Previously, we saw that using a time step (Δt) that exceeds a certain threshold, risks rendering the entire simulation unstable when using the Forward Euler Method.

This is a "defect" of Forward Euler ... not the mass-spring formulation per se. In fact similar instability can be seen in an effort to simulate just a single spring.

Going even further, we turn our attention to the simplest physical system that exhibits similar behavior: The "model" first-order O.D.E.:

$$y'(t) = \lambda \cdot y(t)$$

Before we attempt a physical interpretation, we can see that it is possible to compute the unknown function $y(t)$ on-paper:

$$\frac{y'(t)}{y(t)} = \lambda \leftrightsquigarrow [\ln y(t)]' = \lambda \leftrightsquigarrow \ln y(t) = \lambda t + c_0$$

$$\leftrightsquigarrow y(t) = e^{\lambda t} \underbrace{e^{c_0}}_{:= c} = c e^{\lambda t}$$ We can further specify "c" by plugging in $t=0$:

$$y(0) = c e^0 = c \Rightarrow \boxed{y(t) = y(0) e^{\lambda t}}$$

As we see, this differential equation describes a physical quantity that changes exponentially over time: either increasing exponentially (when $\lambda > 0$), or decaying exponentially (when $\lambda < 0$). We will only focus on the case $\lambda < 0$, which corresponds to the inherent tendency of many mechanical systems to come to rest.

The Forward Euler method suggests a process for approximating values $y(dt), y(2dt), \ldots, y(k \cdot dt)$ of the unknown function algorithmically, rather than solving it on paper.

Let us introduce the notation:

$$\left\{ \begin{array}{l} t_i = t_0 + i \cdot dt \\ y_i = y(t_i) \end{array} \right\}$$

Forward Euler (along with some additional variants which we will see) is a so-called time integration method. The name is due to the following approach for solving the ODE $y'(t) = \lambda y(t)$, or any 1st order ODE, for that matter.

In fact we will look at the slightly more general differential equation $y'(t) = f(t, y)$

[Note that, in our case $f(t,y) = \lambda y$, i.e. f just happens to not explicitly contain t.]

other examples: $y'(t) = -t^2 y(t)$ $\quad (f(t,y) = -t^2 y)$

$y'(t) = -\sin(t)\, y^3(t)$ $\quad (f(t,y) = -\sin(t) y^3)$

Then, we have

$$y'(t) = f(t, y(t)) \rightsquigarrow \int_{t_k}^{t_{k+1}} y'(t)\,dt = \int_{t_k}^{t_{k+1}} f(t, y(t))\,dt$$

$$\rightsquigarrow y(t_{k+1}) - y(t_k) = \int_{t_k}^{t_{k+1}} f(t, y(t))\,dt$$

$$\rightsquigarrow \boxed{y(t_{k+1}) = y(t_k) + \int_{t_k}^{t_{k+1}} f(t, y(t))\,dt}$$

Forward Euler, or any alternative methods amount to approximating the integral $\int_{t_k}^{t_{k+1}} f(t,y)$ with something more "practical" (hence the name "integration schemes").

Next we examine some properties of these schemes:

I. IMPLICIT / EXPLICIT NATURE

F. Euler $\quad y_{k+1} = y_k + dt\, f(t_k, y_k)$

B. Euler $\quad y_{k+1} = y_k + dt\, f(t_{k+1}, y_{k+1})$

Trapezoidal $\quad y_{k+1} = y_k + (dt/2)\{f(t_k, y_k) + f(t_{k+1}, y_{k+1})\}$

Forward Euler is an explicit method: The computation of y_{k+1} is a matter of simple substitution, as y_{k+1} is isolated on the left-hand side.

Backward Euler & Trap. Rule contain y_{k+1} both on left- and right-hand sides of the equation. It is still possible to compute y_{k+1}, but we have to treat the respective rule as an equation (possibly nonlinear) that needs to be solved for y_{k+1}.

e.g. $f(t,y) = -ty^2$ $\quad$ (i.e. $y'(t) = -ty^2(t)$)

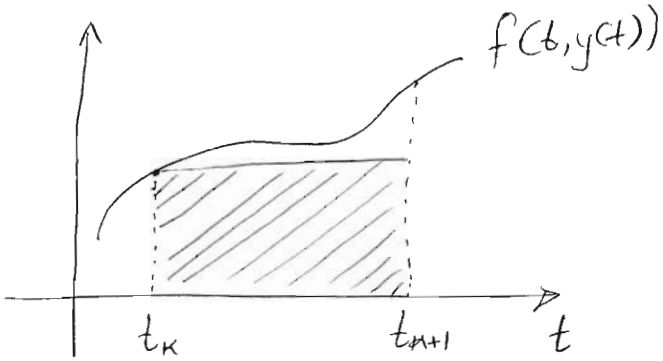
Backward Euler yields:

$$y_{k+1} = y_k + dt\,(-t_{k+1} y_{k+1}^2) \Rightarrow \text{Solve quadratic eqn for } y_{k+1}!$$

If $f(t,y)$ is a linear function in y, we have to solve a relatively simple equation (or equations, see later) for y_{k+1}.

