2.6 Runge-Kutta Method

solve $y' = f(t, y)$

basic idea : replacing the slope part in Euler

$$y_{n+1} = y_n + \underbrace{f(t_n, y_n)}_{\text{slope}} h$$

Runge-Kutta order 4 (RK 4) :

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)h$$

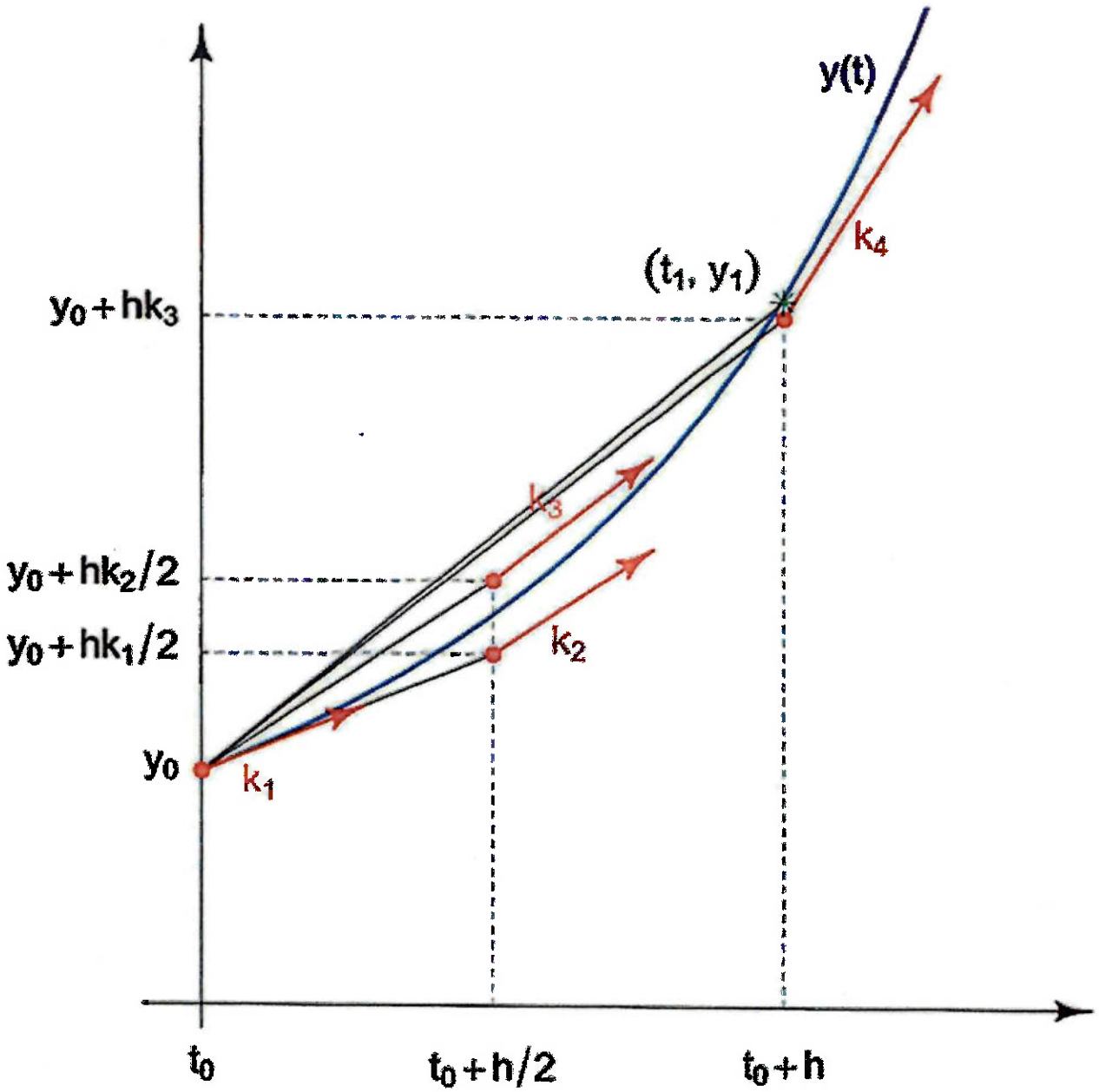
$k_1 = f(t_n, y_n)$ slope at beginning (Euler)

$k_2 = f(t_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1 h)$ slope at mid pt, using Euler (k_1) to get there

