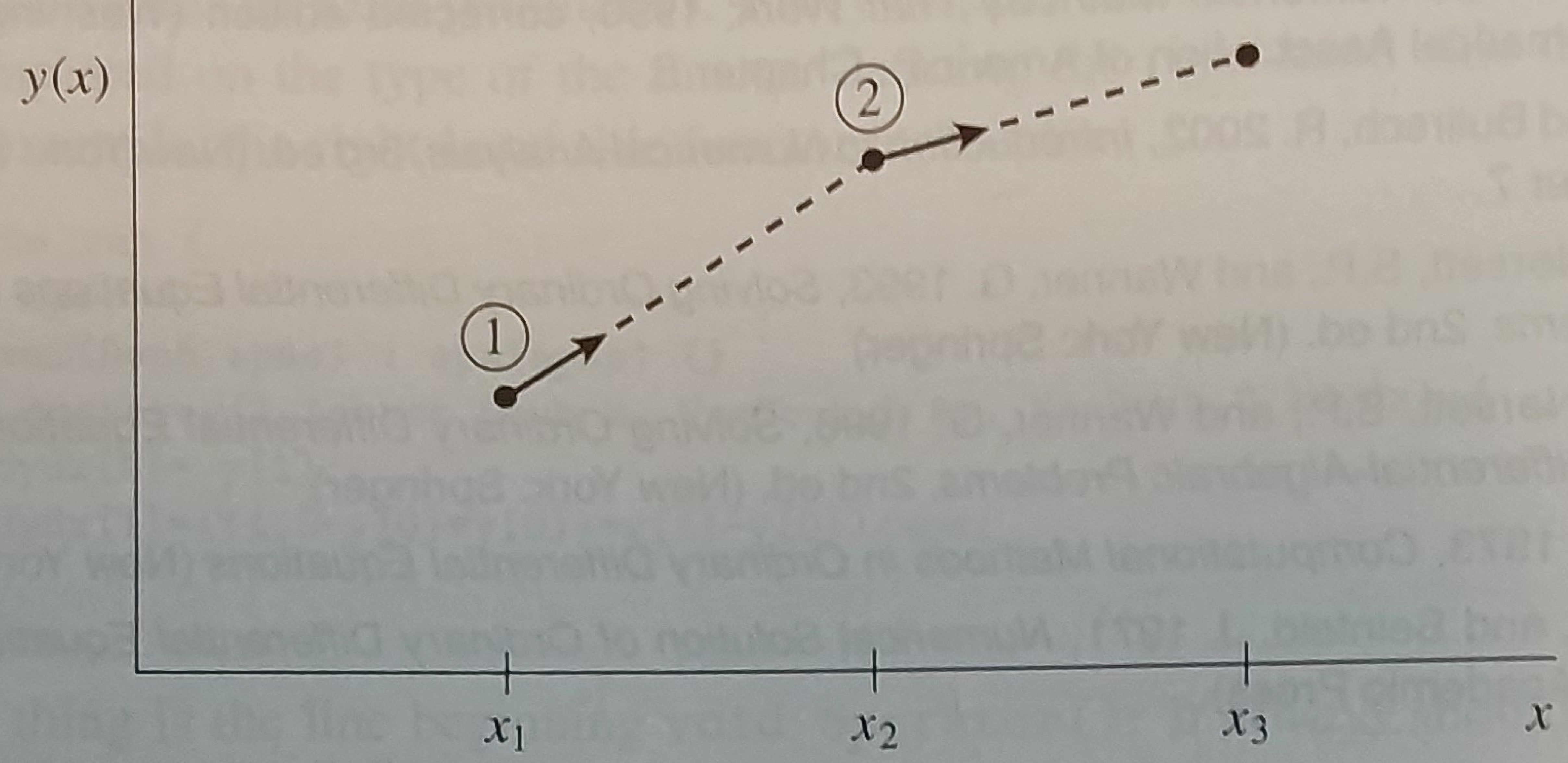


**Figure 17.1.1.** Euler's method. In this simplest (and least accurate) method for integrating an ODE, the derivative at the starting point of each interval is extrapolated to find the next function value. The method has first-order accuracy.

