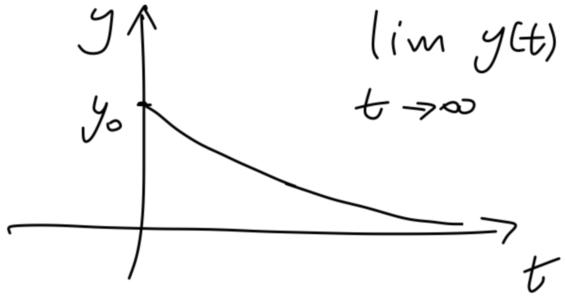
Numerical stability

• Introduction

* Model problem: $y'(t) = \underbrace{\lambda y(t)}^{f(y)}$, $y(0) = y_0$, $\lambda < 0$



$\lim_{t\to\infty} y(t) = 0$

→ Analytical solution : $y(t) = y_0 e^{\lambda t}$

* Euler's method : $y_{n+1} = y_n + hf(t_n, y_n) = y_n + h(\lambda y_n) = \boxed{(1+\lambda h)y_n}$

$\Rightarrow \boxed{y_n = (1+\lambda h)y_{n-1}} = (1+\lambda h)[(1+\lambda h)y_{n-2}] = (1+\lambda h)^2 y_{n-2} = \cdots = (1+\lambda h)^n y_0$

→ $\boxed{y_n = (1+\lambda h)^n y_0} \Rightarrow \boxed{|y_n| = |1+\lambda h|^n |y_0|}$

→ if $|1+\lambda h| > 1$, then $|y_n| > |y_{n-1}| > |y_{n-2}| > \dots > |y_0|$, that is, the numerical solution will keep increasing (in magnitude) with time, which is the opposite of what should happen! That will also lead to $\lim_{n\to\infty} |y_n| = \infty$ (the solution will "blow up")

- Since $|y_n| = |1+\lambda h|^n |y_0|$, we can only get the desired decay ($\lim_{n\to\infty} |y_n| = 0$) if we have $|1+\lambda h| < 1$. Since $\lambda < 0$, we can write $\lambda = -|\lambda|$:

$$|1-|\lambda|h| < 1 \iff -1 < |\lambda|h - 1 < 1 \iff \boxed{0 < h < \frac{2}{|\lambda|}}$$

$\to$ time-step size restriction

- When $|\lambda|$ is large, we say the ODE is "stiff". In that case, we get a severe limitation on the range of time-step sizes h we can use.

Ex.: $y'(t) = -200\, y(t) \to \lambda = -200 \Rightarrow 0 < h < \frac{2}{|-200|} \Rightarrow \boxed{0 < h < 0.01}$

↳ anything outside this range will yield an unstable solution!

• Implicit methods

- To remedy this instability issue, we can (sometimes must) use implicit methods. Let's start from the "backward Euler method":

$$y'(t) = f(t, y(t)) \Rightarrow y'(t_{n+1}) = f(t_{n+1}, y(t_{n+1}))$$

$\to$ Backward finite difference: $\frac{y_{n+1} - y_n}{h} = f(t_{n+1}, y_{n+1}) \Rightarrow \boxed{y_{n+1} = y_n + h f(t_n + h, y_{n+1})}$

↳ Implicit Euler

- Let's see how the numerical solution of the model problem behaves when using implicit Euler (a.k.a backward Euler):

$$\to y_{n+1} = y_n + h\underbrace{f(y_{n+1})}_{\lambda y_{n+1}} = y_n + h(\lambda y_{n+1}) \Rightarrow (1-\lambda h)y_{n+1} = y_n \Rightarrow \boxed{y_{n+1} = \left(\frac{1}{1-\lambda h}\right) y_n}$$

$\to$ Similarly as before, we can write $y_n = \left(\frac{1}{1-\lambda h}\right)^n y_0$, but since $h>0$ and $\lambda<0$, we know that $1-\lambda h = 1+|\lambda| h > 1$, so that $\boxed{\frac{1}{1-\lambda h} < 1}$ for any $h>0$.

- This means we will always have $|y_{n+1}| < |y_n|$ and $\lim_{n\to\infty} |y_n| = 0$, regardless of the time-step size! We thus say that the backward Euler method is unconditionally stable, at least for the linear model problem under consideration.

* Here, a large h will mean that $|y_{n+1}| << |y_n|$, since

$$y_{n+1} = \left(\frac{1}{1+|\lambda|h}\right) y_n$$

$\hookrightarrow << 1$ for large h

This phenomenon is know as numerical damping.

• Stability region

- This analysis motivates the definition of a stability region for h. Let's consider a slightly more general version of the linear model problem:

$$y'(t) = \lambda y(t), \quad y(0) = y_0, \quad \underline{\lambda \in \mathbb{C}}$$

- Since the ODE is linear, we will always be able to write the numerical solution as

$$y_{n+1} = [R(\lambda h)]\, y_n, \quad \text{so that} \quad y_n = [R(\lambda h)]^n y_0$$

↳ R is called stability function (it depends on the numerical method)

- If $|R(\lambda h)| > 1$, then $|y_n| \to \infty$ as $n \to \infty$. Therefore, we want h such that $|R(\lambda h)| \leq 1$. In other words, we want $\underline{\lambda h}$ to be in the stability region S, defined as:

$$S := \{ z \in \mathbb{C} : |R(z)| < 1 \}$$

<u>Requirement</u>: $\lambda h \in S$

*Examples:

→ Euler (explicit): $y_{n+1} = \overbrace{(1+\lambda h)}^{R(\lambda h)} y_n \Rightarrow R(z) = 1 + z$

→ Implicit Euler: $y_{n+1} = \underbrace{\left(\dfrac{1}{1-\lambda h}\right)}_{R(\lambda h)} y_n \Rightarrow R(z) = \dfrac{1}{1-z}$

• Definition: a method is **A**-stable when S contains the left part of the complex plane ($\mathbb{C}^- = \{z \in \mathbb{C} : Re(z) \leq 0\}$)

▶ Example: find the stability region for Heun's method and, for the real case ($\lambda \in \mathbb{R}$), determine the stability interval for h.

