Initial value problems

- Recap:
  - Trying to solve problem of the form $\frac{dy}{dt} = f(t, y)$
    - $y(t)$ is unknown
    - Know initial value, $y(t_0)$
    - $f(t, y)$ is known
  - Often we have multiple such first-order eqs. we need to solve <u>simultaneously</u> (<u>coupled</u> eqs.) (Proj. 3)

- Last time:
  - Forward Euler
    - Local error $O(h^2)$, global error $O(h)$
    - <u>One</u> eval. of f per step
  - Predictor - Corrector
    - Local error $O(h^3)$, global error $O(h^2)$
    - <u>Two</u> evals. of f pr step.
  - Runge-Kutta 4th order
    - Local error $O(h^5)$, global error $O(h^4)$
    - <u>Four</u> evals of f pr step

- Today:
  - Proof of error in Pred. - Corr.
  - Leapfrog
  - Verlet

Predictor – Corrector method

- Simple improvement to Euler
- Also a single-step method (only requires knowing Y_i)

Algorithm

1) Predict: $Y^*_{i+1} = Y_i + h f_i$

$$\hookrightarrow f^*_{i+1} = f(t_{i+1}, Y^*_{i+1})$$

2) Correct: $Y_{i+1} = Y_i + h \dfrac{f^*_{i+1} + f_i}{2}$

Alt. notation:

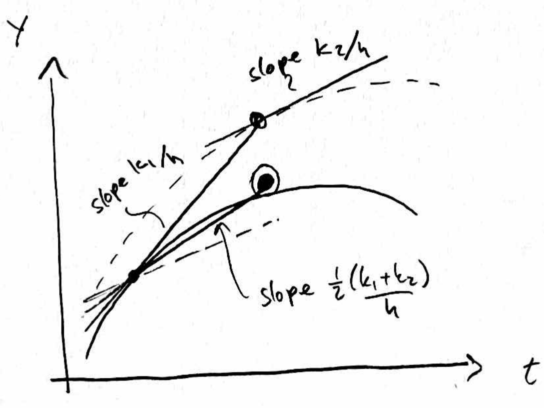
$$k_1 \equiv h f_i = h f(t_i, Y_i)$$

$$k_2 \equiv h f^*_{i+1} = h f(t_{i+1}, Y^*_{i+1})$$

$$\Rightarrow \quad Y_{i+1} = Y_i + \frac{\cancel{h}\,1}{2}(k_1 + k_2)$$

- Euler uses gradient at a single point (f_i) to predict next point.
- Could improve by using <u>average</u> gradient between the two points t_i and t_{i+1}
- We <u>want</u> $Y_{i+1} = Y_i + h\left(\dfrac{f_i + f_{i+1}}{2}\right)$

  but we don't know f_{i+1}. But we can <u>predict it</u> using a simple forward Euler step.



(Predictor - Corrector cont.)

- Local error (trunc.) is $\mathcal{O}(h^3)$ $\Rightarrow$ Global error $\mathcal{O}(h^2)$

- One order better than FE, but also req. one extra evaluation of f (at $f(t_{i+1}, Y^*_{i+1})$)

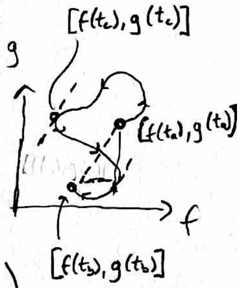
Proof:

$$Y(t+h) = Y(t) + hY'(t) + \tfrac{1}{2}h^2 Y''(t) + \underbrace{R_2(t)}_{\mathcal{O}(h^3)}$$

Remainder

- Know from Mean Value Theorem (Cauchy's MVT, a.k.a. extended MVT) that there exist a $\xi \in [t, t+h)$ such that

$$\underset{\text{exact!}}{Y(t+h) =} Y(t) + hY'(t) + \tfrac{1}{2}h^2Y''(t) + \tfrac{1}{3!}h^3 Y'''(\xi)$$