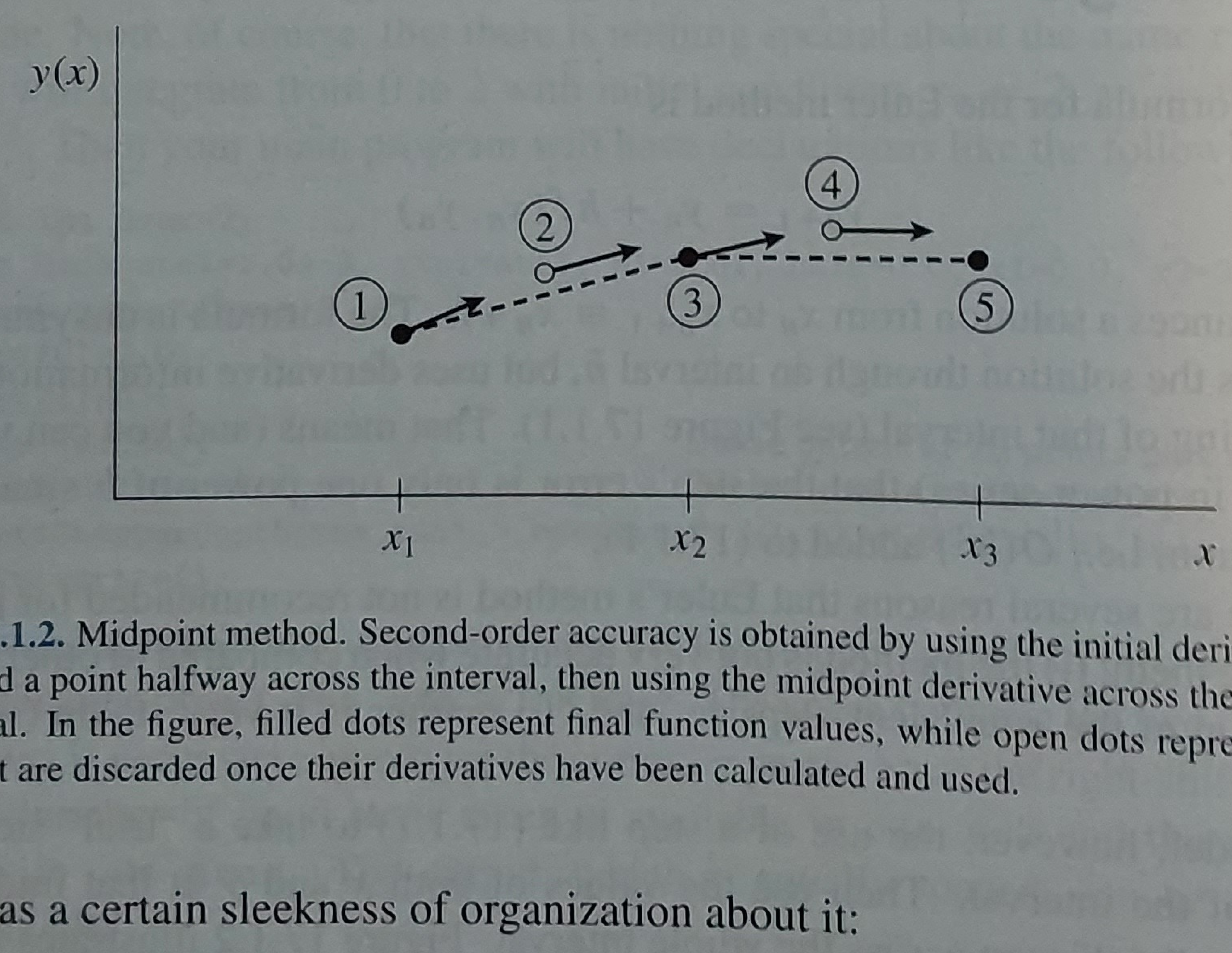


**Figure 17.1.2.** Midpoint method. Second-order accuracy is obtained by using the initial derivative at each step to find a point halfway across the interval, then using the midpoint derivative across the full width of the interval. In the figure, filled dots represent final function values, while open dots represent function values that are discarded once their derivatives have been calculated and used.

which has a certain sleekness of organization about it:

$$
\begin{aligned}
k_1 &= hf(x_n, y_n) \\
k_2 &= hf(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}k_1) \\
k_3 &= hf(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}k_2)
\end{aligned}
\qquad (17.1.3)
$$

# ~~Taylor~~ Series

## Rolle's Theorem

$f$ defined + continuous on $[a,b]$ and differentiable in $(a,b)$.

