LECTURE: EULER'S METHOD (II)

1. Error

Euler's method generally gives you a *good* approximation to your solution y. What does "good" mean in this context? For this, we have to talk about the **error**

Notice: There are two quantities at play here:

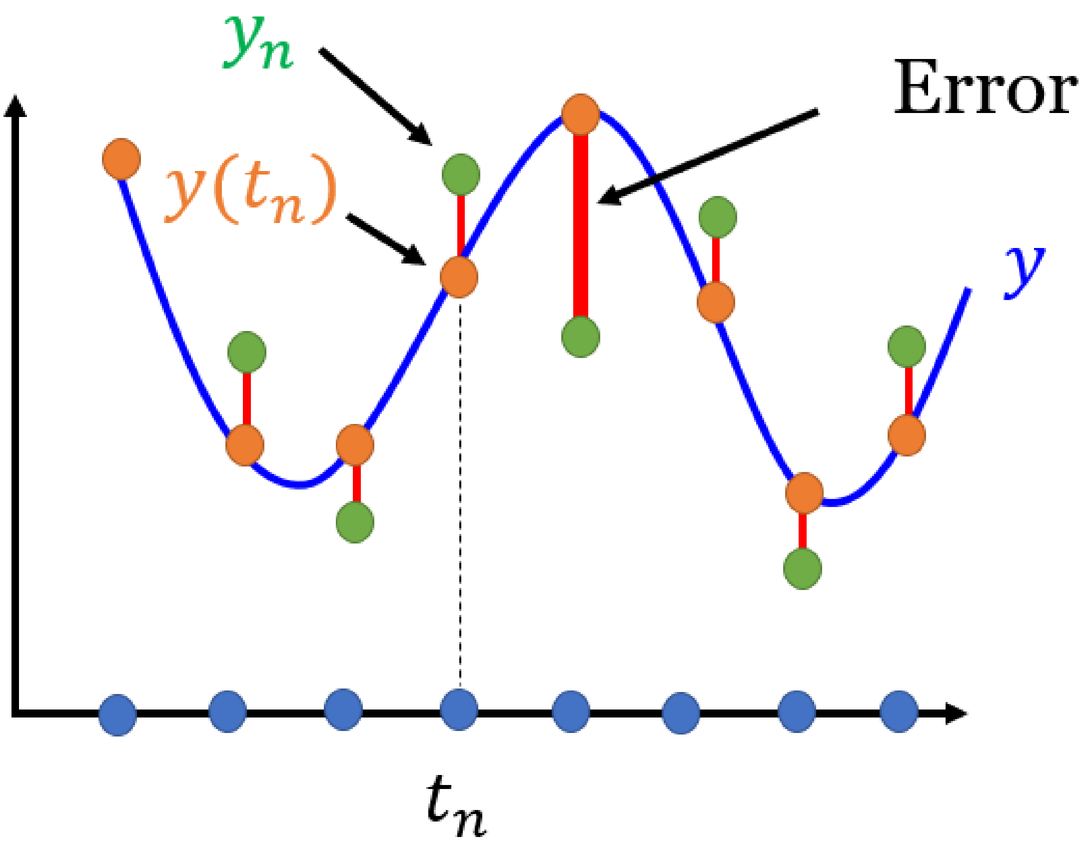
(1) Our approximations

$$y_0 \qquad y_1 \qquad y_2 \qquad y_N$$

(2) The values of the actual solution y

$$y(t_0) \qquad y(t_1) \qquad y(t_2) \qquad y(t_N)$$

It makes sense to compare the two:



Definition: Error

$$E(h) = \max\left\{ |y_1 - y(t_1)|\,,\ |y_2 - y(t_2)|\,,\ \ldots\,,\ |y_N - y(t_N)| \right\}$$

Think of the error as the "worst possible scenario" If just one of the y_k is far from $y(t_k)$ then $|y_k - y(t_k)|$ is big, so the error is big Conversely, if the error is small, then *all* the y_k are close to $y(t_k)$

In the picture above, the error is the length of the thick red line. Also, we don't include y_0 because $y(t_0) = y_0$ anyway

Ideally: we want the error to be small if h is small, that is:

$$\lim_{h\to 0} E(h) = 0$$

And in fact this is true with Euler's method:

Fact:

For Euler's method, there is a constant $C > 0$ such that for all h

$$|E(h)| \leq Ch$$

In particular this implies $\lim_{h\to 0} E(h) = 0$, which is what we want

Why? This really just follows from Taylor expansion:

Recall: Taylor Expansion

$$f(x+h) = f(x) + hf'(x) + O(h^2)$$

Note: $O(h^2)$ here just means h^2 terms, such as $\frac{h^2}{2}f''(x)$

$$\begin{aligned} y(t_1) =& y(t_0 + h) = y(t_0) + hy'(t_0) + O(h^2) \\ \overset{\text{ODE}}{=}& y(t_0) + hf(y(t_0), t_0) + O(h^2) \\ =& \underbrace{y_0 + hf(y_0, t_0)}_{y_1} + O(h^2) \\ =& y_1 + O(h^2) \end{aligned}$$

Hence $y(t_1) - y_1 = O(h^2) \Rightarrow |y(t_1) - y_1| \leq Ch^2$

Note: Here we get Ch^2, but that's also because we had $y(t_0) = y_0$. In general we need to repeat this N times for all the terms $y(t_k)$ which, instead of Ch^2, gives $NCh^2 = \left(\frac{b-a}{h}\right)Ch^2 = \underbrace{C(b-a)}_{C}h = Ch$

2. Problems with Euler

Welcome to the dark side of Euler! What could *possibly* go wrong?

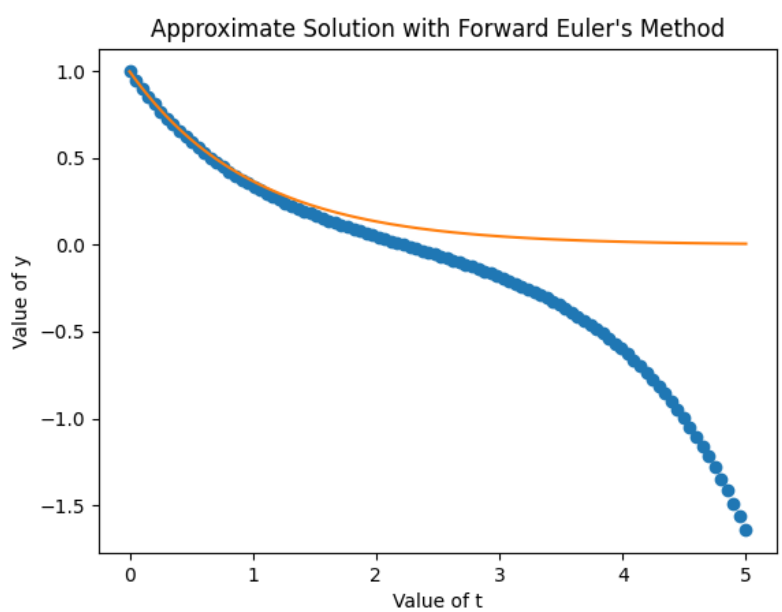
Issue 1: Sensitivity to Initial Conditions

Example 1:

$$\begin{cases} y' = y - 2e^{-t} \\ y(0) = 1 \end{cases}$$

The solution is $y = e^{-t}$ (using integrating factors)

If you use Euler's method, then you'll see that the true solutions and your approximations start deviating after a while!



What went wrong? The reason is that the general solution of the ODE (without intial conditions) is

$$y = e^{-t} + Ce^{t}$$

If you use *exactly* $y(0) = 1$ then $C = 0$ and you get $y = e^{-t}$. But the problem is that computers don't use exact values, but approximate values, like $y(0) = 0.999$

Even a tiny rounding error like that will cause $C \neq 0$, and in the end we end up getting an extra e^t term. So the approximate solution might look like $e^{-t} + 0.02e^{t}$ which blows up for large t.

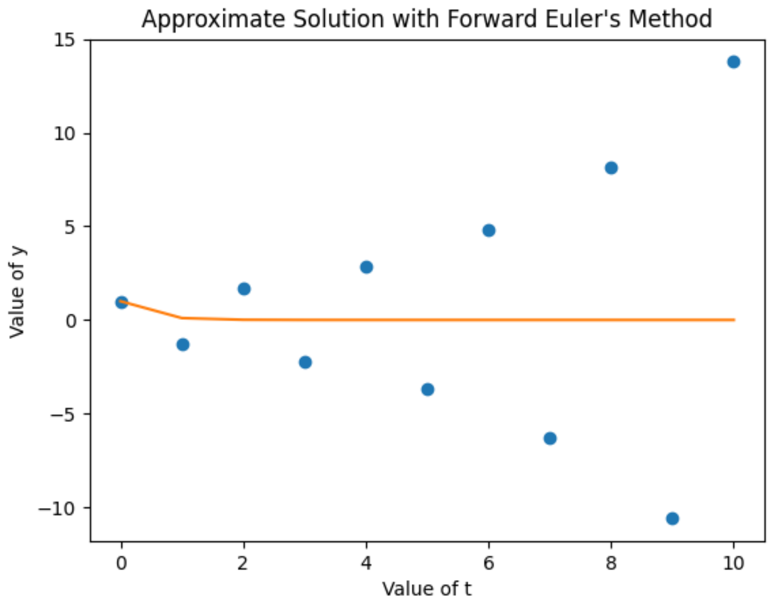
Issue 2: Instability

Sometimes the solutions can oscillate, like the following:

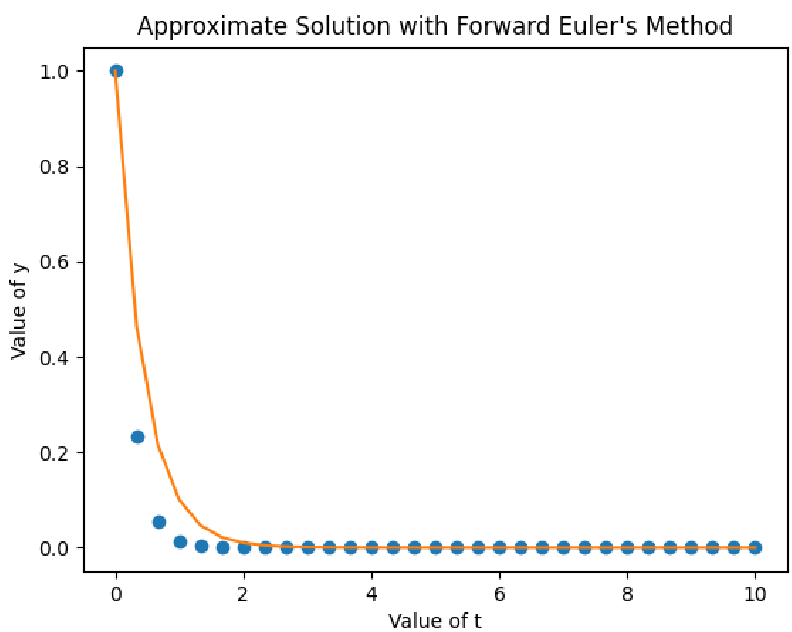
Example 2:

$$\begin{cases} y' = & -2.3y \\ y(0) = & 1 \end{cases}$$

The exact solution is $y = e^{-2.3t}$, but if you apply Euler with $h = 1$, then the solution oscillates and is *not* close to the exact solution.



If you apply Euler with a relatively smaller value of h like $h = 0.3$ here, or any h with $(2.3)h < 1$ then the solution decays to 0



Here the behavior is heavily dependent on the value of h you use; sometimes you need to make h *really* small to make this work, which is called **instability**.

3. Variations of Euler

To get around this, applied mathematicians sometimes use variations of Euler's method, such as

Backward Euler:

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1})$$

This is an *implicit* method, because we first have to solve for y_{n+1} first before we can apply it

Multistep Method:

$$y_{n+1} = y_n + \frac{3}{2}hf(t_n, y_n) - \frac{1}{2}hf(t_{n-1}, y_{n-1})$$

Multistep because it uses both present y_n and past y_{n-1} values here

Runge-Kutta Methods:

$$y_{n+1} = y_n + hf\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}f(t_n, y_n)\right)$$

The idea is to evaluate f here at several points.

Although more complicated, the methods achiever a higher accuracy and resolve some of the problems discussed.

4. DSOLVE APP

Here is a cool program in Python that solves ODE symbolically.

Example 3:

$$t^2 y' + 2ty = \cos(t)$$

Notice this is the same as $t^2 y' + 2ty - \cos(t) = 0$

```
from sympy import *
t=symbols('t')
y=Function('y')

deq=t**2*diff(y(t),t)+2*t*y(t)-cos(t)
ysoln=dsolve(deq,y(t))
print(ysoln)
```

Remarks:

- deq is the ODE, make sure to write it in the form $= 0$
- $\star\star$ means "square"
- dsolve is the program that solves our differential equation. It has two inputs: The first one is the differential equation, the second one is our unknown
- Print prints out the solution
- To write things like e^{2t}, use $\exp(2 \star \text{t})$

What this says here is that the solution is $y(t) = \frac{C+\sin(t)}{t^2}$

Example 4:

$$\begin{cases} ty' + (t+1)y = 2te^{-t} \\ y(1) = 2 \end{cases}$$

To specify initial conditions, you need to add ics $= \{y(1) : 2\}$ as a third input in your dsolve command

```
from sympy import *
t=symbols('t')
y=Function('y')

deq=t*diff(y(t),t)+(t+1)*y(t)-2*t*exp(-t)
ysoln=dsolve(deq,y(t),ics={y(1):2})
print(ysoln)
```

So here the solution is $y = \left(t^2 - 1 + 2e\right) \frac{e^{-t}}{t}$

Example 5:

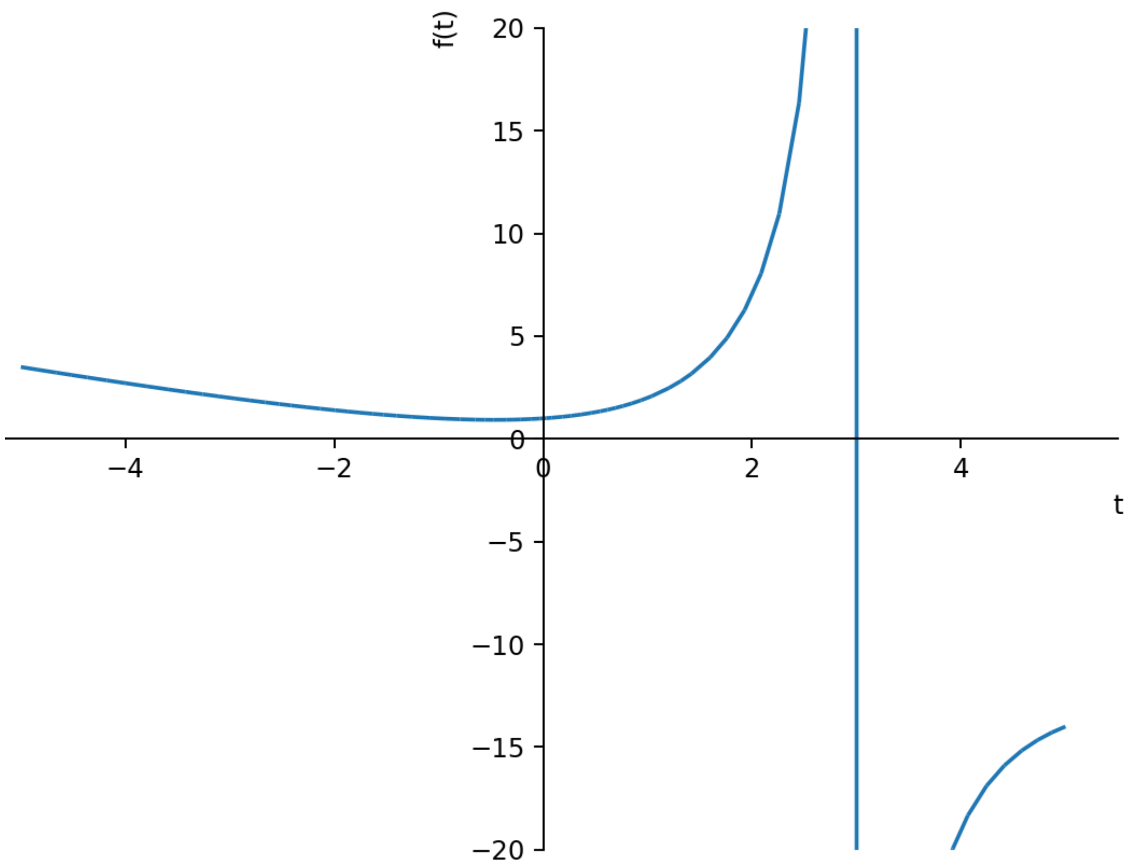
Plot the solution of

$$\begin{cases} \left(-y^2 - 2ty\right) + \left(3 + t^2\right) y' = 0 \\ y(0) = 1 \end{cases}$$

Warning: Do not forget about the .rhs in your plot command!

```
from sympy import *
from matplotlib import pyplot as plt
```

```
t=symbols('t')
y=Function('y')
deq=(-y(t)**2-2*t*y(t)) + (3+t**2)*diff(y(t),t)
ysoln=dsolve(deq,y(t),ics={y(0):1})
print(ysoln)
plot(ysoln.rhs,(t,-5,5),ylim=[-20,20])
```



5. Second-Order ODE

Welcome to the magical world of second-order ODE! Those are equations involving y'' instead of just y'

Existence-Uniqueness: The theorem is the same as for first-order ODE, the only difference is that now we need to specify an initial position $y(0)$ **and** an initial velocity $y'(0)$

Example 6:

$$\begin{cases} y'' = 2y' + t\left(y^2\right) \\ y(0) = 2 \\ y'(0) = 3 \quad \text{NEW} \end{cases}$$

Theorem:

Consider the ODE

$$\begin{cases} y'' = f(y, y', t) \\ y(0) = y_0 \\ y'(0) = v_0 \end{cases}$$

Where y_0 and v_0 are given

If f and its partial derivatives are continuous, then there is a unique solution $y = y(t)$ for t near 0

Fun Application: A nice visualization of this is the game Angry Birds, where you determine the initial position and the initial velocity of a bird and try to find a trajectory that goes through a pig. Non-existence would mean the bird blows up, and non-uniqueness would mean that one bird splits into two birds (two trajectories).

Application: Second-order ODE are used to study harmonic oscillators in physics; there will be a whole lecture dedicated to them.