The various options are:

I.

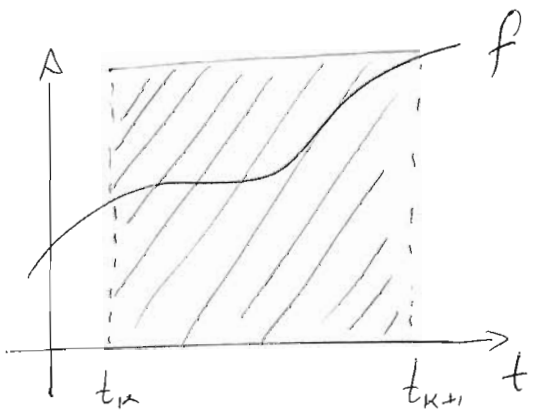


$$\int_{t_K}^{t_{K+1}} f(t,y(t))\,dt \approx (t_{K+1}-t_K)\, f(t_K, y(t_K)) = dt\, f(t_K, y_K)$$

Substituting into the previous integral equation, yields:

$$\boxed{y_{K+1} = y_K + dt\, f(t_K, y_K)}$$ Forward Euler

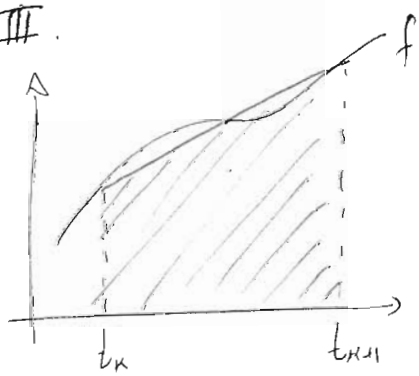
II.



$$\int_{t_K}^{t_{K+1}} f(t,y) \approx dt\, f(t_{K+1}, y_{K+1})$$

Substituting: $\boxed{y_{K+1} = y_K + dt\, f(t_{K+1}, y_{K+1})}$ Backward Euler

III.



$$\int_{t_K}^{t_{K+1}} f(t,y) \approx dt\, \frac{f(t_K, y_K) + f(t_{K+1}, y_{K+1})}{2}$$

$$\boxed{y_{K+1} = y_K + \frac{dt}{2}\left[f(t_K, y_K) + f(t_{K+1}, y_{K+1})\right]}$$

Trapezoidal Rule

II. STABILITY

This greatly important property has 2 facets. We shall describe them qualitatively, leaving the formal mathematical definitions as optional reading (Recommended: CS412 notes, Heath "Scientific Computing: An Introductory Survey", J. Strikwerda "Finite Difference Schemes & Partial Differential Equations" Burden & Faires: "Numerical Analysis")

A. Is an ODE (or the physical process) it describes, stable?

⇒ Yes, when there is a "limit behavior" which does not greatly depend on the starting conditions of the system.

e.g. • A (free-)falling object will attain the same (vertical) terminal velocity (considering air drag) despite its velocity at launch

• An oscillating spring will ultimately come to rest, regardless of its original extension/velocity.

• In the case of the model ODE $y'(t) = \lambda y(t)$
$\lambda < 0$ is a stable behavior (exp. decay $\xrightarrow{\lim} 0$)
$\lambda > 0$ is an unstable case (exp. growth).

B. When trying to approximate the behavior of a stable ODE (most equations we will see in CS838 are stable), does the integration method capture the long-term behavior of the physical system?

⇒ If yes, the algorithm is practical and sustainable for modeling the phenomenon in question

⇒ If not, the algorithm produces unreliable (even completely useless, at times) results, when modeling a phenomenon over a non-trivial amount of time

★ For the model ODE $y'(t) = \lambda y(t)$, $\lambda < 0$ an integration scheme is <u>stable</u> if it reproduces the long-term behavior $y(t) \xrightarrow{t \to \infty} 0$. <u>Conveniently</u>, this is a quite reliable benchmark to assess if an integration scheme will be stable on <u>other stable ODEs, too</u>.

