Section 8.1. To be specific, the lines in Step 6 in the Euler algorithm must be replaced by the following:

**Step 6.**

\[
\begin{align*}
    k_1 &= f(t, y) \\
    k_2 &= f(t + 0.5h, y + 0.5hk_1) \\
    k_3 &= f(t + 0.5h, y + 0.5hk_2) \\
    k_4 &= f(t + h, y + hk_3) \\
    y &= y + (h/6) * (k_1 + 2k_2 + 2k_3 + k_4) \\
    t &= t + h
\end{align*}
\]

Note that if \(f\) does not depend on \(y\), then

\[
    k_{n1} = f(t_n), \quad k_{n2} = k_{n3} = f(t_n + h/2), \quad k_{n4} = f(t_n + h),
\]

and Eq. (2) reduces to

\[
y_{n+1} - y_n = \frac{h}{6}[f(t_n) + 4f(t_n + h/2) + f(t_n + h)].
\]

Equation (5) can be identified as Simpson’s (1710–1761) rule for the approximate evaluation of the integral of \(y' = f(t)\). The fact that Simpson’s rule has an error proportional to \(h^4\) is consistent with the local truncation error in the Runge–Kutta formula.

Use the Runge–Kutta method to calculate approximate values of the solution \(y = \phi(t)\) of the initial value problem

\[
y' = 1 - t + 4y, \quad y(0) = 1.
\]

Taking \(h = 0.2\) we have

\[
\begin{align*}
    k_{01} &= f(0, 1) = 5; \quad h_{k01} = 1.0, \\
    k_{02} &= f(0 + 0.1, 1 + 0.5) = 6.9; \quad h_{k02} = 1.38, \\
    k_{03} &= f(0 + 0.1, 1 + 0.69) = 7.66; \quad h_{k03} = 1.532, \\
    k_{04} &= f(0 + 0.2, 1 + 1.532) = 10.928.
\end{align*}
\]

Thus

\[
y_1 = 1 + \frac{0.2}{6}[5 + 2(6.9) + 2(7.66) + 10.928] \\
    = 1 + 1.5016 = 2.5016.
\]

Further results using the Runge–Kutta method with \(h = 0.2, h = 0.1, \) and \(h = 0.05\) are given in Table 8.3.1. Note that the Runge–Kutta method yields a value at \(t = 2\) that differs from the exact solution by only 0.122% if the step size is \(h = 0.1\), and by only 0.00903% if \(h = 0.05\). In the latter case the error is less than one part in ten thousand, and the calculated value at \(t = 2\) is correct to four digits.

For comparison, note that both the Runge–Kutta method with \(h = 0.05\) and the improved Euler method with \(h = 0.025\) require 160 evaluations of \(f\) to reach \(t = 2\). The improved Euler method yields a result at \(t = 2\) that is in error by 1.22%. While this error may be acceptable for some purposes, it is more than 135 times the error yielded by the Runge–Kutta method with comparable computing effort. Note also that the Runge–Kutta method with \(h = 0.2\), or 40 evaluations of \(f\), produces a value at \(t = 2\)