\Delta/15 = -2h^5 C$$

If $\Delta < \epsilon$, set $y_{n+1} = y_{RK4}$

or

set $y_{n+1} = y_{RK4} - \frac{\Delta}{15} = y(x+2h) - 2h^5 C + \mathcal{O}(h^6) - \frac{\Delta}{15} + \mathcal{O}(h^6)$

$= y(x+2h) + \mathcal{O}(h^6)$

for a possible improvement (assuming C is constant)

and make h bigger

$$h_{next} \propto h \left(\frac{\epsilon}{\Delta} \right)^{1/5}$$

If $\Delta > \epsilon$, try again with h smaller. Again

$h_{next} \propto h \left(\frac{\epsilon}{\Delta} \right)^{1/5}$ → want h_{next} to produce an error of $\sim \epsilon$.

If the current step size h produces error $\Delta \propto h^n$

want h_{next} to produce an error of $\Delta_{next} \lesssim \epsilon$

expect $h_{new} = h \left(\frac{\Delta_{next}}{\Delta} \right)^{1/n}$, so set $h_{new} = S\, h \left(\frac{\epsilon}{\Delta} \right)^{1/n}$,

where $S < 1$ (0.9, say) is a "safety factor."

How "expensive" is this variable step size approach?

Number of function evaluations is

$$\underset{\text{(full step)}}{4} + \underbrace{4+4}_{\text{2 half steps}} = 12$$

→ actually only 11 since first evaluation of f is used in both y^*_{RK4} and y_{RK4}

"cost" of this variable step size approach is

$$\frac{11}{8} = 1.375$$

since we could get the better accuracy of

y_{RK4} , which takes 8 steps.