Cauchy's MVT:
There exists a t_c such that
$$\frac{g'(t_c)}{f'(t_c)} = \frac{g(t_b)-g(t_a)}{f(t_b)-f(t_a)}$$

- Use $Y'(t) = f(t)$:

$$Y(t+h) = Y(t) + hf(t) + \tfrac{1}{2}h^2 f'(t) + \tfrac{1}{3!}h^3 f''(\xi)$$

- Replace $f'(t)$ with a forward diff. + remainder (MVT again...)

$$f'(t) = \frac{f(t+h) - f(t)}{h} - \tfrac{1}{2}hf''(\eta)$$

$$\Rightarrow Y(t+h) = Y(t) + hf(t) + \tfrac{1}{2}h^2\left[\frac{f(t+h)-f(t)}{h} - \tfrac{1}{2}hf''(\eta)\right] + \tfrac{1}{3!}h^3 f''(\xi)$$

$$= Y(t) + hf(t) + \tfrac{1}{2}h\left[f(t+h)-f(t)\right] \underbrace{- \tfrac{1}{4}h^3 f''(\eta) + \tfrac{1}{3!}h^3 f''(\xi)}_{\mathcal{O}(h^3)}$$

$$\boxed{Y(t+h) = Y(t) + h\frac{f(t+h)-f(t)}{2} + \mathcal{O}(h^3)}$$

$$\boxed{Y_{i+1} = Y_i + h\frac{f_{i+1}-f_i}{2} + \mathcal{O}(h^3)}$$

- So, the local (trunc.) error is $\mathcal{O}(h^3)$, giving a global error $\mathcal{O}(h^2)$

Leapfrog (multi-step) method

$$\frac{dy}{dt} = f(t,y)$$

$$y_i' = f_i$$

- Recall approximations for first derivatives
  - $Y'(t) = \frac{Y(t+h) - Y(t)}{h} + O(h)$ Forward difference
  - $Y'(t) = \frac{Y(t) - Y(t-h)}{h} + O(h)$ Backward difference
  - $Y'(t) = \frac{Y(t+h) - Y(t-h)}{2h} + O(h^2)$ Central difference

- Leapfrog idea: Use the central difference scheme combined with previously computed y values

- Discretized differential eq.

$$Y_i' = \frac{Y_{i+1} - Y_{i-1}}{2h} + O(h^2) = f_i$$

$$\Rightarrow \quad Y_{i+1} = Y_{i-1} + 2hf_i + O(h^3)$$

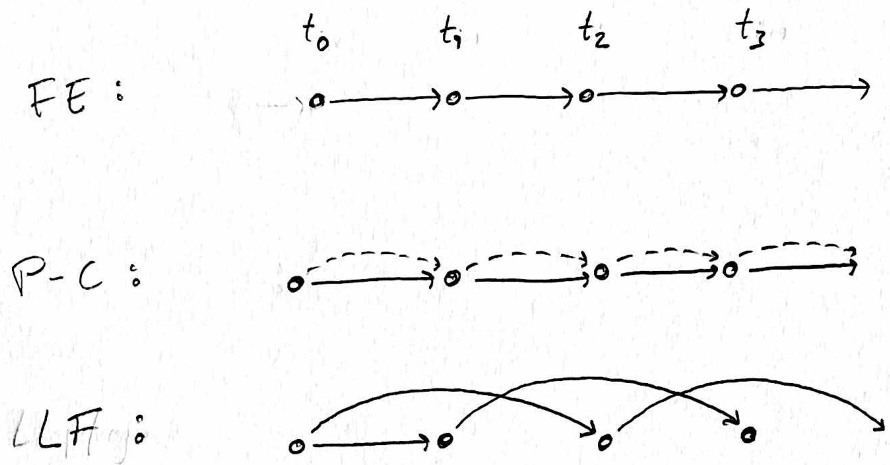
- Algorithm: $\boxed{Y_{i+1} = Y_{i-1} + 2hf_i}$ (At this point we have the values $Y_0, Y_1, \ldots, Y_{i-1}, Y_i$)

  - Local error $O(h^3)$, global error $O(h^2)$ (same as P-C)
  - Only requires one evaluation of $f(t,y)$ per step. (more efficient than P-C)
  - But, requires us to also keep track of previous step Y_{i-1} in order to compute Y_{i+1}. (So not a single-step method)

- Not "self-stating", since it needs both Y_0 and Y_1 to compute Y_2.
- Solution: Start with a simple FE step, or similar.

