Example 143. Spell out the Taylor method of order 3 for numerically solving the IVP

$$y' = \cos(x)y, \quad y(0) = 1.$$

Solution. The Taylor method of order 3 is based on the Taylor expansion

$$y(x+h) = y(x) + y'(x)h + \frac{1}{2}y''(x)h^2 + \frac{1}{6}y'''(x)h^3 + O(h^4),$$

where we have a local truncation error of $O(h^4)$ so that the global error will be $O(h^3)$.
From the DE we know that $y'(x) = \cos(x)y$, which is $f(x,y)$. As in the previous example, we differentiate this to obtain

$$\begin{aligned} y''(x) &= \frac{\mathrm{d}}{\mathrm{d}x}\cos(x)y = -\sin(x)y + \cos(x)y' = -\sin(x)y + \cos^2(x)\,y \\ &= (-\sin(x) + \cos^2(x))y, \end{aligned}$$

which is $f'(x,y)$. Once more differentiating this, we obtain

$$\begin{aligned} y'''(x) &= \frac{\mathrm{d}}{\mathrm{d}x}(-\sin(x) + \cos^2(x))y \\ &= (-\cos(x) - 2\cos(x)\sin(x))y + (-\sin(x) + \cos^2(x))y' \\ &= (\cos^2(x) - 3\sin(x) - 1)\cos(x)y, \end{aligned}$$

which is $f''(x,y)$. Hence, the Taylor method of order 3 takes the form:

$$\begin{aligned} y_{n+1} &= y_n + f(x_n,y_n)h + \frac{1}{2}f'(x_n,y_n)h^2 + \frac{1}{6}f''(x_n,y_n)h^3 \\ &= y_n + \cos(x_n)y_n\,h + \frac{1}{2}((-\sin(x_n) + \cos^2(x_n))y_n)h^2 + \frac{1}{6}((\cos^2(x_n) - 3\sin(x_n) - 1)\cos(x_n)y_n)h^3 \end{aligned}$$

Comment. The exact solution of this IVP is $y(x) = e^{\sin(x)}$. We use this in the next example for comparison.

Example 144. Python Let us use Python to approximate the solution $y(x)$ of the IVP from the previous example for $x \in [0,2]$.

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

```
>>> def taylor_3_cosy(x0, y0, xmax, n):
        h = (xmax - x0) / n
        ypoints = [y0]
        for i in range(n):
            y0 = y0 + cos(x0)*y0*h + 1/2*(cos(x0)**2-sin(x0))*y0*h**2 + \
                 1/6*(cos(x0)**2-3*sin(x0)-1)*cos(x0)*y0*h**3
            x0 = x0 + h
            ypoints.append(y0)
        return ypoints
```

Since the exact solution is $y(x) = e^{\sin(x)}$, we have $y(2) = e^{\sin(2)} \approx 2.48258$.

```
>>> taylor_3_cosy(0, 1, 2, 4)
```

```
[1, 1.625, 2.3475297541746047, 2.7350418255304874, 2.476391322837691]
```

For comparison, the exact values of the solution at these four steps are:

```
>>> [e**sin(i/2) for i in range(5)]
```

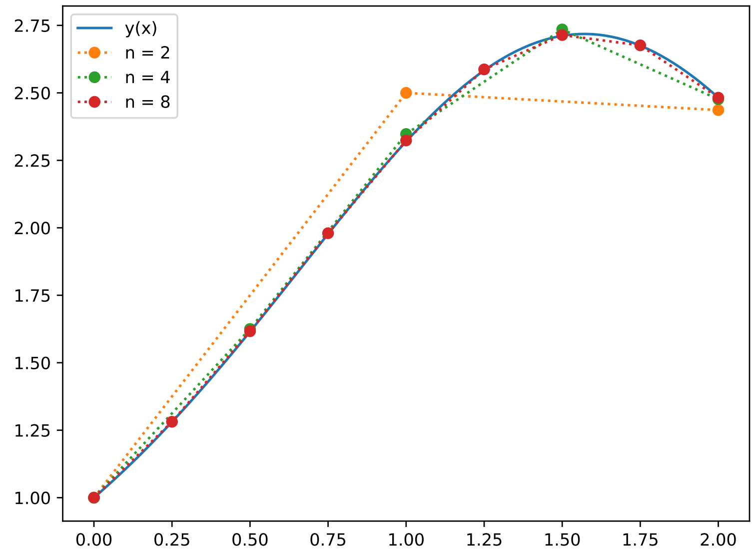
```
[1.0, 1.6151462964420837, 2.319776824715853, 2.7114810176821584, 2.4825777280150003]
```

The following convincingly illustrates that the error is indeed $O(h^3)$.