Let $f(a) = f(b) = 0$

Then $\exists c \in (a,b)$ where $f'(c) = 0$

**MVT** $\exists c \in (a,b)$ s.t.

$$f(b) - f(a) = f'(c)(b-a)$$

**EMVT** special $\exists c_2 \in (a,b)$ st.

$$f(b) = f(a) + f'(a)(b-a) + \frac{1}{2} f''(c_2)(b-a)^2$$

**EMVT** general Let $f(x)$ and its first $n-1$ derivatives, $f'(x), \dots, f^{(n-1)}(x)$ be continuous on $[a,b]$ and $f^{(n)}(x)$ exists at least in $(a,b)$.

Then $\exists c_n \in (a,b)$ st.

$$f(b) = f(a) + f'(a)(b-a) + \dots + \frac{1}{(n-1)!} f^{(n-1)}(a)(b-a)^{n-1} + \frac{1}{n!} f^{(n)}(c_n)(b-a)^n.$$

Taylor series expansion of $f(x)$ about $x = a$

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \dots + \frac{f^{(n)}(a)}{n!}(x-a)^n + R_n(x,a).$$

**Remainder.**

$$\boxed{R_n(x,a) = \int_a^x \frac{(x-t)^n}{n!} f^{(n+1)}(t)\,dt \quad \text{Taylor}}$$

$$\boxed{R_n(x,a) = \frac{f^{(n+1)}(c)}{(n+1)!}(x-a)^{n+1} \quad \text{for } c \in (a,x) \quad \text{Lagrange}}$$

$$\frac{dy_i(t)}{dt} = f_i(t, \vec{y}) \qquad \text{where } \vec{y} = (y_1, y_2, \ldots, y_N)$$
$$i = 1, \ldots N.$$

Euler:

$$\boxed{\vec{y}_{n+1} = \vec{y}_n + h f(t_n, \vec{y}_n) + O(h^2)} \qquad \text{first-order}$$

Advances $t_n \to t_{n+1} \equiv t_n + h$

RK2
or midpoint

$$\vec{k}_1 = h f(t_n, \vec{y}_n)$$
$$\vec{k}_2 = h f(t_n + \tfrac{h}{2}, \vec{y}_n + \tfrac{1}{2}\vec{k}_1)$$
$$\boxed{\vec{y}_{n+1} = \vec{y}_n + \vec{k}_2 + O(h^3)} \qquad \text{second-order}$$

RK4

$$\vec{k}_3 = h f(t_n + \tfrac{h}{2}, \vec{y}_n + \tfrac{1}{2}\vec{k}_2)$$
$$\vec{k}_4 = h f(t_n + h, \vec{y}_n + \vec{k}_3)$$
$$\boxed{\vec{y}_{n+1} = \vec{y}_n + \tfrac{1}{6}(\vec{k}_1 + 2\vec{k}_2 + 2\vec{k}_3 + \vec{k}_4) + O(h^5)} \qquad \text{fourth-order}$$