➢ Forward Euler (on $y'(t) = \lambda y(t)$, $\lambda < 0$)

$$y_{k+1} = y_k + dt\, f(t_k, y_k) \qquad [f(t,y) = \lambda y]$$

$$y_{k+1} = y_k + \lambda dt\, y_k = (1 + \lambda dt)\, y_k$$

$$\Rightarrow y_1 = (1+\lambda dt)\, y_0$$
$$y_2 = (1+\lambda dt)\, y_1 = (1+\lambda dt)^2 y_0$$
$$\vdots$$
$$y_k = (1+\lambda dt)^k y_0.$$

For stability, we want $y_k \xrightarrow[k\to\infty]{} 0$ $\rightsquigarrow$ $|1+\lambda dt| < 1$

$\rightsquigarrow$ $-1 < \overbrace{1+\lambda dt < 1}^{\text{Always true } (\lambda<0)}$ $\rightsquigarrow$ $-\lambda dt < 2$ $\rightsquigarrow$ $|\lambda| dt < 2$

$\rightsquigarrow$ $\boxed{dt < \frac{2}{|\lambda|}}$

▷ Trapezoidal Rule

$$y_{k+1} = y_k + \frac{dt}{2}\left\{f(t_k, y_k) + f(t_{k+1}, y_{k+1})\right\}$$

$$y_{k+1} = y_k + \frac{dt}{2}\left(\lambda y_k + \lambda y_{k+1}\right)$$

$$\left(1 - \frac{\lambda dt}{2}\right) y_{k+1} = \left(1 + \frac{\lambda dt}{2}\right) y_k$$

$$\Rightarrow \quad y_k = \left(\frac{1 + \lambda dt/2}{1 - \lambda dt/2}\right)^k y_k$$

For stability, we want $\left|\underbrace{\frac{1+\lambda dt/2}{1-\lambda dt/2}}_{>0}\right| < 1$

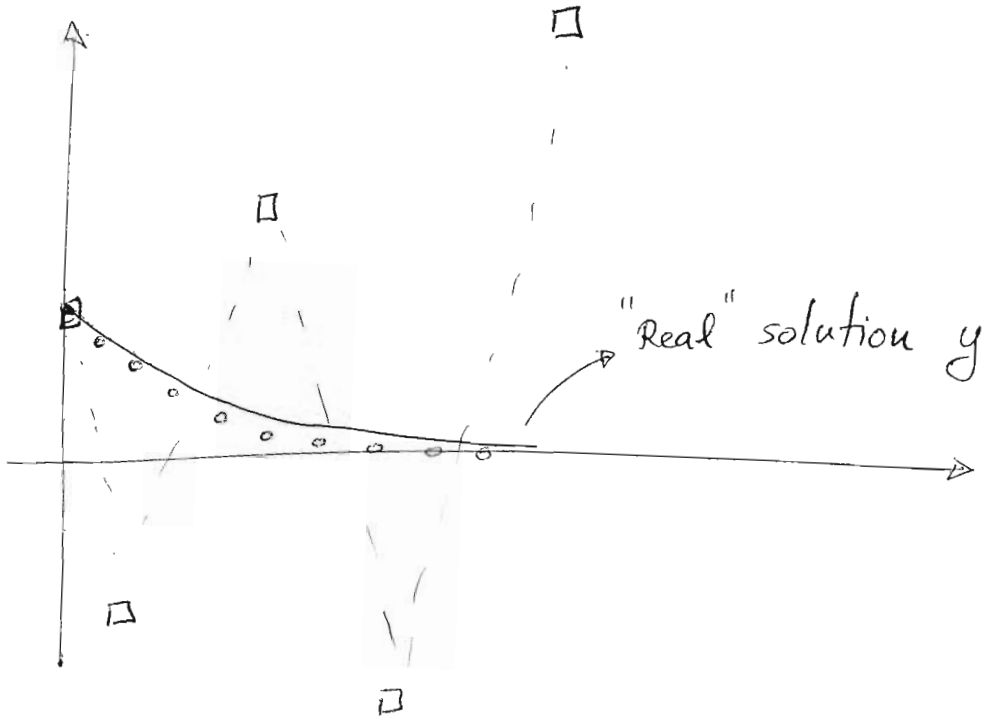
$$|1 + \lambda dt/2| < 1 - \lambda dt/2$$

$$\rightsquigarrow \quad -1 + \lambda dt/2 < \overbrace{1 + \lambda dt/2}^{\text{always true}} < \underbrace{1 - \lambda dt/2}_{\text{always true } (\lambda < 0)}$$

Thus trapezoidal rule is unconditionally stable, too.

III. ACCURACY

Ultimalely, we want to approximate the real solution to $y'(t) = f(t,y)$. As we devote more effort (i.e. smaller, more numerous time-steps dt), we would like the quality of our approximation to improve (and, ultimalely converge to the analytic solution when $dt \to 0$).



