Review : Methods for solving $y'(t) = f(t,y)$

* Forward Euler : $y_{n+1} = y_n + dt\, f(t_n, y_n)$
  ↳ Explicit, conditionally stable, 1st order accurate
* Backward Euler : $y_{n+1} = y_n + dt\, f(t_{n+1}, y_{n+1})$
  ↳ Implicit, unconditionally stable, 1st order accurate
* Trapezoidal Rule : $y_{n+1} = y_n + \frac{dt}{2}\left\{ f(t_n, y_n) + f(t_{n+1}, y_{n+1}) \right\}$
  ↳ Implicit, unconditionally stable, 2nd order accurate.

IV. (The last property) OSCILLATORY BEHAVIOR
(For B.E. & T.R. only)

Implicit methods allow us (in theory) to take arbitrarily large steps $dt \gg 1$. But, what happens if we do?

Check on model equation $y' = \lambda y$

B.E.

$$y_{n+1} = y_n + \lambda dt\, y_{n+1} \Rightarrow y_{n+1} = \frac{1}{1-\lambda dt} y_n \xrightarrow[dt \to \infty]{} 0$$

TR.

$$y_{n+1} = \frac{1 + \lambda dt/2}{1 - \lambda dt/2} y_n \xrightarrow[dt \to \infty]{} -y_n$$

Thus, when using large timesteps, B.E. tends to quickly settle on the steady-state solution, while trapezoidal rule may oscillate for a prolonged period before settling.

Systems of ODE's

We often have systems of differential equations, with more than one unknown function. E.g

(Nonlinear)

$$y_1'(t) = f_1(t, y_1(t), y_2(t), \dots, y_n(t))$$
$$y_2'(t) = f_2(t, y_1(t), y_2(t), \dots, y_n(t))$$
$$\vdots$$
$$y_n'(t) = f_n(t, y_1(t), y_2(t), \dots, y_n(t)).$$

A special case arises when each function f_i is <u>linear</u> in the unknown functions $y_j(t)$, i.e. $f_i = c_{i1}y_1(t) + c_{i2}y_2(t) + \dots + c_{in}y_n(t)$

e.g.

$$\left.\begin{aligned} y_1' &= a_{11}y_1 + a_{12}y_2 + a_{13}y_3 \\ y_2' &= a_{21}y_1 + a_{22}y_2 + a_{23}y_3 \\ y_3' &= a_{31}y_1 + a_{32}y_2 + a_{33}y_3 \end{aligned}\right\} \Rightarrow \underbrace{\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}'}_{\vec{y}'(t)} = \underbrace{\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}}_{A} \cdot \underbrace{\begin{pmatrix} y_1(t) \\ y_2(t) \\ y_3(t) \end{pmatrix}}_{\vec{y}(t)}.$$

In each case, we can extend the concept of an integration method to a system of ODEs (linear or nonlinear) in the "obvious" fashion:

g

$$y_1' = f_1(t, y_1, y_2)$$
$$y_2' = f_2(t, y_1, y_2)$$

→ Using forward Euler

$$y_1^{n+1} = y_1^n + dt\, f_1(t^n, y_1^n, y_2^n)$$
$$y_2^{n+1} = y_2^n + dt\, f_2(t^n, y_1^n, y_2^n)$$

→ Using B.E.

$$\underbrace{\begin{aligned} y_1^{n+1} &= y_1^n + dt\, f_1(t^{n+1}, y_1^{n+1}, y_2^{n+1}) \\ y_2^{n+1} &= y_2^n + dt\, f_2(t^{n+1}, y_1^{n+1}, y_2^{n+1}) \end{aligned}}_{\text{Needs solver for system of nonlinear equations}}$$

2 Linear case $\vec{y}' = A\vec{y}$ (or $y' = Ay$ for brevity)

$$y \in \mathbb{R}^n,\ A \in \mathbb{R}^{n\times n}$$

Forward Euler: $y^{n+1} = y^n + dt\, Ay^n \Rightarrow y^{n+1} = (I + dt A)\, y^n$

Backward Euler: $y^{n+1} = y^n + dt\, Ay^{n+1} \Rightarrow (I - dtA)\, y^{n+1} = y^n$

(or $y^{n+1} = (I - dtA)^{-1} y^n$) $\Rightarrow$ Need to solve a linear system.