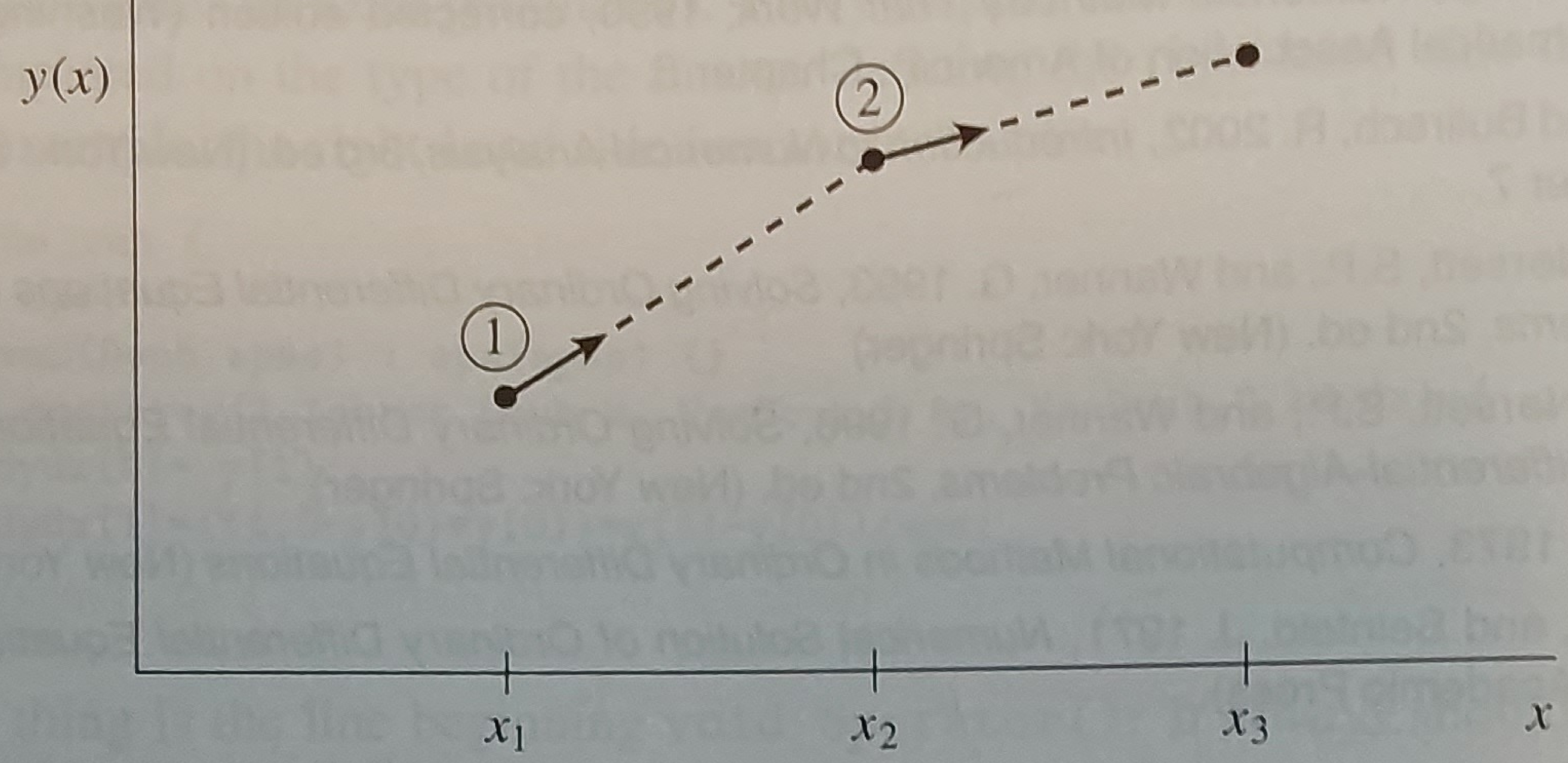


**Figure 17.1.1.** Euler's method. In this simplest (and least accurate) method for integrating an ODE, the derivative at the starting point of each interval is extrapolated to find the next function value. The method has first-order accuracy.