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

→ Equation: $y'(t) = f(y) = \lambda y$, $\lambda \in \mathbb{C}$

$$\Rightarrow \begin{cases} f(y_n) = \lambda y_n \\ f(y_n + hf(y_n)) = f(y_n + h\lambda y_n) = f((1+\lambda h) y_n) = \lambda\left[(1+\lambda h) y_n\right] = \lambda y_n + \lambda^2 h y_n \end{cases}$$

$$\Rightarrow y_{n+1} = y_n + \frac{h}{2}\Big[\underbrace{\lambda y_n}_{f(y_n)} + \underbrace{\lambda y_n + \lambda^2 h y_n}_{f(y_n + hf(y_n))}\Big] \Rightarrow y_{n+1} = \underbrace{\left[1+\lambda h + \frac{(\lambda h)^2}{2}\right]}_{R(\lambda h)} y_n$$

$$\hookrightarrow R(z) = 1 + z + z^2/2$$

$\rightarrow S = \{z \in \mathbb{C} : |1+z+z^2/2| \le 1\}$ → we will have stability if, and only if: $\lambda h \in S$

* Now, let's consider the case where $\lambda \in \mathbb{R}$, $\lambda < 0$. In that case, we have

$$R(\lambda h) = 1+\lambda h + \frac{(\lambda h)^2}{2} = \underbrace{\left(1+\frac{\lambda h}{2}\right)^2 + \left(\frac{\lambda h}{2}\right)^2}_{\text{the sum of two squares can never be negative (for real numbers)}} \ge 0 \quad \text{for all } h \quad (*)$$

→ For stability, we need $|R(\lambda h)| \le 1$, that is,

$\boxed{R(\lambda h) \le 1}$ and $\boxed{R(\lambda h) \ge -1}$ → already satisfied, due to (*)

* Remember: $\lambda = -|\lambda|$ (since $\lambda < 0$). Therefore, the condition $R(\lambda h) \le 1$ becomes:

$$\cancel{1} - |\lambda| h + \frac{(|\lambda| h)^2}{2} \le \cancel{1} \Leftrightarrow \frac{|\lambda| h}{|\lambda| h}\left(\frac{|\lambda| h}{2} - 1\right) \le \frac{0}{|\lambda| h} \Leftrightarrow h \le \frac{2}{|\lambda|} \quad \left(0 < h < \frac{2}{|\lambda|}\right)$$

- **Exercise:** find S for the (implicit) trapezoidal method:

  $$y_{n+1} = y_n + \frac{h}{2}\left[f(t_n, y_n) + f(t_{n+1}, y_{n+1})\right],$$ and show that it is unconditionally stable for the equation $y'(t) = \lambda y(t)$, with $\lambda < 0$.

- **Numerical example:** $y' = -4y$, $y(0) = 1 \Rightarrow y(t) = e^{-4t}$

→ Stability intervals:

* Euler: $0 < h < \frac{2}{|-4|} \Rightarrow 0 < h < 0.5$ (Python)
* Implicit Euler: $h > 0$ (always stable)
* Heun: $0 < h < \frac{2}{|-4|} = 0.5$

- **Stability in linear ODE systems**

$$\underline{Y}'(t) = \underline{\underline{A}}\, Y(t), \quad \underline{\underline{A}} \in \mathbb{C}^{N\times N} \text{ (}N\times N\text{ matrix)}$$

→ In that case, we will have a stable solution if, and only if all the eigenvalues of $\underline{\underline{A}}$, multiplied by h, are in the stability region S of our numerical method:

$$\lambda_j h \in S \text{ for all } j = 1, 2, \ldots, N$$

↳ each λ_j is an eigenvalue of $\underline{\underline{A}}$

• Implicit stepping: practical aspects

- If implicit methods tend to be more stable, why not always use them?

Unfortunately, implicit methods can be much more costly when we are solving, for instance, ODE systems or nonlinear ODEs.

* Example (nonlinear ODE): $y'(t) = \overbrace{2^{t-y}}^{f(t,y)}$, $y(0)=1$, $h=1$

- • Explicit Euler: $y_1 = y_0 + h f(t_0, y_0) = 1 + 1 \cdot 2^{0-1} = 1.5$ ✓
- • Implicit Euler: $y_1 = y_0 + h f(t_1, y_1) \Rightarrow \boxed{y_1 = 1 + 2^{1-y_1}}$?

↳ nonlinear equation: $y_1 - \frac{2}{2^{y_1}} - 1 = 0$ → implement Newton's method!

* Example (linear ODE system): $\underline{Y}'(t) = \underline{\underline{A}}\, Y(t)$, $\underline{Y}(0) = \underline{Y}_0$

- • Explicit Euler: $\underline{Y}_1 = \underline{Y}_0 + h \underline{\underline{A}}\, Y_0 = (\underline{\underline{I}} + h\underline{\underline{A}})\underline{Y}_0$ ($\underline{\underline{I}}$: $N \times N$ identity matrix)
- • Implicit Euler: $\underline{Y}_1 = \underline{Y}_0 + h \underline{\underline{A}}\, \underline{Y}_1 \Rightarrow \underbrace{(\underline{\underline{I}} - h\underline{\underline{A}})}_{\text{coefficient matrix}} \underline{Y}_1 = \underline{Y}_0$ ($N \times N$ linear system to solve!)