Example 149. Python Let us apply the midpoint method to $y' = y$, $y(0) = 1$.

```
>>> def midpoint(f, x0, y0, xmax, n):
        h = (xmax - x0) / n
        ypoints = [y0]
        for i in range(n):
            y0 = y0 + f(x0+h/2,y0+f(x0,y0)*h/2)*h
            x0 = x0 + h
            ypoints.append(y0)
        return ypoints
```

```
>>> def f(x, y):
        return y
```

The exact solution is $y(x) = e^x$ with $y(1) = e \approx 2.718$.

```
>>> midpoint(f, 0, 1, 1, 4)
```

```
[1, 1.28125, 1.6416015625, 2.103302001953125, 2.6948556900024414]
```

The following numerically confirms that the error in the midpoint method is $O(h^2)$.

```
>>> from math import e
```

```
>>> [midpoint(f, 0, 1, 1, 10**n)[-1] - e for n in range(6)]
```

```
[-0.2182818284590451, -0.004200981850821073, -4.49658990882007e-05,
-4.5270728232793545e-07, -4.530157138304958e-09, -4.530020802917534e-11]
```

Runge–Kutta methods

The midpoint method can be written as:

$$\begin{aligned} x_{n+1} &= x_n + h \\ y_{n+1} &= y_n + K_1 h \\ K_0 &= f(x_n, y_n) \\ K_1 &= f\left(x_n + \frac{h}{2}, y_n + K_0 \frac{h}{2}\right) \end{aligned}$$

Note that replacing the rule by $y_{n+1} = y_n + K_0 h$ results in Euler's method. Indeed, both K_0 and K_1 are approximations of the slope y' that we need for stepping from x_n to $x_{n+1} = x_n + h$.

Adding further such approximations K_i to the mix, one can eliminate further terms in the error expansion and obtain higher order methods known as **Runge–Kutta methods**.

The midpoint method is an example of a Runge–Kutta method of order 2 (but there are others as well).
https://en.wikipedia.org/wiki/Runge%E2%80%93Kutta_methods

Of particular practical importance is the following instance:

Armin Straub
straub@southalabama.edu

(Runge–Kutta method of order 4)

$$\begin{aligned}
x_{n+1} &= x_n + h\\
y_{n+1} &= y_n + \frac{1}{6}(K_0 + 2K_1 + 2K_2 + K_3)h\\
K_0 &= f(x_n, y_n)\\
K_1 &= f\left(x_n + \frac{h}{2}, y_n + K_0\frac{h}{2}\right)\\
K_2 &= f\left(x_n + \frac{h}{2}, y_n + K_1\frac{h}{2}\right)\\
K_3 &= f(x_n + h, y_n + K_2 h)
\end{aligned}$$

Comment. Note how each of K_0, K_1, K_2, K_3 is an approximation of y' on the interval $[x_n, x_{n+1}]$ (with K_0 approximating $y'(x_n)$ and K_3 approximating $y'(x_{n+1})$). By taking the appropriate weighted average, we are able to get an approximation with a higher order.

Advanced comment. Note that the weights (with K_1 and K_2 combined because they both correspond to the midpoint $x_n + h/2$) are the same as in Simpson's rule for numerical integration. That is more than a coincidence. Indeed, if $f(x, y) = f(x)$ does not depend on y, then solving the DE is equivalent to integrating $f(x)$ and the Runge–Kutta method of order 4 turns into Simpson's rule.

Example 150. `Python` Let us implement the Runge–Kutta method of order 4.

```
>>> def runge_kutta4(f, x0, y0, xmax, n):
        h = (xmax - x0) / n
        ypoints = [y0]
        for i in range(n):
            K0 = f(x0,y0)
            K1 = f(x0+h/2, y0+K0*h/2)
            K2 = f(x0+h/2, y0+K1*h/2)
            K3 = f(x0+h, y0+K2*h)
            y0 = y0 + (K0 + 2*K1 + 2*K2 + K3)*h/6
            x0 = x0 + h
            ypoints.append(y0)
        return ypoints
```

First, for comparison with earlier methods, let us apply the method to the IVP $y' = y$, $y(0) = 1$, which has the exact solution $y(x) = e^x$ with $y(1) = e \approx 2.718$.

```
>>> def f(x, y):
        return y
```

```
>>> runge_kutta4(f, 0, 1, 1, 4)
```

```
[1, 1.2840169270833333, 1.648699469036526, 2.1169580259162033, 2.718209939201323]
```

The following convincingly illustrates that the error is indeed $O(h^4)$.

```
>>> from math import e
```

```
>>> [runge_kutta4(f, 0, 1, 1, 10**n)[-1] - e for n in range(6)]
```

```
[-0.009948495125712054, -2.0843238792700447e-06, -2.2464119453502462e-10,
-2.042810365310288e-14, 1.1546319456101628e-14, 6.217248937900877e-15]
```

Pause for a moment to really appreciate how much better these errors are in comparison with Euler's method! Whereas computing 10^5 values with Euler's method resulted in an error of $1.36 \cdot 10^{-5}$, we are now able to obtain an error of $2.04 \cdot 10^{-14}$ with only 10^3 values.