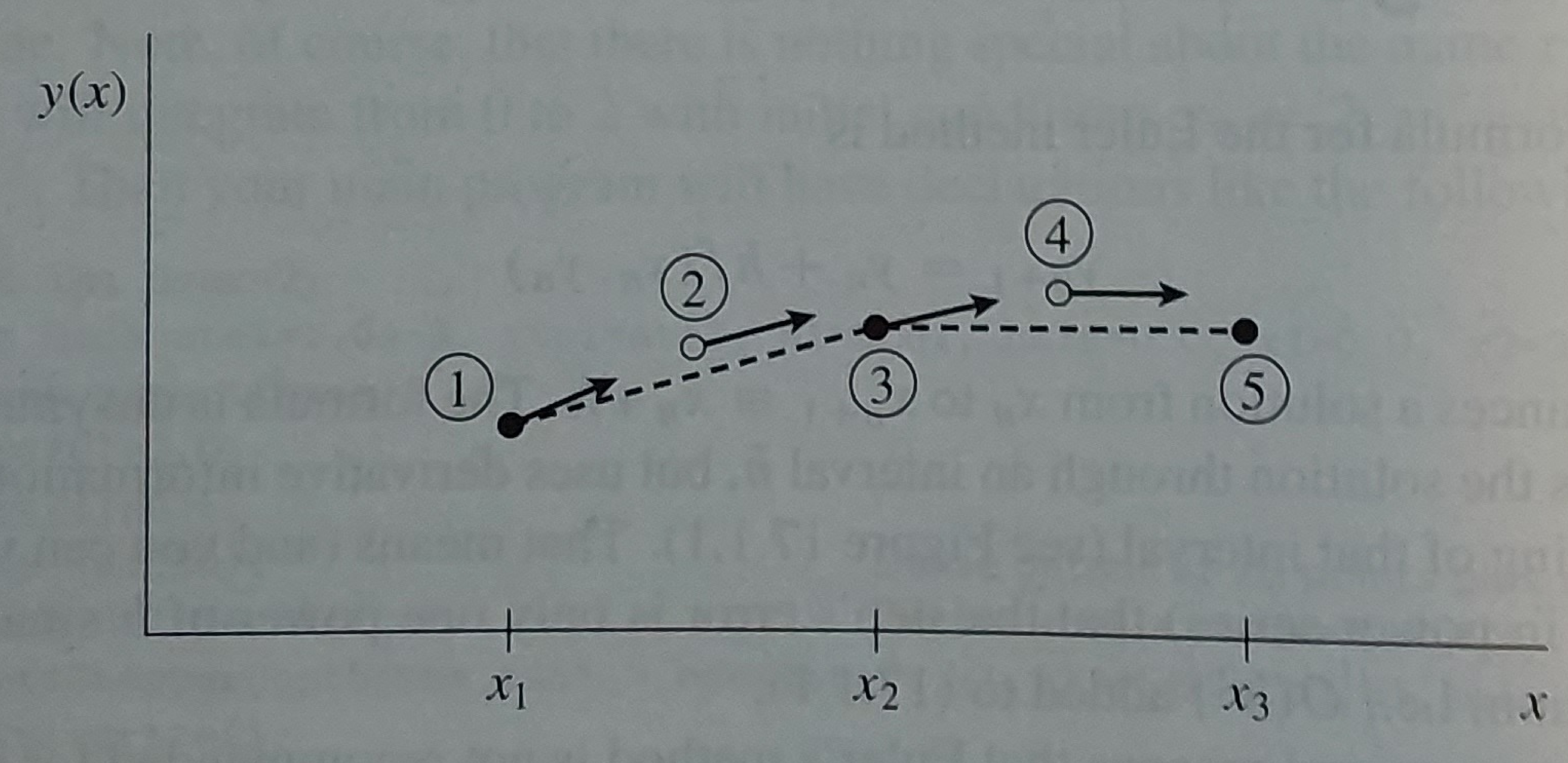


**Figure 17.1.2.** Midpoint method. Second-order accuracy is obtained by using the initial derivative at each step to find a point halfway across the interval, then using the midpoint derivative across the full width of the interval. In the figure, filled dots represent final function values, while open dots represent function values that are discarded once their derivatives have been calculated and used.

which has a certain sleekness of organization about it:

$$
\begin{aligned}
k_1 &= hf(x_n, y_n) \\
k_2 &= hf(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}k_1) \\
k_3 &= hf(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}k_2) \qquad (17.1.3) \\
k_4 &= hf(x_n + h, y_n + k_3)
\end{aligned}
$$

