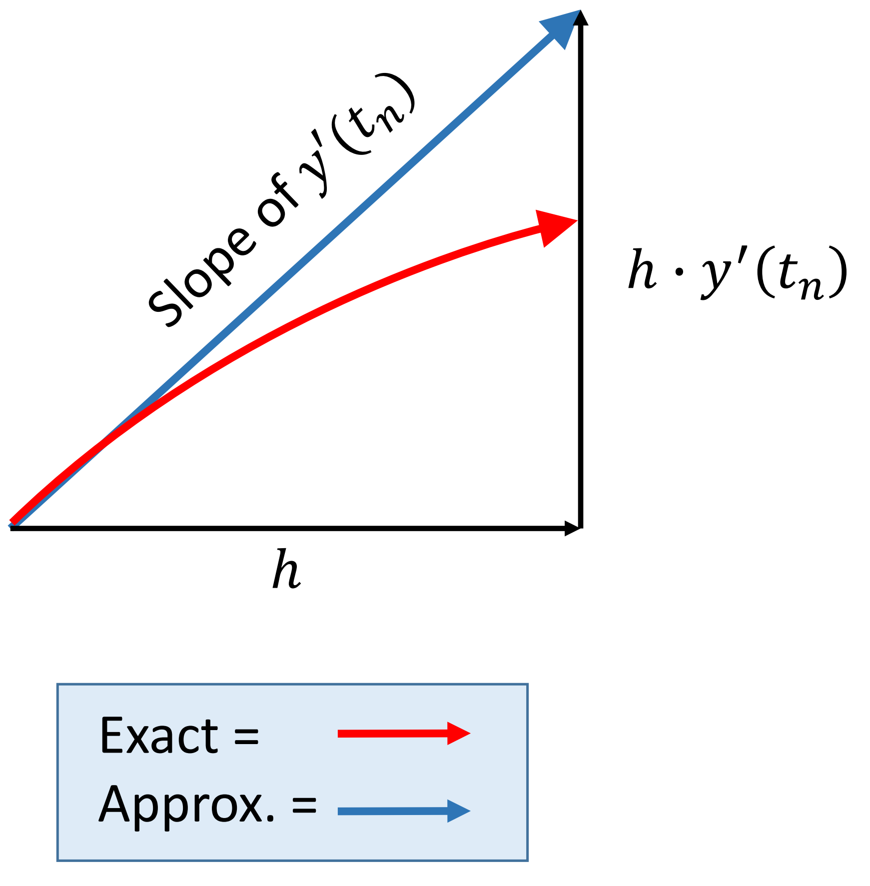
Understanding forward Euler error

Recall: Forward Euler makes a linear approximation at each step.

Smaller step size h → more frequent estimates of the slope → less error in approximate solution.

Let's determine this error!



More accurate time-stepping

Use of Taylor expansions hints at how to derive *higher order* schemes.

$$y(t_{n+1}) = \underbrace{\overbrace{y(t_n) + h \cdot y'(t_n)}^{\text{Forward Euler: } O(h^2)} + \frac{h^2}{2} y''(t_n)}_{\text{A better method, } O(h^3) \text{ ?}} + \frac{h^3}{6} y'''(t_n) + \cdots$$

Keep more terms in the series, so error is even higher order (smaller)?

Keeping more terms

Again, Taylor series is

$$y(t_{n+1}) = y(t_n) + h \cdot y'(t_n) + \frac{h^2}{2} y''(t_n) + O(h^3)$$

Problem: **we don't know $\boldsymbol{y''}$**! (It may be hard/costly to find exactly.)
Our ODE only gives us the 1st derivative, $y'(t) = f(t, y)$.

Solution: Use another forward (finite) difference to *approximate* y''.

$$y''(t) = \frac{y'(t_{n+1}) - y'(t_n)}{h} + O(h)$$

Let's try doing this….

Trapezoidal Rule ("Crank-Nicolson")

In the end, we have:

$$\underbrace{y(t_{n+1}) = y(t_n) + \frac{h}{2}\Big(f\big(t_{n+1}, y(t_{n+1})\big) + f\big(t_n, y(t_n)\big)\Big)}_{\text{Trapezoidal Rule}} + \underbrace{O(h^3)}_{\text{Error Term}}$$

Therefore the *local truncation error* for trapezoidal rule is $O(h^3)$.

Reducing step size h now reduces per-step error cubically!

Trapezoidal Rule – Intuition?