Trapezoidal Rule:

$$y^{n+1} = y^n + \frac{dt}{2}\left[Ay^n + Ay^{n+1}\right] \Rightarrow$$

$$\Rightarrow \underbrace{\left(I - \frac{dt}{2}A\right) y^{n+1} = \left(I + \frac{dt}{2}A\right) y^n}_{\text{Linear system to solve}}$$

The properties of an integration rule mostly carry over from the scalar case to the case of ODE systems.
i.e.

↳ If we use B.E. or T.R to solve a system of ODEs, the resulting method will be unconditionally stable

↳ The order of accuracy in a system mirrors the order of accuracy observed on $y' = \lambda y$.

Some questions remain more complicated, e.g:

→ An ODE $y' = \lambda y$ had stable solutions when $\lambda < 0$ (i.e. the solutions exhibited exponential decay).
What happens with systems?

Answer: This can be answered easily for the case of linear systems, i.e. $y' = Ay$.

Remember: λ is an eigenvalue of A, iff $\det(A - \lambda I) = 0$.
Eigenvalues can be complex numbers, and found by solving the polynomial equation of degree n: $\det(A - \lambda I) = 0$.

Theorem: If $Re\{\lambda_i\} < 0$ for all eigenvalues of A, the solutions are stable (i.e. decay to zero).

→ An integration method for $y' = \lambda y$ was ultimately written as $y_{n+1} = k \cdot y_n$, and $|k| < 1$ was the condition for stability. What happens with systems?

Answer: An integration scheme for $\vec{y}' = A\vec{y}$ is also ultimately written as $y^{n+1} = K y^n$

(i.e. for F.E $K = I + dtA$, for B.E. $K = (I - dtA)^{-1}$).

The stability condition translates to $\|\lambda_i\| < 1$ (complex magnitude) for any eigenvalue λ_i of K.

POSITION / VELOCITY SYSTEMS

We previously saw that a mass-spring system is governed by the 2nd order ODE

$$f(t, x, v) = ma \quad \text{or} \quad f(t, x(t), x'(t)) = m x''(t)$$

We then converted this equation to the 1-st order system

$$\begin{pmatrix} x(t) \\ v(t) \end{pmatrix}' = \begin{pmatrix} v(t) \\ \frac{1}{m} f(t, x(t), v(t)) \end{pmatrix} \begin{matrix} = F_1(t, x, v) \\ = F_2(t, x, v) \end{matrix}$$

By directly mapping the previous methods to this system, we get, e.g.

Forward Euler

$$x^{n+1} = x^n + dt\, v^n$$

$$\underbrace{v^{n+1} = v^n + \frac{dt}{m} f(t^n, x^n, v^n)}_{\text{Easy implementation -- - we already demonstrated it}}$$

Backward Euler

$$x^{n+1} = x^n + dt\, v^{n+1}$$

$$\underbrace{v^{n+1} = v^n + \frac{dt}{m} f(t^{n+1}, x^{n+1}, v^{n+1})}_{\text{Somewhat unclear how to solve!}}$$

We will soon examine how the implicit system can be solved in practice; in the meantime, we examine yet another method which becomes an option for ODE systems:

⇒ We can design new integration methods by mixing/matching elements from different methods, for the distinct equations of the system.

e.g.

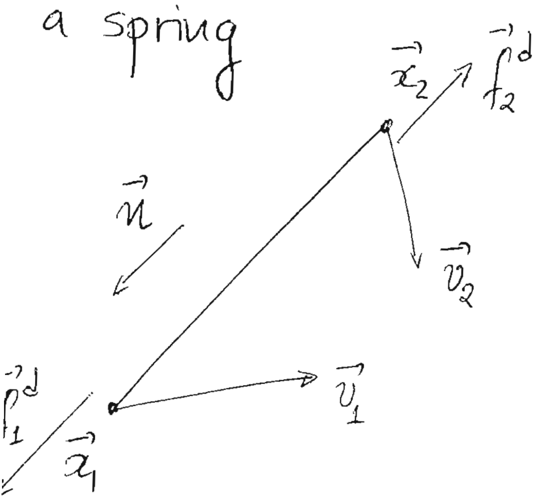
$$x^{n+1} = x^n + \frac{dt}{2}\left\{v^n + v^{n+1}\right\} \quad \text{Trapezoidal-like}$$

$$v^{n+1} = v^n + \frac{dt}{m} f(x^n, v^{n+1}) \quad \begin{array}{l}\text{F.E.-like for } \underline{x} \\ \text{B.E-like for } \underline{v}\end{array}$$

These methods combine features of their "component" methods. For example, this method manages to be 2nd order overall, while the dt restriction depends <u>only</u> on k (young's modulus) and not on b

Additionally, this "hybrid" method will make it easier for us to describe a practical method for computing the values x^{n+1} & v^{n+1}.