Armin Straub
straub@southalabama.edu

As a second example, let us consider as in Example 147 the IVP $y' = \cos(x)y$ with $y(0) = 1$, which has the exact solution $y(x) = e^{\sin(x)}$ with $y(2) = e^{\sin(2)} \approx 2.48258$.

```
>>> from math import e, cos, sin
```

```
>>> def f_cosx_y(x, y):
        return cos(x)*y
```

```
>>> runge_kutta4(f_cosx_y, 0, 1, 2, 4)
```

```
[1, 1.614859377441316, 2.3191895982789603, 2.7107641474177457, 2.481902218021582]
```

The following again convincingly illustrates that the error is indeed $O(h^4)$.

```
>>> [runge_kutta4(f_cosx_y, 0, 1, 2, 10**n)[-1] - e**sin(2) for n in range(5)]
```

```
[-0.12999578105593113, -1.726387102785054e-05, -1.6494263732624859e-09,
-1.6431300764452317e-13, 3.419486915845482e-13]
```

Important comment. Note that, in contrast to Example 147, we did not have to compute partial derivatives of $f(x, y) = \cos(x)y$ by hand. Instead, we were able to simply use $\cos(x)y$ in our `runge_kutta4` function.

Applying Richardson extrapolation to Euler's method

In this section we illustrate that Richardson extrapolation can be applied repeatedly to increase the order as desired.

Example 151. Suppose that $A(h)$ is an approximation of some quantity A^* of order 1. Combine the approximations $A(1)=2$, $A\left(\frac{1}{2}\right)=\frac{9}{4}$ and $A\left(\frac{1}{3}\right)=\frac{64}{27}$ to an approximation of order 3.

Solution. Since $A(h)$ is an approximation of order 1, we assume that $A(h)=A^*+C_1h+C_2h^2+O(h^3)$ for some (unknown) constants C_1, C_2.

Correspondingly, $A(1)\approx A^*+C_1+C_2$, $A\left(\frac{1}{2}\right)\approx A^*+\frac{1}{2}C_2+\frac{1}{4}C_2$ and $A\left(\frac{1}{3}\right)\approx A^*+\frac{1}{3}C_1+\frac{1}{9}C_2$.

We want to combine these three in such a way that we get an approximation of A^* with C_1 and C_2 eliminated. We can do this in different (but ultimately equivalent) ways:

(1) If we take the combination $\alpha A(1)+\beta A\left(\frac{1}{2}\right)+\gamma A\left(\frac{1}{3}\right)$, then we get:

$$\alpha A(1)+\beta A\left(\frac{1}{2}\right)+\gamma A\left(\frac{1}{3}\right)\approx(\alpha+\beta+\gamma)A^*+\left(\alpha+\frac{\beta}{2}+\frac{\gamma}{3}\right)C_1+\left(\alpha+\frac{\beta}{4}+\frac{\gamma}{9}\right)C_2$$

Since we want $1\cdot A^*+0\cdot C_1+0\cdot C_2$, we therefore get three equations for the three unknowns α,β,γ:

$$\begin{aligned}\alpha+\beta+\gamma &= 1\\ \alpha+\frac{\beta}{2}+\frac{\gamma}{3} &= 0\\ \alpha+\frac{\beta}{4}+\frac{\gamma}{9} &= 0\end{aligned}$$

We can solve (this is some work—do it!) these equations to find $\alpha=\frac{1}{2}$, $\beta=-4$, $c=\frac{9}{2}$.

Therefore, our combined approximation is $\frac{1}{2}A(1)-4A\left(\frac{1}{2}\right)+\frac{9}{2}A\left(\frac{1}{3}\right)=\frac{1}{2}\cdot 2-4\cdot\frac{9}{4}+\frac{9}{2}\cdot\frac{64}{27}=\frac{8}{3}$.

(2) As in a single step of Richardson extrapolation, we can combine any two approximations to get one of higher order (meaning that C_1 is eliminated). For instance:

$$\begin{aligned}\text{Approx}_1 := \frac{2A\left(\frac{1}{2}\right)-A(1)}{2-1} &\approx \frac{2\left(A^*+\frac{1}{2}C_2+\frac{1}{4}C_2\right)-(A^*+C_1+C_2)}{2-1}=A^*-\frac{1}{2}C_2\\ \text{Approx}_2 := \frac{3A\left(\frac{1}{3}\right)-A(1)}{3-1} &\approx \frac{3\left(A^*+\frac{1}{3}C_1+\frac{1}{9}C_2\right)-(A^*+C_1+C_2)}{3-1}=A^*-\frac{1}{3}C_2\end{aligned}$$

We combine these two approximations into $\frac{3\text{Approx}_2-2\text{Approx}_1}{3-2}\approx A^*$ where C_2 is eliminated:

$$\frac{3\text{Approx}_2-2\text{Approx}_1}{3-2}=3\left(\frac{3}{2}A\left(\frac{1}{3}\right)-\frac{1}{2}A(1)\right)-2\left(2A\left(\frac{1}{2}\right)-A(1)\right)=\frac{9}{2}A\left(\frac{1}{3}\right)-4A\left(\frac{1}{2}\right)+\frac{1}{2}A(1)$$

This is the same combined approximation as above so that we again get $\frac{8}{3}$.

[Instead of spelling out the combination, we could also use $\text{Approx}_1=\frac{5}{2}$ and $\text{Approx}_2=\frac{23}{9}$ so that the final combination is, once more, $3\text{Approx}_2-2\text{Approx}_1=3\cdot\frac{23}{9}-2\cdot\frac{5}{2}=\frac{8}{3}$.]

Comment. The numbers above are not random. Instead $A(h)$ is an approximation of $e\approx 2.718$ that is obtained by applying the Euler method to $y'=y$, $y(0)=1$. As explained in Example 144, we find that, using n steps (which means that $h=\frac{1-0}{n}=\frac{1}{n}$), our approximation for $y(1)$ is $\left(1+\frac{1}{n}\right)^n$. Using $n=1,2,3$, we find

$$A(1)=2,\quad A\left(\frac{1}{2}\right)=\frac{9}{4}=2.25,\quad A\left(\frac{1}{3}\right)=\frac{64}{27}\approx 2.370.$$