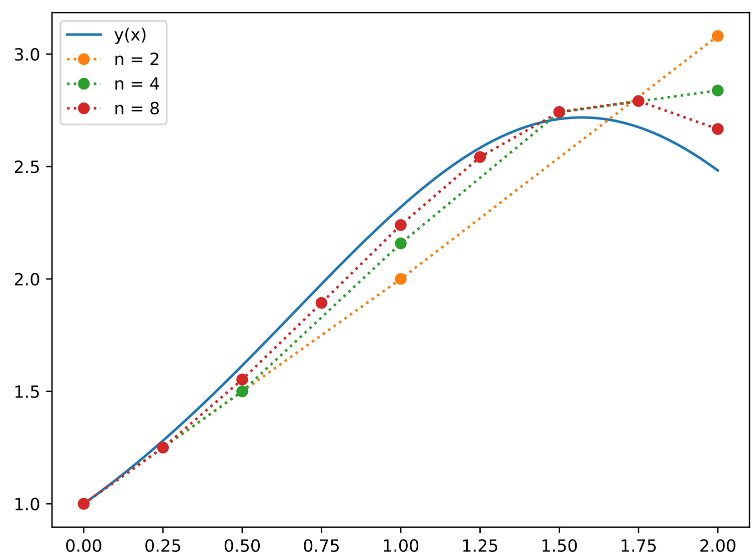
```
>>> [taylor_3_cosy(0, 1, 2, 10**n)[-1] - e**sin(2) for n in range(5)]
```

```
[2.5174222719849997, -0.0002461575553160955, -1.6375769584797695e-07, -
1.5647971807197791e-10, 2.0339285811132868e-13]
```

The following is a plot of the exact solution together with our approximations when using 2, 4 and 8 steps. Already for 4 steps, we obtain an approximation that, at least visually, is remarkably good.



On the other hand, for comparison, the following plot shows the corresponding approximations when using Euler's method instead.



Midpoint method (also known as the modified Euler method)

Recall that, in Euler's method (which is the same as the Taylor method of order 1) with step size h, we use $y_{n+1} = y_n + f(x_n, y_n)h$ because $f(x_n, y_n)$ is an approximation of $y'(x_n)$.

To get a better approximation $\tilde{y}_{n+1}$, we could do two small Euler steps (of size $h/2$ each) instead. We can then use the idea of Richardson extrapolation to combine Euler's y_{n+1} and the twice-Euler $\tilde{y}_{n+1}$ to get an approximation of higher order. This results in the following method:

(midpoint method) The following is an order 2 method for solving IVPs:

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

Advantage over Taylor method. When compared with the Taylor method of order 2, this has the advantage of not requiring us to determine partial derivatives of $f(x, y)$.

Why "midpoint"? In Euler's method, we use $y'(x_n) \approx f(x_n, y_n)$ as the slope for stepping from x_n to $x_{n+1} = x_n + h$. That is a bit unbalanced since we use the slope at the left endpoint for the entire interval (that's why Euler's method in the form we have seen is often called the **forward Euler method**). Note that the midpoint method is more balanced because it uses (an approximation of) the derivative at the midpoint $x_n + h/2$ instead.

How to derive the midpoint method. As mentioned above, we can derive the midpoint method by applying the idea of Richardson extrapolation to the Euler method.

- If we do one Euler step, then our approximation of $y(x_{n+1})$ is $y_{n+1}^{(1)} = y_n + f(x_n, y_n)h$.

  We know that the error in this approximation is roughly Ch^2 (in fact, $C \approx \frac{1}{2}y''(x_n)$ if h is small).

- If we do two Euler steps, then our approximation is $y_{n+1}^{(2)} = y_{\text{half}} + f(x_{\text{half}}, y_{\text{half}})\frac{h}{2}$ where $x_{\text{half}} = x_n + \frac{h}{2}$ and $y_{\text{half}} = y_n + f(x_n, y_n)\frac{h}{2}$ are the result of the first Euler step with step size $\frac{h}{2}$.

  Combined, we get $y_{n+1}^{(2)} = y_n + f(x_n, y_n)\frac{h}{2} + f\left(x_n + \frac{h}{2}, y_n + f(x_n, y_n)\frac{h}{2}\right)\frac{h}{2}$.