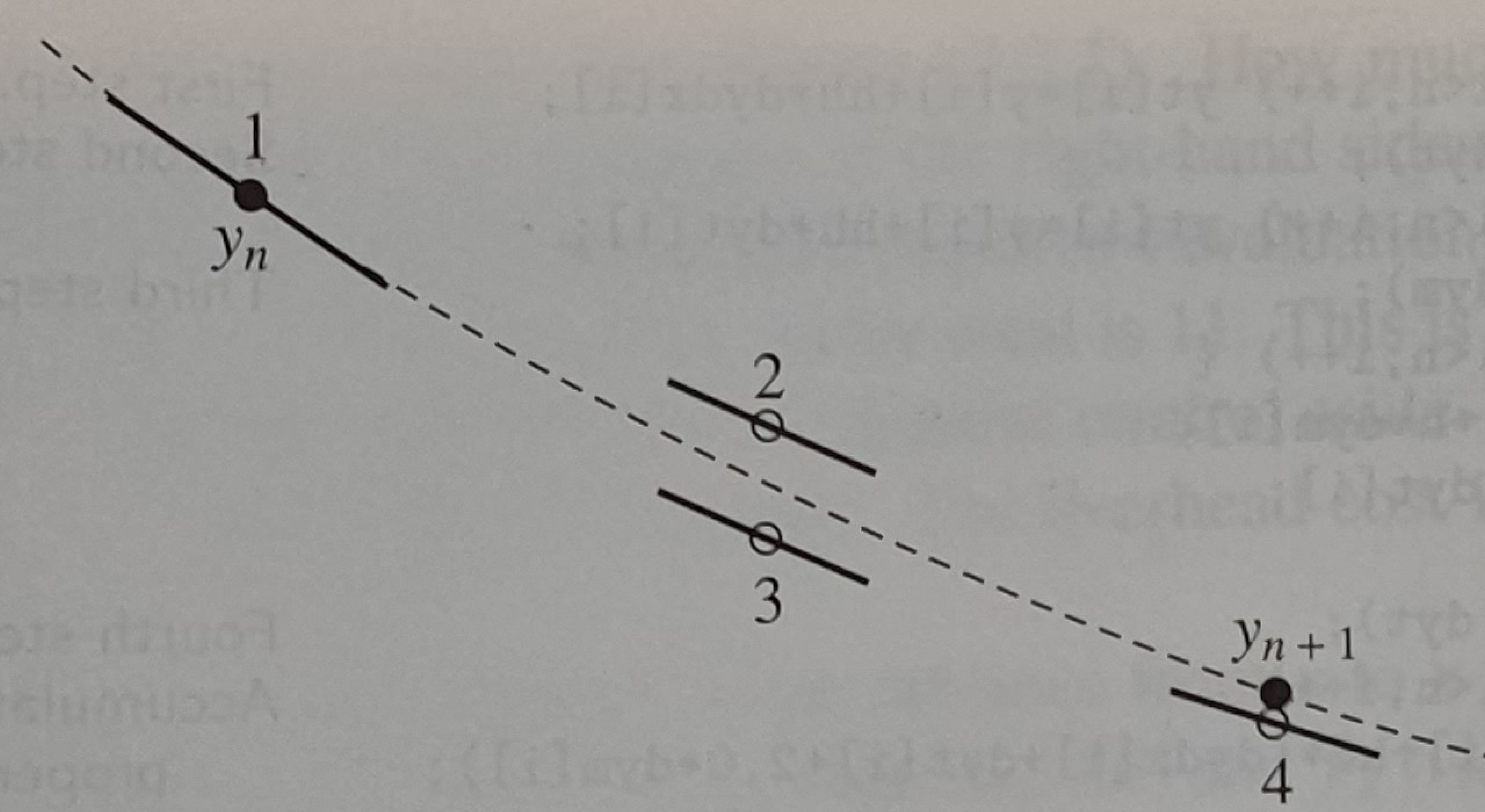


**Figure 17.1.3.** Fourth-order Runge-Kutta method. In each step the derivative is evaluated four times: once at the initial point, twice at trial midpoints, and once at a trial endpoint. From these derivatives the final function value (shown as a filled dot) is calculated. (See text for details.)

on ODE integrators, but the last word as well. In fact, you can get pretty far on this old workhorse, especially if you combine it with an adaptive stepsize algorithm. Keep in mind, however, that the old workhorse's last trip may well be to take you to the poorhouse: Newer Runge-Kutta methods are *much* more efficient, and Bulirsch-Stoer or predictor-corrector methods can be even more efficient for problems where very high accuracy is a requirement. Those methods are the high-strung racehorses. Runge-Kutta is for ploughing the fields. However, even the old workhorse is more nimble with new horseshoes. In §17.2 we will give a modern implementation of a Runge-Kutta method that is quite competitive as long as very high accuracy is not required. An excellent discussion of the pitfalls in constructing a good Runge-Kutta code is given in [3].

Here is the routine `rk4` for carrying out one classical Runge-Kutta step on a

## Adaptive Stepsize Control.

$x \rightarrow x + 2h$   exact solution $y(x+2h)$

Two approximate solutions, $y_1$ with one step of $2h$

and $y_2$ with two steps of $h$

$$y(x+2h) = y_1 + (2h)^5 \phi + O(h^6) \qquad (1)$$

$$y(x+2h) = y_2 + 2(h^5)\phi + O(h^6) \qquad (2)$$

error on each step is $h^5 \phi$.

$$\phi \sim \frac{y^{(5)}(x)}{5!}$$

Indication of truncation error,

$$\Delta \equiv y_2 - y_1$$

Keep this small enough by adjusting $h$

We can also solve (1), (2), (eliminating $\phi$) giving.

$$y(x+2h) = y_2 + \frac{\Delta}{15} + O(h^6)$$

local extrapolation

for $c \in (a, x)$

Lagrange

(4)

(2) − (1).

$$0 = (y_2 - y_1)\cancel{\phi} - 30h^5\phi + O(h^6)$$

(2)' $\quad y(x+2h) = y_2 + 2h^5\phi + O(h^6)$

(3) $\quad 30h^5\phi = \Delta$

$\Rightarrow \quad (2)' \rightarrow \quad y(x+2h) = y_2 + \dfrac{\Delta}{15} + O(h^6).$