In the end, we have:

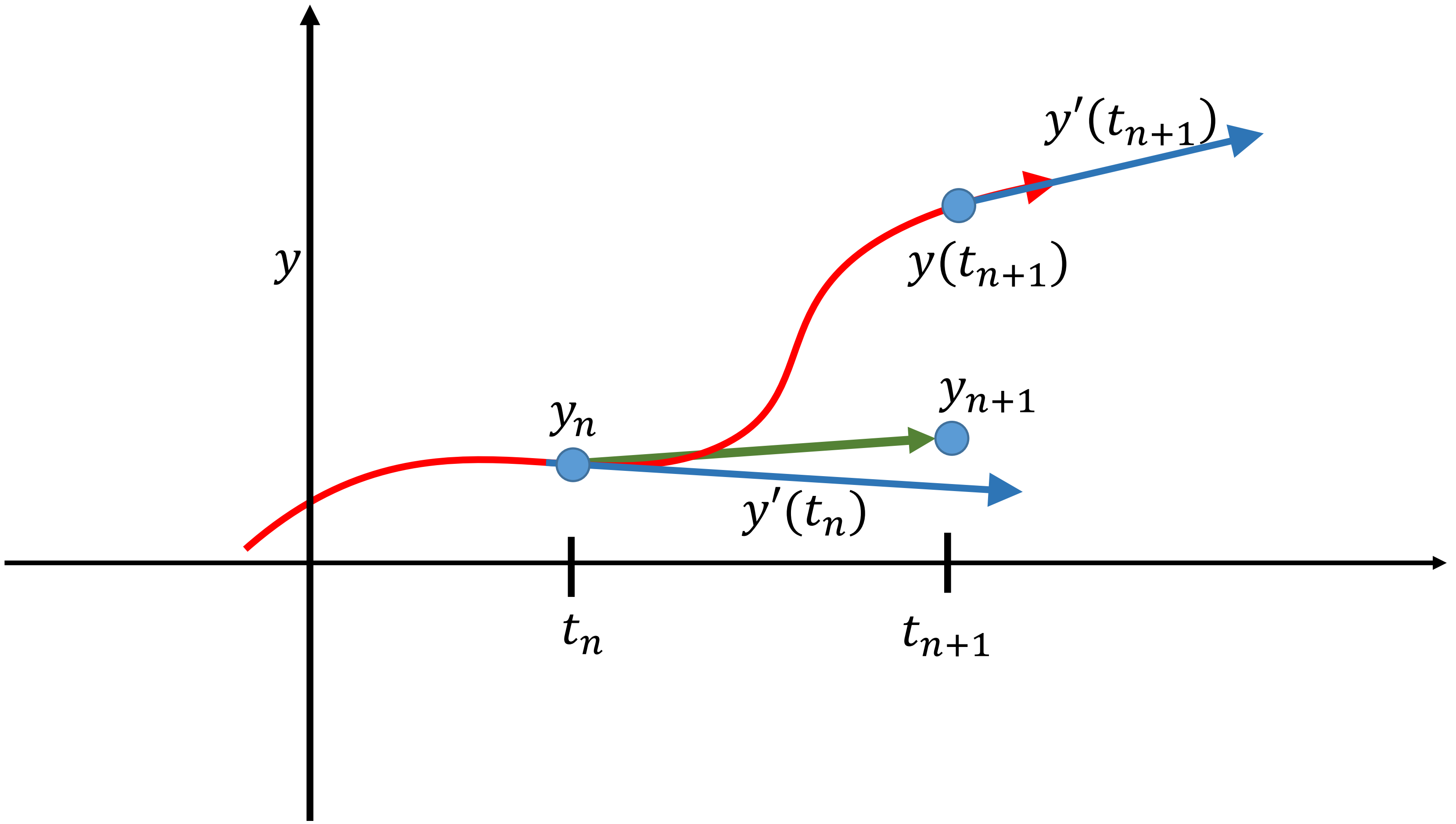
$$\underbrace{y(t_{n+1}) = y(t_n) + \frac{h}{2}\Big(f\big(t_{n+1}, y(t_{n+1})\big) + f\big(t_n, y(t_n)\big)\Big)}_{\text{Trapezoidal Rule}} + \underbrace{O(h^3)}_{\text{Error Term}}$$

Q: What is the ***geometric*** intuition for this scheme?

A: Evaluate the slope $y' = f$ at the start ***and*** end of the time step, and step forward using the ***average slope***.