  The error should be roughly $2C\left(\frac{h}{2}\right)^2 = \frac{1}{2}Ch^2$ because we are doing 2 steps with step size $\frac{h}{2}$.

  [For small h, the constant C should still be as before, at least to first order.]

- As for Richardson extrapolation, $2y_{n+1}^{(2)} - y_{n+1}^{(1)}$ should therefore be an approximation of $y(x_{n+1})$ of order 3 (for the local truncation error).

  Indeed, $2y_{n+1}^{(2)} - y_{n+1}^{(1)} = y_n + f\left(x_n + \frac{h}{2}, y_n + f(x_n, y_n)\frac{h}{2}\right)h$ is precisely the midpoint method, which therefore should have global error of order 2.

An alternative way to show that the order is 2. We can also show that the midpoint method is of order 2 by computing and comparing Taylor expansions. The true solution has the expansion

$$\begin{aligned} y(x+h) &= y(x) + y'(x)h + \tfrac{1}{2}y''(x)h^2 + O(h^3) \\ &= y(x) + f(x, y(x))h + \frac{1}{2}\underbrace{\left[\frac{\mathrm{d}}{\mathrm{d}x} f(x, y(x))\right]}_{= f_x(x,y) + f_y(x,y)\cdot y'(x)} h^2 + O(h^3) \\ &= y + f(x, y)h + \tfrac{1}{2}(f_x(x, y) + f_y(x, y) f(x, y))h^2 + O(h^3). \end{aligned}$$

On the other hand, expanding the $f(\ldots)$ term on the right-hand side of the midpoint method (with x and y in place of x_n and y_n) using the multivariate chain rule, we find

$$y + f\left(x + \frac{h}{2}, y + f(x, y)\frac{h}{2}\right)h = y + \left(f(x, y) + \left(f_x(x, y)\frac{1}{2} + f_y(x, y)\frac{f(x, y)}{2}\right)h + O(h^2)\right)h,$$

which agrees with the true expansion up to $O(h^3)$.

Example 145. Consider the IVP $y' = y$, $y(0) = 1$. Approximate the solution $y(x)$ for $x \in [0, 1]$ using the midpoint method with 4 steps. In particular, what is the approximation for $y(1)$?

Comment. Compare with Example 139. The real solution is $y(x) = e^x$ so that $y(1) = e \approx 2.71828$.

Solution. The step size is $h = \frac{1-0}{4} = \frac{1}{4}$. We apply the midpoint method with $f(x, y) = y$:

$$\begin{array}{ll}
x_0 = 0 & y_0 = 1 \\
x_1 = \frac{1}{4} & y_1 = y_0 + f\left(x_0 + \frac{h}{2}, y_0 + f(x_0, y_0)\frac{h}{2}\right)h = \frac{41}{32} \approx 1.2813 \\
x_2 = \frac{1}{2} & y_2 = y_1 + f\left(x_1 + \frac{h}{2}, y_1 + f(x_1, y_1)\frac{h}{2}\right)h = \frac{41^2}{32^2} \approx 1.6416 \\
x_3 = \frac{3}{4} & y_3 = y_2 + f\left(x_2 + \frac{h}{2}, y_2 + f(x_2, y_2)\frac{h}{2}\right)h = \frac{41^3}{32^3} \approx 2.1033 \\
x_4 = 1 & y_4 = y_3 + f\left(x_3 + \frac{h}{2}, y_3 + f(x_3, y_3)\frac{h}{2}\right)h = \frac{41^4}{32^4} \approx 2.6949
\end{array}$$

In particular, the approximation for $y(1)$ is $y_4 \approx 2.6949$, which is a noticeable improvement over Example 139, where we used the Euler method instead.

Comment. The above computations become particularly transparent if we realize that, for $f(x, y) = y$, each Euler step takes the simple form $y_{n+1} = y_n + f\left(x_n + \frac{h}{2}, y_n + f(x_n, y_n)\frac{h}{2}\right)h = y_n\left(1 + h + \frac{h^2}{2}\right)$.
It follows that $y_n = \left(1 + h + \frac{h^2}{2}\right)^n$. Can you see how, as $h \to 0$, this allows us to recover the exact solution?

Example 146. $\boxed{\texttt{Python}}$ 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_y(x, y):
        return y
```

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

```
>>> midpoint(f_y, 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_y, 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]
```

Armin Straub
straub@southalabama.edu