We start by revisiting the damping force incurred by a spring



$$\vec{f}_1^d = -b n n^T (\vec{v}_1 - \vec{v}_2)$$
$$\vec{f}_2^d = -\vec{f}_1$$

$$\left(\text{where } \vec{n} = \frac{\vec{x}_1 - \vec{x}_2}{\|\vec{x}_1 - \vec{x}_2\|} \right)$$

Let us manipulate this definition, a bit:

$$\vec{f}_1^d = -b\vec{n}\vec{n}^T \vec{v}_1 + b\vec{n}\vec{n}^T \vec{v}_2$$
$$\vec{f}_2^d = b\vec{n}\vec{n}^T v_1 - b\vec{n}\vec{n}^T \vec{v}_2$$

We observe that we can write this in matrix form:

$$\underbrace{\begin{pmatrix} f_1^d \\ f_2^d \end{pmatrix}}_{\mathbb{R}^6} = \underbrace{\begin{pmatrix} -bnn^T & bnn^T \\ bnn^T & -bnn^T \end{pmatrix}}_{\mathbb{R}^{6\times 6}} \underbrace{\begin{pmatrix} v_1 \\ v_2 \end{pmatrix}}_{\mathbb{R}^6}$$

More generally, if this spring was connecting particles i & j the resulting damping force could be computed as:

$$\underbrace{\begin{pmatrix} f_1^d = 0 \\ \vdots \\ f_{i-1}^d = 0 \\ f_i^d \neq 0 \\ f_{i+1}^d = 0 \\ \vdots \\ f_j^d \neq 0 \\ \vdots \\ f_n^d = 0 \end{pmatrix}}_{F^d} = \underbrace{\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ \cdots 0 & -b\,n_{ij}n_{ij}^T & \cdots 0 \cdots & b\,n_{ij}n_{ij}^T & \cdots 0 \cdots \\ 0 & 0 & 0 & 0 & 0 \\ \cdots 0 \cdots & b\,n_{ij}n_{ij}^T & \cdots 0 \cdots & -b\,n_{ij}n_{ij}^T & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}}_{G_{ij}} \underbrace{\begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_i \\ \vdots \\ v_j \\ \vdots \\ v_n \end{pmatrix}}_{V}$$

where $n_{ij} = (x_i - x_j)/\|x_i - x_j\|$

By adding all the matrices from all springs we get:

$$G = \sum_{(i,j) \text{ is a spring}} G_{ij}$$

and finally $\vec{F}^d = G \cdot \vec{V}$ for all damping forces, collectively

To be precise: G is not a constant; it depends on the positions (X), via the normals n_{ij}. Overall we can write:

$$f^d(x,v) = G(x) \cdot v$$

Adding the damping forces to the elastic forces (f^{el}) we get:

$$f(x,v) = f^{el}(x) + f^{d}(x,v)$$
$$= f^{el}(x) + G(x)\cdot v$$

Thus, the second equation of our "hybrid" integration rule becomes:

$$v^{n+1} = v^{n} + \frac{dt}{m} f(x^{n}, v^{n+1})$$
$$= v^{n} + \frac{dt}{m}\left\{ f^{el}(x^{n}) + G(x^{n})\, v^{n+1} \right\}$$

$$\Rightarrow \underbrace{\left(I - \frac{dt}{m} G(x^{n})\right) v^{n+1} = v^{n} + \frac{dt}{m} f^{el}(x^{n})}$$

After solving this linear system for v^{n+1}, we substitute into

$$x^{n+1} = x^{n} + \frac{dt}{2}\left\{ v_n + v_{n+1} \right\}$$ to get x^{n+1}!

Implementation demonstration $\Rightarrow$ next class!