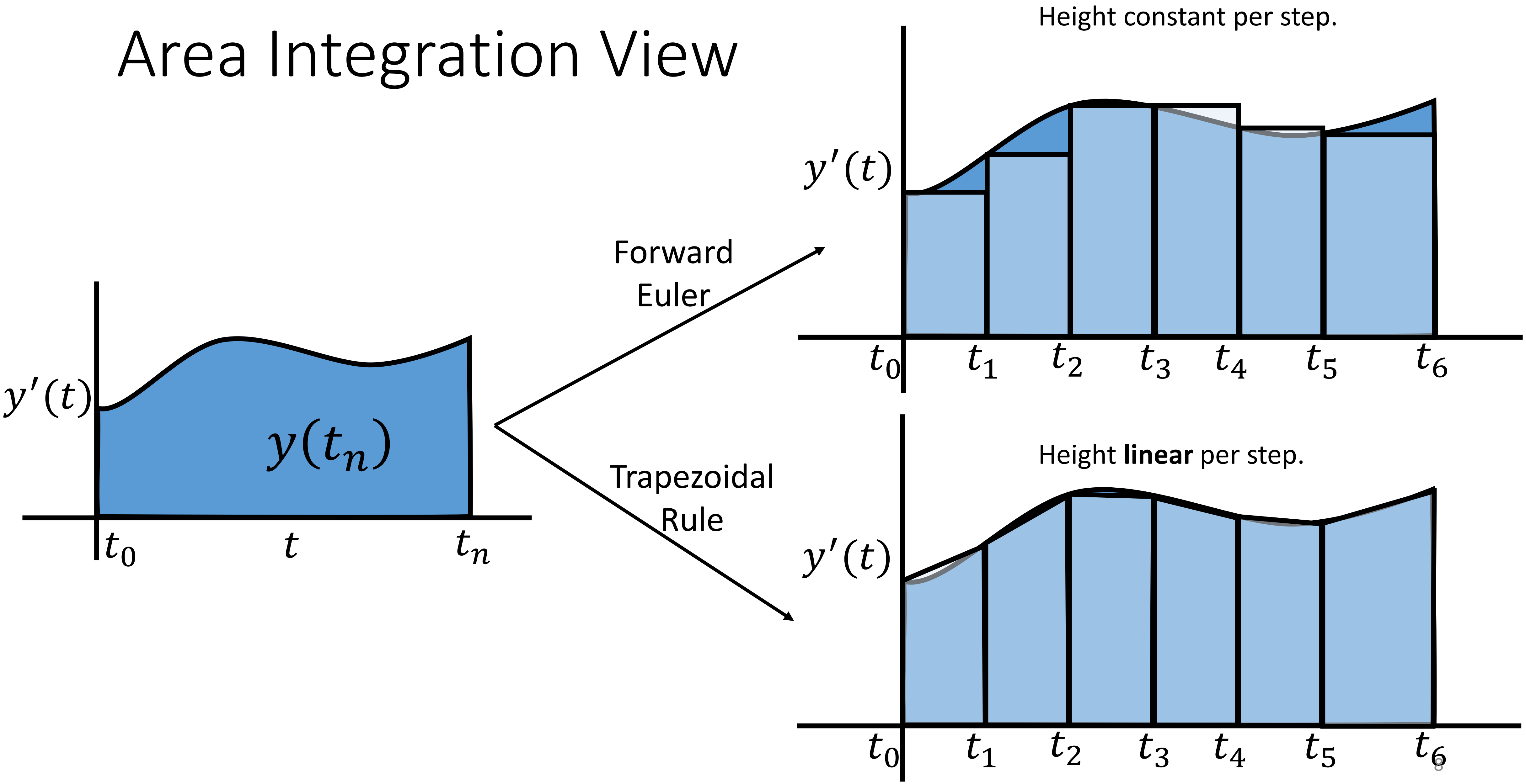
Trapezoidal Rule - Illustrated

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



Evaluate the slope $y' = f$ at the start ***and*** end of the time step, and step along the ***average slope***.

Area Integration View



Explicit v.s. Implicit Schemes

Forward Euler is an ***explicit scheme***.

$$y_{n+1} = y_n + hf(t_n, y_n)$$

The right-hand-side involves only known, time t_n quantities. Plug in and directly evaluate!

Trapezoidal is an ***implicit scheme.***

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

The right-hand-side involves some quantities from the (***currently unknown***) time t_{n+1}?!

We just need to solve the implicit equation for y_{n+1}.

Explicit v.s. Implicit Schemes - Tradeoffs

Explicit

- Simpler, and fast to compute ***per step.***
- Less stable – require smaller timesteps to avoid “blowing up”. (More later.)

Implicit

- Often more complex and expensive to solve **per step**.
- More stable – can safely take somewhat larger steps.