$k_3 = f(t_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2 h)$ slope at mid pt, using k_2 to get there

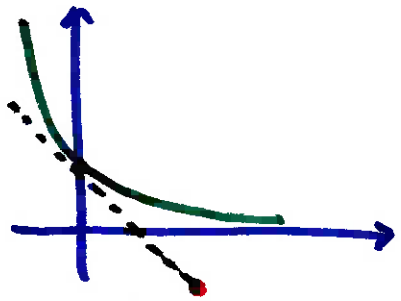
$k_4 = f(t_n + h, y_n + k_3 h)$ slope at end, using k_3 to get there

$\frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$ is weighted avg. of slopes


y(t)
(t_1, y_1)
k_4
$y_0 + hk_3$
k_3
$y_0 + hk_2/2$
k_2
$y_0 + hk_1/2$
y_0
k_1
t_0
$t_0 + h/2$
$t_0 + h$

advantages over Euler :

Stability



$y' = -10y$

could blow up if step is too big

RK4 uses two midpt slopes and the "corrects" the error w/ large h, so the situation above is less likely to happen

accuracy:

error in Euler — global error is proportional to h

in Imp. Euler (RK2) — " " " h^2

in RK4 — " " " h^4

half the step size → reduce error by factor of 16

Example $y' = 1 - t + 4y \quad y(0) = 1$

$h = 0.1$ to estimate $y(0.1)$

$t_0 = 0$, $y_0 = 1$ } given

$t_1 = t_0 + h = 0.1$ (target)

$k_1 = f(t_0, y_0) = 1 - (0) + 4(1) = 5$

$k_2 = f(t_0 + \frac{1}{2}h, y_0 + \frac{1}{2}k_1 h)$

$= 1 - (0 + 0.05) + 4(1 + \frac{1}{2} \cdot 5 \cdot 0.1) = 5.95$

$k_3 = f(t_0 + \frac{1}{2}h, y_0 + \frac{1}{2}k_2 h)$

$= 1 - (0 + 0.05) + 4(1 + \frac{1}{2} \cdot 5.95 \cdot 0.1) = 6.14$

$k_4 = f(t_0 + h, y_0 + k_3 h)$

$= 1 - (0 + 0.1) + 4(1 + 6.14 \cdot 0.1) = 7.356$

$y_1^* = y_0 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)h$

$= 1.60893$

How good? RK4 w/ $h=0.1 \rightarrow 1.60893$ error 1.085×10^{-4}

true value $\rightarrow 1.60904$

RK4 w/ $h=0.05 \rightarrow 1.609034$ error 8×10^{-6}

(reduction of by factor of 14)

Euler w/ $h=0.1 \rightarrow 1.5$

$h=0.01 \rightarrow 1.59529$ error 1×10^{-2}

$h=0.005 \rightarrow 1.60206$ error 7×10^{-3}

(50 steps)

RK2 w/ $h=0.1 \rightarrow 1.595$

w/ $h=0.01 \rightarrow 1.60886$

in this situation, 1 RK 4 step beats all w/ same h

beyond RK 4 (4th order), the math gets messy

common ones: RK 4/5

Runge-Kutta-Fehlberg (RKF45)

The RKF45 method uses six stages to calculate both a 4th-order and a 5th-order approximation, allowing for adaptive step-size control.

$$k_1 = f(t_n, y_n)$$

$$k_2 = f(t_n + \frac{1}{4}h, y_n + \frac{1}{4}hk_1)$$

$$k_3 = f(t_n + \frac{3}{8}h, y_n + \frac{3}{32}hk_1 + \frac{9}{32}hk_2)$$

$$k_4 = f(t_n + \frac{12}{13}h, y_n + \frac{1932}{2197}hk_1 - \frac{7200}{2197}hk_2 + \frac{7296}{2197}hk_3)$$

$$k_5 = f(t_n + h, y_n + \frac{439}{216}hk_1 - 8hk_2 + \frac{3680}{513}hk_3 - \frac{845}{4104}hk_4)$$

$$k_6 = f(t_n + \frac{1}{2}h, y_n - \frac{8}{27}hk_1 + 2hk_2 - \frac{3544}{2565}hk_3 + \frac{1859}{4104}hk_4 - \frac{11}{40}hk_5)$$

