Example 149. 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.

As a second example, let us consider as in Example 146 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 146, 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

The next example illustrates that Richardson extrapolation can be applied repeatedly to increase the order as desired.

Example 150. 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, $A(1)\approx A^*+C_1+C_2$, $A\left(\frac{1}{2}\right)\approx A^*+\frac{1}{2}C_1+\frac{1}{4}C_2$ 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:

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

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

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

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 $\mathrm{Approx}_1=\frac{5}{2}$ and $\mathrm{Approx}_2=\frac{23}{9}$ so that the final combination is, once more, $3\mathrm{Approx}_2-2\mathrm{Approx}_1=3\cdot\frac{23}{9}-2\cdot\frac{5}{2}=\frac{8}{3}$.]

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 142, 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

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

As an example, we apply repeated Richardson extrapolation to Euler's method (which is order 1) until we have an approximation of order 4. While this is comparable to RK4, we observe that the latter has less numerical issues.

Example 151. Suppose that $A(h)$ is an approximation of A^* of order 1.

(a) Combine $A(h)$ and $A(h/2)$ to get a new approximation $B(h)$ of order 2 (or higher).

(b) Combine $B(h)$ and $B(h/2)$ to get a new approximation $C(h)$ of order 3 (or higher).

(c) Spell out the approximation $C(h)$ in terms of the original approximation.

(d) By continuing the above, produce an order 4 approximation.

Solution.

(a) Since $A(h)$ is of order 1, we have $A(h)=A^*+Ch+O(h^2)$ as well as $A\left(\frac{h}{2}\right)=A^*+\frac{1}{2}Ch+O(h^2)$.
It follows that $\frac{2A\left(\frac{h}{2}\right)-A(h)}{2-1}=A^*+O(h^2)$. This is just Richardson extrapolation.
In conclusion, $B(h)=2A\left(\frac{h}{2}\right)-A(h)$ is an approximation of A^* of order (at least) 2.

(b) We assume that $B(h)=A^*+Ch^2+O(h^3)$ (for a new constant C). Hence, $B\left(\frac{h}{2}\right)=A^*+\frac{1}{4}Ch^2+O(h^3)$.
It follows that $C(h):=\frac{4B\left(\frac{h}{2}\right)-B(h)}{4-1}=A^*+O(h^3)$. Again, this is just another Richardson extrapolation.

(c) We have:

$$\begin{aligned} C(h) &= \frac{4}{3}B\left(\frac{h}{2}\right)-\frac{1}{3}B(h)=\frac{4}{3}\left(2A\left(\frac{h}{4}\right)-A\left(\frac{h}{2}\right)\right)-\frac{1}{3}\left(2A\left(\frac{h}{2}\right)-A(h)\right) \\ &= \frac{8}{3}A\left(\frac{h}{4}\right)-2A\left(\frac{h}{2}\right)+\frac{1}{3}A(h) \end{aligned}$$

(d) Since $C(h)$ is an order 3 approximation, the following is an order 4 (or higher) approximation:

$$\frac{8C\left(\frac{h}{2}\right)-C(h)}{8-1}=\frac{64}{21}A\left(\frac{h}{8}\right)-\frac{8}{3}A\left(\frac{h}{4}\right)+\frac{2}{3}A\left(\frac{h}{2}\right)-\frac{1}{21}A(h)$$

Comment. Instead of going via the intermediate approximation $B(h)$, we could have also proceeded as in the first part of the next example: starting with $A(h)=A^*+C_1h+C_2h^2+O(h^3)$, we would look for a combination $\alpha A(h)+\beta A\left(\frac{h}{2}\right)+\gamma A\left(\frac{h}{4}\right)$ such that C_1 and C_2 get eliminated. This alternative approach is probably a bit more tedious to spell out but has the advantage that we can combine any three approximations.

Comment. When stating that $A(h)$ is an approximation of A^* of order 1, we impliciyly made the stronger assumption that $A(h)$ takes the form $A(h)=A^*+C_1h+C_2h^2+C_3h^3+O(h^4)$. This is not necessarily the case (there could, for instance, be error terms of the form $h^{3/2}$) in general but it is a reasonable assumption for many standard situations.

Example 152. Python Let us use the Euler method as implemented in Example 143. We apply it to the IVP $y' = \frac{2}{5}y^2$ with $y(0) = 2$, which has the exact solution $y(x) = \frac{10}{5-4x}$ (note that this has a singularity at $x = \frac{5}{4}$) with $y(1) = 10$.

```
>>> def f(x, y):
        return 2/5*y**2
```

The following confirms that the error in Euler's method is $O(h)$.

```
>>> [euler(f, 0, 2, 1, 10**n)[-1] - 10 for n in range(7)]
```

```
[-6.4, -3.153039180044801, -0.5786318726117212, -0.06365106772819829,
-0.006430401713281242, -0.0006437015789977352, -6.437677910753337e-05]
```

If $A(h)$ is Euler (which is order 1) then $B(h) = 2A\left(\frac{h}{2}\right) - A(h)$ is an approximation of order 2:

```
>>> [2*euler(f, 0, 2, 1, 2*10**n)[-1]-euler(f, 0, 2, 1, 10**n)[-1] - 10 for n in range(7)]
```

```
[-4.864000000000001, -1.0300367298617736, -0.03059704698782717,
-0.0003608641572583376, -3.672563401835305e-06, -3.678901450143712e-08,
-3.710773910370335e-10]
```

And we know how to further increase the order. The following is order 3:

```
>>> def A(n):
        return euler(f, 0, 2, 1, n)[-1]
```

```
>>> [8/3*A(4*10**n)-2*A(2*10**n)+1/3*A(10**n) - 10 for n in range(7)]
```

```
[-3.261606554763264, -0.23097082059404528, -0.0009539131127027645,
-1.1759506524100516e-06, -1.2013980921210532e-09, -4.835243316847482e-12,
2.644640062499093e-11]
```

Observe that the last approximation reveals numerical issues. These worsen if we push things to order 4:

```
>>> [64/21*A(8*10**n)-8/3*A(4*10**n)+2/3*A(2*10**n)-1/21*A(10**n) - 10 for n in range(7)]
```

```
[-1.7873721196484613, -0.03201840214106966, -1.63900245677695e-05,
-2.0771775410821647e-09, 1.0391687510491465e-12, 5.448086426440568e-12,
-2.8705926524708048e-11]
```

For comparison, let us also apply RK4 which is of order 4 as well:

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

```
[-1.6372498407533236, -0.0029737693332965165, -3.5183410496131273e-07,
-3.524114333686157e-11, -6.394884621840902e-14]
```

Here, the numerical issue with the final approximation is less pronounced.