- Can be generalized to use more previous steps ...

- Pictorial comparison of FE, P-C and LF:



Leapfrog for coupled equations

- Leapfrog and Verlet algorithms particularly popular for solving Newton's second law when the force does not dep. on velocity

$$\frac{d^2x}{dt^2} = \frac{1}{m} F(t,x) \equiv a(t,x)$$

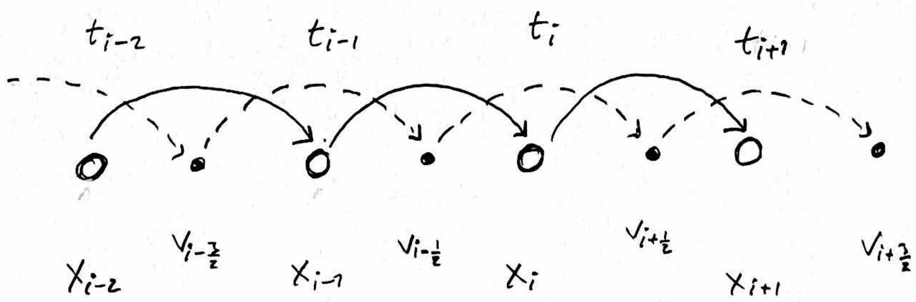
known

- Computationally cheap and very stable — conserve energy (symplectic)
- Example: can simulate planet orbits for long time-periods with orbits drifting
- Now: Leapfrog
- Get two first-order eqs:

$$\frac{dv}{dt} = a(t,x)$$

$$\frac{dx}{dt} = v(x,t)$$

- Will solve these using a Leapfrog pattern with x and v



- Notation: $t_{i-\frac{1}{2}} \equiv t_i - \frac{1}{2}h$ such that $x_{i-\frac{1}{2}} = x(t-\frac{1}{2}h)$

$v_{i-\frac{1}{2}} = v(t-\frac{1}{2}h, x(t-\frac{1}{2}h))$

etc.

x end of lecture

- Algorithm is as follows:

$$\boxed{\begin{aligned} v_{i+\frac{1}{2}} &= v_{i-\frac{1}{2}} + a_i h \\ x_{i+1} &= x_i + v_{i+\frac{1}{2}} h \end{aligned}}$$

→ Uses acceleration at midpoint between $t_{i-\frac{1}{2}}$ and $t_{i+\frac{1}{2}}$

→ Uses velocity at midpoint between t_i and t_{i+1}

  - If we need velocity at t_{i+1}, can compute $\boxed{v_{i+1} = v_{i+\frac{1}{2}} + a_{i+1}\frac{h}{2}}$

  - Local error $O(h^3)$, global error $O(h^2)$

- <u>Derivation of algorithm</u>

  - Taylor exp. of $x(t+h)$:

$$\begin{aligned} x(t+h) &= x(t) + h x'(t) + \tfrac{1}{2}h^2 x''(t) + O(h^3) \\ &= x(t) + h\left[x'(t) + \tfrac{1}{2}h x''(t)\right] + O(h^3) \qquad (1) \end{aligned}$$

  - Taylor exp of $x'(t+\frac{1}{2}h)$ (step $\frac{1}{2}h$):

$$x'(t+\tfrac{1}{2}h) = \underbrace{x'(t) + \tfrac{1}{2}h\, x''(t)} + O(h^2) \qquad (2)$$

- Insert (2) into (1):

$$x(t+h) = x(t) + h\left[x'(t+\tfrac{1}{2}h) + O(h^2)\right] + O(h^3)$$

$$\boxed{x(t+h) = x(t) + h\,x'(t+\tfrac{1}{2}h) + O(h^3)}$$

[Needs velocity at midpoint between t and $t+h$]