Final Estimates

The 4th-order estimate and the 5th-order estimate are calculated as follows:

$$y_{n+1} = y_n + h\left(\frac{25}{216}k_1 + \frac{1408}{2565}k_3 + \frac{2197}{4104}k_4 - \frac{1}{5}k_5\right)$$

$$\hat{y}_{n+1} = y_n + h\left(\frac{16}{135}k_1 + \frac{6656}{12825}k_3 + \frac{28561}{56430}k_4 - \frac{9}{50}k_5 + \frac{2}{55}k_6\right)$$

Matlab ode45 uses Dormand-Prince 4/5-order

compares y_{n+1} from 4th vs 5th

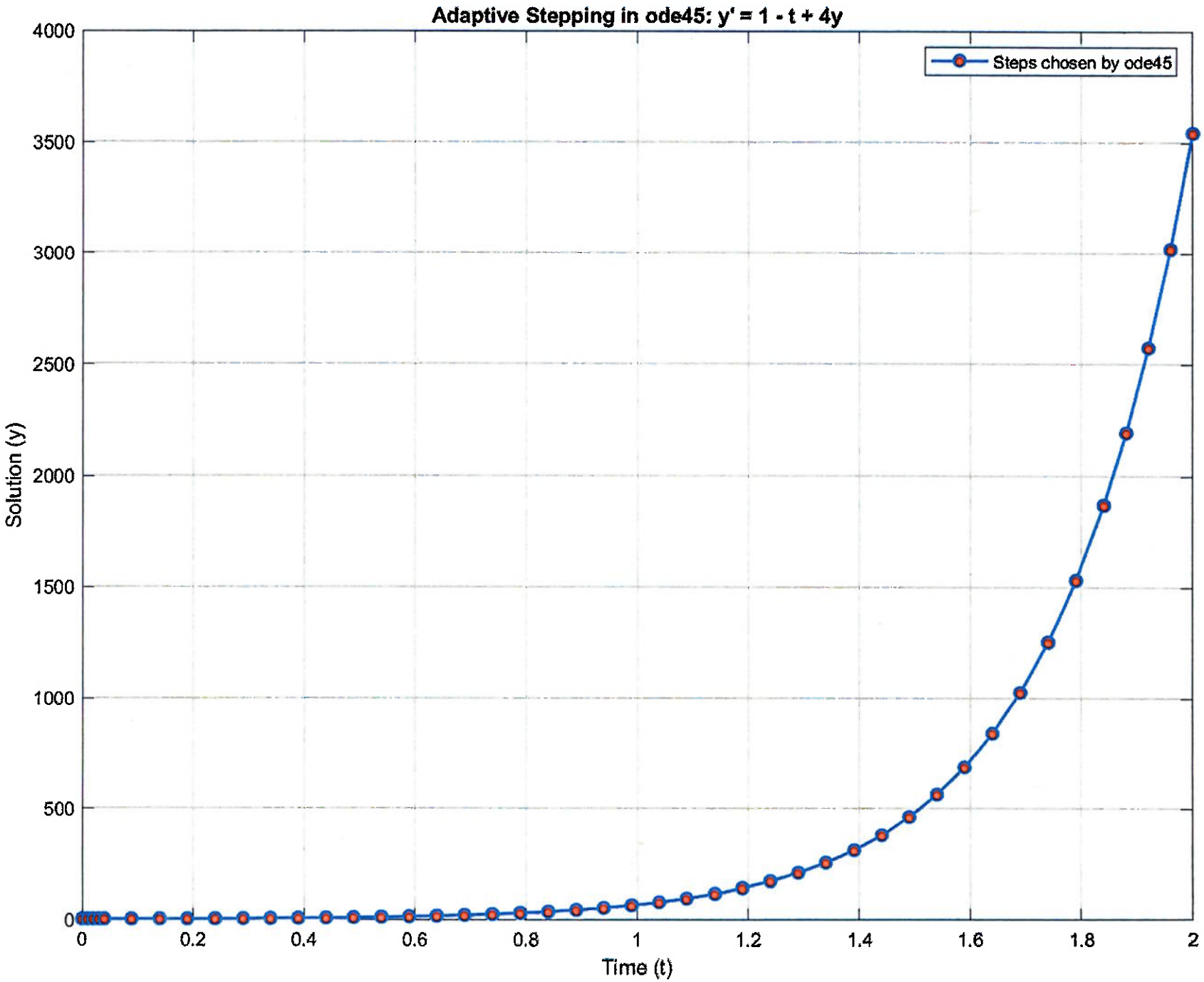
if |difference| is small → h is ok → moves on

