- Next methods:
  - Leapfrog
  - Verlet

- Typically used for physics simulations based on Newton's second law

  2nd order diff. eq. $\Longrightarrow$ Two coupled 1st order diff. eqs.

- Good properties:

  - Symplectic: Conserve energy, get stable orbits if simulating orbital systems

    Ex: Simulate planetary orbits. Can run for a long time without having the planets "drift away" due to numerical instability

- Drawback:

  - Assume that the force (accel.) is <u>not</u> dependent on the velocity.
    - OK for e.g. a gravitational force
    - Not OK for our Project 3 due to Lorentz force: $\vec{F} = q\vec{E} + q\vec{v} \times \vec{B}$

- Ongoing research: How to best extend applicability of such methods (e.g. "Magnetic Verlet")

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)

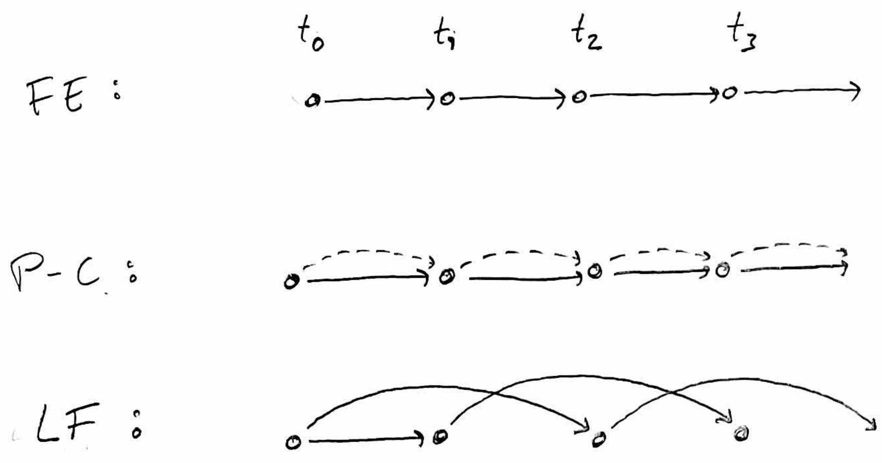
  - Only requires one evaluation of $f(t,y)$ per step. (more efficient than P-C)

  - 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 <u>not</u> dep. on velocity

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

- (computationally cheap and very stable — conserve energy (symplectic))
- Example: can simulate planet orbits for long time-periods with orbits drifting

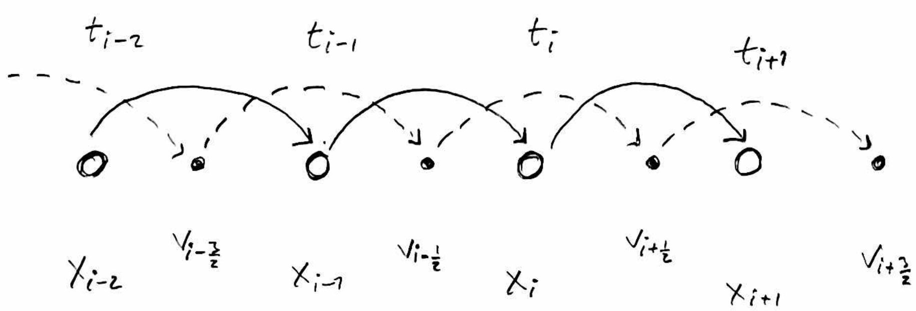
- Now: Leapfrog

- 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 <u>x and v</u>



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

$$V_{i-\frac{1}{2}} = V(t-\tfrac{1}{2}h,\ X(t-\tfrac{1}{2}h))$$

etc.

× 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}

Remember: $a_i = a(x_i, t_i)$

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

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

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

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 :

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

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

- Typically used to simulate problems with <u>conservative forces</u> (force only depends on position)

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 $x(t-h)$:

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

- Add the eqs:

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

Note!

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

- Discretize
- Approx.

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

- 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:

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

$\left.\begin{matrix}1a)\\1b)\end{matrix}\right\} x_{i+1} = x_i + h v_i + \frac{1}{2} h^2 a_i$

[Note: Here we rely on the fact that a does not dep. on velocity. (Conservative force)]

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

- Can combine step 1. and 2. as

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

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 2 before step 3

- 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

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

- 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

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

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

- Conserves energy
- Reversible (as is Leapfrog)

- Algorithms can be classified using different concepts

  - Consistency:

    when $h \neq 0$, an algorithm is effectively solving a slightly different diff. eq., called the modified diff. eq.

    An algorithm is consistent if

    modified diff. eq. $\longrightarrow$ original diff. eq.

    when $h \to 0$

  - Order of global error

  - One-step vs multi-step

  - Stability (next page)

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 $\longrightarrow$ unstable!
- (But unstable algos. can still be useful! But can't be run too long.)

- Some examples:
  - FE and P-C are conditionally stable for decaying solutions ($y' = -ay$) (requirement on h being small enough)
  - 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 ($y' = ay$) [see example]

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

- Compromise between efficiency, accuracy and stability