  - Discretize
  - Approximate

$$\boxed{x_{i+1} = x_i + h\,v_{i+\frac{1}{2}}}$$

- Now use the central difference scheme for $x''(t)$, with step $(\frac{1}{2}h)$, to obtain expression for $x'(t+\frac{1}{2}h)$:

$$x''(t) = \frac{x'(t+\frac{1}{2}h) - x'(t-\frac{1}{2}h)}{2(\frac{1}{2}h)} + \mathcal{O}(h^2)$$

$$\Rightarrow \quad \boxed{x'(t+\tfrac{1}{2}h) = x'(t-\tfrac{1}{2}h) + h\,x''(t) + \mathcal{O}(h^3)}$$

Uses acceleration at midpoint between $t-\frac{1}{2}h$ and $t+\frac{1}{2}h$

- Discretize
- Approx.

$$\boxed{v_{i+\frac{1}{2}} = v_{i-\frac{1}{2}} + h\,a_i}$$

- If we need velocity at same time-step as position :

$$x'(t+h) = x'(t+\tfrac{1}{2}h) + (\tfrac{1}{2}h)\,x''(t+h) + \mathcal{O}(h^2)$$

$$\Rightarrow \quad \boxed{v_{i+1} = v_{i+\frac{1}{2}} + \tfrac{1}{2}h\,a_{i+1}}$$

Verlet algorithm

- Again consider $\frac{d^2x}{dt^2} = \frac{1}{m}F(x,t) \equiv a(t,x)$ (not dep. on velocity)

which gives rise to $\frac{dv}{dt} = a(x,t)$ and $\frac{dx}{dt} = v(x,t)$, but now let's consider the second-order diff eq. directly. Two alt. derivations:

Alt. der 2)

- Recall expr. for second derivative
$$x''(t) = \frac{x(t+h) - 2x(t) + x(t-h)}{h^2} + \mathcal{O}(h^2)$$
- Rearrange for $x(t+h)$ to get

Alt. der. 1)

- Look at Taylor exp. for $x(t+h)$ and $x(t-h)$:

$$x(t+h) = x(t) + h\,x'(t) + \frac{1}{2}h^2 x''(t) + \mathcal{O}(h^3)$$
$$x(t-h) = x(t) - h\,x'(t) + \frac{1}{2}h^2 x''(t) - \mathcal{O}(h^3)$$

- Add the eqs:

$$x(t+h) + x(t-h) = 2\,x(t) + h^2 x''(t) + \mathcal{O}(h^4)$$

Note!

$$\boxed{x(t+h) = 2\,x(t) - x(t-h) + h^2 x''(t) + \mathcal{O}(h^4)}$$

- Discretize
- Approx.

$$\boxed{x_{i+1} = 2\,x_i - x_{i-1} + h^2 a_i}$$

- Local error in position is $\mathcal{O}(h^4)$
- But global error is $\mathcal{O}(h^2)$ Note!

(Cumulative error after n steps is $\frac{n(n+1)}{2}\mathcal{O}(h^4) = \mathcal{O}(h^2)$)

- The velocity is not needed/included in update formula for x_{i+1}!
- Can compute it as $\boxed{v_i = \frac{x_{i+1} - x_{i-1}}{2h}}$ with both local and global error $\mathcal{O}(h^2)$

Note: Different timestep than x_{i+1}!