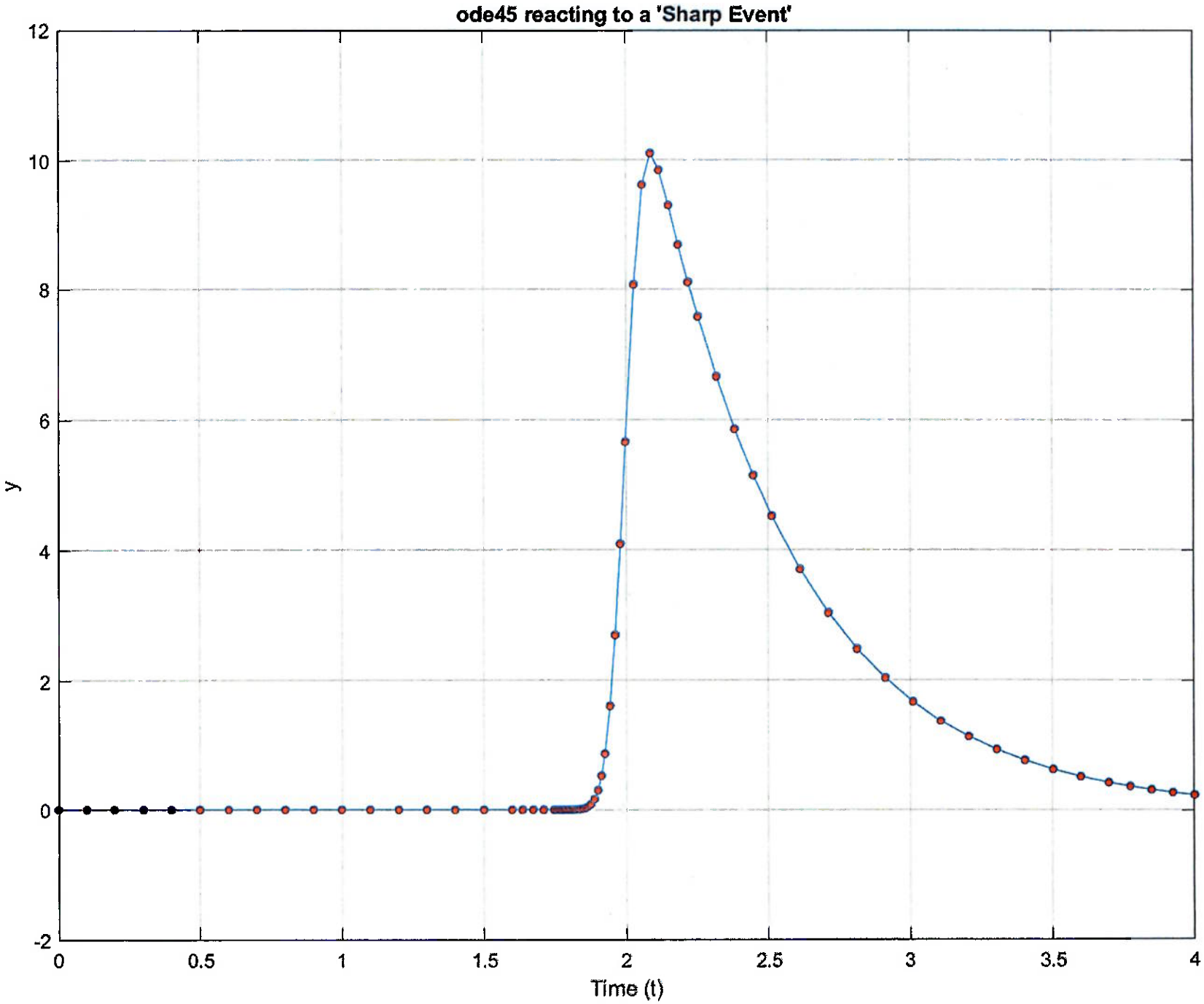
if |difference| > tolerance, shrink h, repeat

"adaptive step size"

straight portion: big h ok

windy portion: need small h


Adaptive Stepping in ode45: y' = 1 - t + 4y
Steps chosen by ode45
Solution (y)
Time (t)
0
500
1000
1500
2000
2500
3000
3500
4000
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2


ode45 reacting to a 'Sharp Event'
12
10
8
6
4
2
0
-2
y
0
0.5
1
1.5
2
2.5
3
3.5
4
Time (t)