Example 140. (cont'd) Consider the IVP $y' = y$, $y(0) = 1$. Approximate the solution $y(x)$ for $x \in [0, 1]$ using Euler's method with n steps for several values of n. In each case, what is the approximation for $y(1)$?

Solution. Since the real solution is $y(x) = e^x$ so that, in particular, the exact solution is $y(1) = e \approx 2.71828$. We proceed as we did in Example 139 in the case $n = 4$ and apply Euler's method with $f(x, y) = y$:

$$\begin{aligned} x_{n+1} &= x_n + h \\ y_{n+1} &= y_n + h \underbrace{f(x_n, y_n)}_{= y_n} = (1 + h) y_n \end{aligned}$$

We observe that it follows from $y_{n+1} = (1 + h) y_n$ that $y_n = (1 + h)^n y_0$. Since $y_0 = 1$ and $h = \frac{1 - 0}{n} = \frac{1}{n}$, we conclude that

$$x_n = 1, \quad y_n = \left(1 + \frac{1}{n}\right)^n.$$

[For instance, for $n = 4$, we get $x_4 = 1$, $y_4 = \left(\frac{5}{4}\right)^4 \approx 2.4414$ as in Example 139.]

In particular, our approximation for $y(1)$ is $\left(1 + \frac{1}{n}\right)^n$.

Here are a few values spelled out:

$$\begin{aligned} n = 1&: \quad \left(1 + \frac{1}{n}\right)^n = 2 \\ n = 4&: \quad \left(1 + \frac{1}{n}\right)^n = 2.4414\ldots \\ n = 12&: \quad \left(1 + \frac{1}{n}\right)^n = 2.6130\ldots \\ n = 100&: \quad \left(1 + \frac{1}{n}\right)^n = 2.7048\ldots \\ n = 365&: \quad \left(1 + \frac{1}{n}\right)^n = 2.7145\ldots \\ n = 1000&: \quad \left(1 + \frac{1}{n}\right)^n = 2.7169\ldots \\ n \to \infty&: \quad \left(1 + \frac{1}{n}\right)^n \to e = 2.71828\ldots \end{aligned}$$

We can see that Euler's method converges to the correct value as $n \to \infty$. On the other hand, we can see that it doesn't converge impressively fast. That is why, for serious applications, one usually doesn't use Euler's method directly but rather higher-order methods derived from the same principles (such as Runge–Kutta methods).

Interpretation. Note that we can interpret the above values in terms of compound interest. We start with initial capital of $y(0) = 1$ and we are interested in the capital $y(1)$ after 1 year if we receive interest at an annual rate of 100%:

- If we receive a single interest payment at the end of the year, then $y(1) = 2$ (case $n = 1$ above).
- If we receive quarterly interest payments of $\frac{100\%}{4} = 25\%$ each, then $y(1) = (1.25)^4 = 2.441\ldots$ (case $n = 4$).
- If we receive monthly interest payments of $\frac{100\%}{12} = \frac{1}{12}$ each, then $y(1) = 2.6130\ldots$ (case $n = 12$).
- If we receive daily interest payments of $\frac{100\%}{365} = \frac{1}{365}$ each, then $y(1) = 2.7145\ldots$ (case $n = 365$).

It is natural to wonder what happens if interest payments are made more and more frequently. Well, we already know the answer! If interest is compounded continuously, then we have e in our bank account after one year.

Armin Straub
straub@southalabama.edu

Example 141. Python Let us implement Euler's method to redo and extend Example 139.

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

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

If we choose the number of steps n to be 4 and xmax to be 1 (because we want $x_n = 1$), then the following matches exactly our computation in Example 139:

```
>>> euler(f_y, 0, 1, 1, 4)
```

```
[1, 1.25, 1.5625, 1.953125, 2.44140625]
```

As expected, increasing the number of steps provides better approximations to the exact solution $y(x) = e^x$ with $y(1) = e \approx 2.718$.

```
>>> euler(f_y, 0, 1, 1, 10)
```

```
[1, 1.1, 1.2100000000000002, 1.3310000000000002, 1.4641000000000002, 1.61051,
1.7715610000000002, 1.9487171, 2.1435888100000002, 2.357947691, 2.5937424601]
```

```
>>> euler(f_y, 0, 1, 1, 100)[-1]
```

```
2.704813829421526
```

If ypoints is a list, then its elements can be accessed as ypoints[0], ypoints[1], ... Moreover, we can access the last element as ypoints[-1]. For instance, above, we used euler_e(f, 0, 1, 1, 100)[-1] to get the last element of the 101 approximations $y_0, y_1, \ldots, y_{100}$. That last element is the approximation of $y(1) = e$.

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

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

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

```
[-0.7182818284590451, -0.124539368359045, -0.013467999037519274, -
0.0013578962231490799, -0.00013590163381849152, -1.3591284549807625e-05]
```

However, note that our computer had to work pretty hard to get the final approximation, because that entailed computing 10^5 values. We clearly need a higher order method in order to compute to higher accuracy.

Armin Straub
straub@southalabama.edu

(Taylor method of order k) The following is an order k method for solving IVPs:

$$\begin{aligned} x_{n+1} &= x_n + h \\ y_{n+1} &= y_n + f(x_n, y_n)h + \tfrac{1}{2} f'(x_n, y_n)h^2 + \cdots + \tfrac{1}{k!} f^{(k-1)}(x_n, y_n)h^k \end{aligned}$$

where $f^{(n)}(x, y)$ is short for $\frac{\mathrm{d}^n}{\mathrm{d}x^n} f(x, y(x))$ (expressed in terms of f and its partial derivatives).

For instance. $f'(x, y) = \frac{\mathrm{d}}{\mathrm{d}x} f(x, y(x)) = f_x(x, y) + f_y(x, y) y'(x) = f_x(x, y) + f_y(x, y) f(x, y)$

Especially for higher derivatives, it is easier to compute these for specific f. See next example.

Comment. As for Euler's method, being an order k method means that the method has a global error that is $O(h^k)$ (while the local truncation error is $O(h^{k+1})$; note that we can see this because we truncate the Taylor expansion of $y(x)$ after h^k so that the next term is $O(h^{k+1})$).

Example 142. Spell out the Taylor method of order 2 for numerically solving the IVP

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

Solution. The Taylor method of order 2 is based on the Taylor expansion

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

where we have a local truncation error of $O(h^3)$ so that the global error will be $O(h^2)$.

From the DE we know that $y'(x) = \cos(x) y$, which is $f(x, y)$. 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 $f'(x, y)$. Hence, the Taylor method of order 2 takes the form:

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

For any choice of h, we can therefore compute $(x_1, y_1), (x_2, y_2), \ldots$ starting with (x_0, y_0) by the above recursive formula combined with $x_{n+1} = x_n + h$.

Armin Straub
straub@southalabama.edu