Velocity Verlet

- Very similar to Leapfrog w/ computation of velocity at same timestep as position
- Now view problem as two first-order eqs.: $\frac{dx}{dt} = v$, $\frac{dv}{dt} = a$
- VV gives exactly same trajectory of x_i points as original Verlet
- Standard algorithm:

$$\boxed{\begin{aligned} &1. \quad v_{i+\frac{1}{2}} = v_i + \tfrac{1}{2} h a_i \\ &2. \quad x_{i+1} = x_i + h v_{i+\frac{1}{2}} \\ &3. \quad \text{Use } x_{i+1} \text{ to obtain } a_{i+1} = a(t_{i+1}, x_{i+1}) \\ &4. \quad v_{i+1} = v_i + h\left(\frac{a_i + a_{i+1}}{2}\right) \end{aligned}}$$

  - End up with both x_{i+1} and v_{i+1}
  - Global error $\mathcal{O}(h^2)$

- Can combine step 1. and 2. as

$$\boxed{x_{i+1} = x_i + h v_i + \tfrac{1}{2} h^2 a_i} \longleftrightarrow$$

To see equivalence with original Verlet algo., insert $v_i = \frac{x_{i+1} - x_{i-1}}{2h} + \mathcal{O}(h^2)$

- Assumes that the acceleration a_{i+1} only depends on x_{i+1}, not on v_{i+1}, so that we can do step 3 before step 4.

- This is why we don't use this in Project 3... $(\vec{F} = q\vec{E} + q\vec{v} \times \vec{B})$

{ Mention "magnetic Verlet" }

- Like Leapfrog:
  - Good stability
  - Conserves energy (symplectic)
  - Reversible in time.

∘ Derivation of velocity Verlet

Notation: $a(t, x(t)) \equiv a(t)$

∘ Expression for x_{i+1} comes directly from Taylor expansion of $x(t+h)$

$$x(t+h) = x(t) + h x'(t) + \frac{1}{2} h^2 x''(t) + \mathcal{O}(h^3)$$

$$= x(t) + h v(t) + \frac{1}{2} h^2 a(t) + \mathcal{O}(h^3)$$

∘ Discretize
∘ Approximate

$$\Rightarrow \quad \boxed{x_{i+1} = x_i + h v_i + \frac{1}{2} h^2 a_i}$$

∘ To find expression for v_{i+1} we start from Taylor exp. of $v(t+h)$

$$v(t+h) = v(t) + h v'(t) + \frac{1}{2} h^2 v''(t) + \mathcal{O}(h^3) \qquad (*)$$

∘ We know $v'(t) = a(t)$, but need expression for $v''(t)$ in terms of known quantities. Use a simple forward difference:

$$v''(t) = \frac{v'(t+h) - v(t)}{h} + \mathcal{O}(h)$$

∘ Insert into (*) to get

$$v(t+h) = v(t) + h v'(t) + \frac{1}{2} h^2 \left[\frac{v'(t+h) - v(t)}{h} + \mathcal{O}(h) \right] + \mathcal{O}(h^3)$$

$$v(t+h) = v(t) + \frac{1}{2} h v'(t) + \frac{1}{2} h v'(t+h) + \mathcal{O}(h^3)$$

$$= v(t) + h \left[\frac{a(t) + a(t+h)}{2} \right] + \mathcal{O}(h^3)$$

∘ Discretize
∘ Approximate

$$\Rightarrow \quad \boxed{v_{i+1} = v_i + h \left[\frac{a_i + a_{i+1}}{2} \right]}$$

Note that $a_{i+1} = a(t_{i+1}, x_{i+1})$ requires that we compute x_{i+1} first!

Stability (Need to match method to problem)

- A method is stable if the amplification factor g satisfies

$$g \equiv \left| \frac{\Delta_{i+1}}{\Delta_i} \right| \leq 1$$

where $\Delta_i = Y_i^{true} - Y_i^{approx}$

(absolute error)

- So $g > 1$ means that the absolute (and relative) error grows for every step $\rightarrow$ unstable!

- Some examples:
  - FE and P-C are conditionally stable for decaying solutions (requirement on h being small enough) $(y' = -\alpha y)$
  - LF is unstable for decaying solutions [see example]
  - P-C is unstable for pure oscillating solutions $(y'' = -\omega^2 y)$
  - FE, P-C, LF, RK4 are all unstable for an exp. growing solution $(y' = \alpha y)$ [see example]

- In short: Need to investigate how suitable an algorithm is for the given problem.