Backward Euler $(y' = \lambda y, \lambda < 0)$

$$y_{k+1} = y_k + dt\, f(t_{k+1}, y_{k+1})$$

$$y_{k+1} = y_k + \lambda dt\, y_{k+1}$$

$$(1 - \lambda dt)\, y_{k+1} = y_k \rightsquigarrow y_{k+1} = \frac{1}{1 - \lambda dt}\, y_k$$

$$\rightsquigarrow y_k = \left(\frac{1}{1 - \lambda dt}\right)^k y_0$$

When $\lambda < 0$, $1 - \lambda dt > 1 \rightsquigarrow \frac{1}{1 - \lambda dt} < 1$!

Thus $y_k \xrightarrow{k \to \infty} 0$ regardless of the value of dt!

We say that Backward Euler is unconditionally stable.

We use the following criterion to assess the accuracy of an integration scheme:

→ Assume y_k is the exact solution of $y' = f(t,y)$, at time $= t_k$

→ Let y_{k+1} be the exact solution at time t_{k+1}

→ Let $\hat{y}_{k+1}$ be the value we compute using an integration scheme, starting from the exact value y_k.

→ If $|\hat{y}_{k+1} - y_{k+1}| = O(dt^{d+1})$, the method is labelled as d-order accurate.

Again, we test accuracy on the model ODE $y' = \lambda y$, $\lambda < 0$

Exact solution:

$$y(t) = y_0 e^{\lambda t} \begin{cases} \rightarrow y_k = y_0 e^{\lambda t_k} \\ \rightarrow y_{k+1} = y_0 e^{\lambda t_{k+1}} = y_0 e^{\lambda(t_k + dt)} = y_0 e^{\lambda t_k} e^{\lambda dt} \end{cases}$$

$$\Rightarrow y_{k+1} = y_k e^{\lambda dt} = y_k \left(1 + \lambda dt + \frac{\lambda^2 dt^2}{2} + \frac{(\lambda dt)^3}{6} + \cdots\right) \quad (1)$$

Forward Euler:

$$\hat{y}_{k+1} = (1 + \lambda dt) y_k \quad (2) \Rightarrow |\hat{y}_{k+1} - y_{k+1}| = O(dt^2)$$

$\Rightarrow$ 1st order

Backward Euler:

$$\hat{y}_{K+1} = y_K \frac{1}{1-\lambda dt} = y_K \left(1 + \lambda dt + (\lambda dt)^2 + (\lambda dt)^3\right)$$

$\checkmark \quad \checkmark \quad \neq \frac{(\lambda dt)^2}{2}$

$$\Rightarrow |\hat{y}_{K+1} - y_{K+1}| = O(dt^2) \Rightarrow \text{1st order}$$

Trapezoidal:

$$\hat{y}_{K+1} = y_K \frac{1+\lambda dt/2}{1-\lambda dt/2} = y_K \left(1 + \lambda dt + \frac{(\lambda dt)^2}{2} + \frac{(\lambda dt)^3}{4} + \frac{(\lambda dt)^4}{8} + \ldots\right)$$

$\checkmark \quad \checkmark \quad \checkmark \quad \neq \frac{(\lambda dt)^3}{6}$

$$\Rightarrow |\hat{y}_{K+1} - y_{K+1}| = O(dt^3) \Rightarrow \text{2nd order}$$

Notes:

→ A method's improved order of accuracy translates into better performance only at the limit of small dt. (There are many 3rd & 4th order accurate explicit methods, that will easily go unstable unless dt is small enough).

→ The error described above is the "local" error (i.e. starting from an exact y_K and transitioning to y_{K+1}). There is also a "global" notion:

→ Starting from y_0 and transitioning to $y(T_{max})$ after taking a number of dt-steps. The difference $|y_{exact}(T_{max}) - y_{computed}(T_{max})|$ is called the global error, and is bounded by $O(dt^d)$ for d-order accuracy.

IV. OSCILLATORY BEHAVIOR

For implicit & unconditionally stable methods:

→ What happens if we use a VERY LARGE dt?

B.E. $y_{k+1} = \frac{1}{1-\lambda dt} y_k \xrightarrow[dt \to \infty]{} 0$ (goes to zero rapidly)

T.R. $y_{k+1} = \frac{1+\lambda dt/2}{1-\lambda dt/2} y_n \xrightarrow[dt \to \infty]{} -y_k$ (oscillates)

Summary

	Forward Euler	Backward Euler	Trap. Rule
Implicit/Explicit	Explicit	Implicit	Implicit
Stability	Only when $dt < 2/\lvert\lambda\rvert$	Unconditional	Unconditional
Accuracy	1st order	1st order	2nd order
Behavior for LARGE dt?	Unstable	Rapid Decay	Oscillatory.