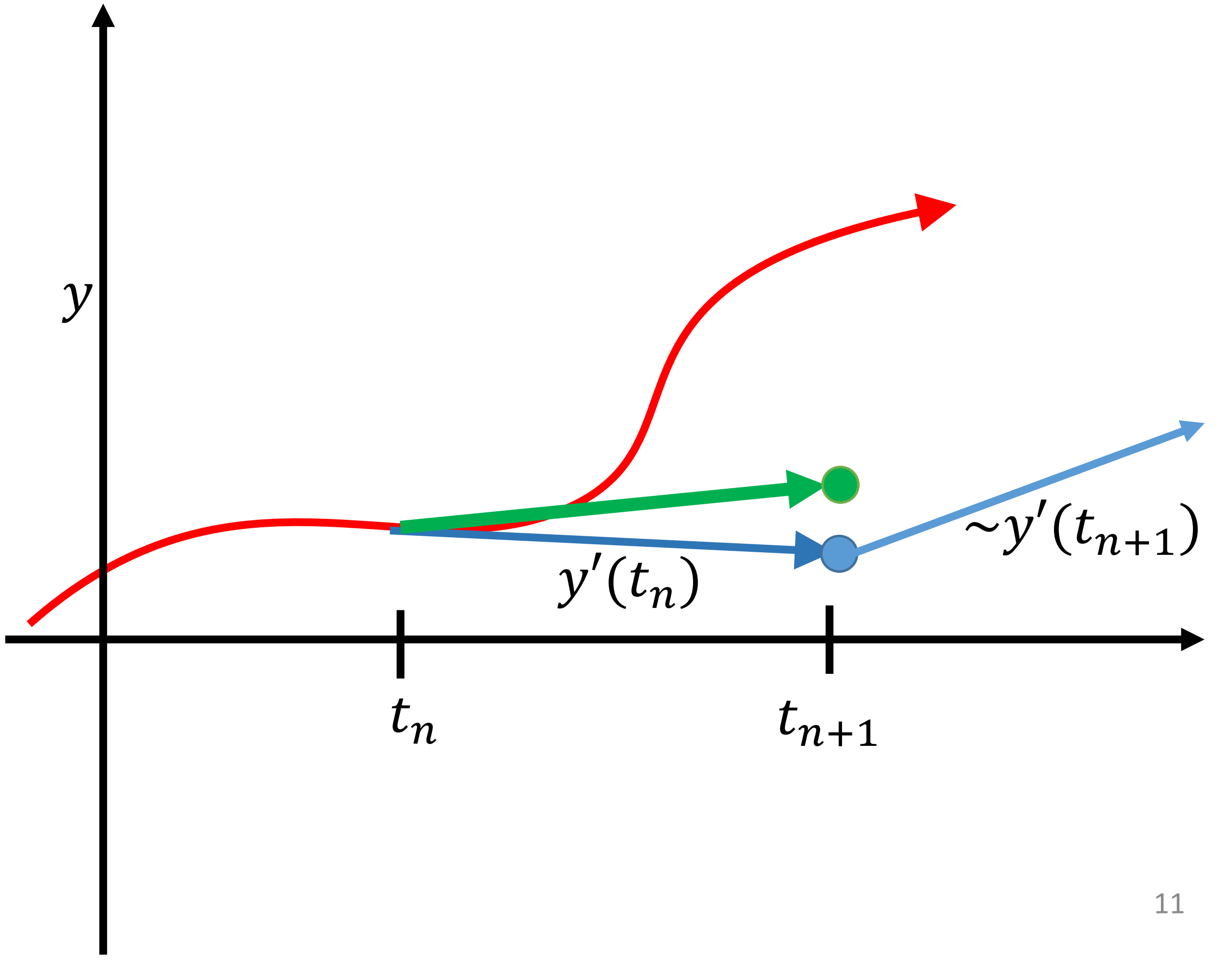
Which is more **computationally efficient**?

Problem-dependent – Tradeoff between solving the implicit equations for one large step v.s., versus cheaply computing many small steps.

Making Trapezoidal Explicit?

Let's derive another scheme that avoids the "implicitness" of trapezoidal.

1. Take a forward Euler step to ***estimate*** the unknown end point.
2. Evaluate slope there instead.
3. Use this ***approximate*** end-of-step slope in the trapezoidal formula.



Modified / Improved Euler

This is *exactly* the idea behind the modified/improved Euler method!

This looks like:

1. $y_{n+1}^{*} = y_n + hf(t_n, y_n)$
2. $y_{n+1} = y_n + \frac{h}{2}\left(f(t_n, y_n) + f(t_{n+1}, y_{n+1}^{*})\right)$

Tentative Forward Euler step

Trapezoidal-***like*** step, using approximate value y_{n+1}^{*}.

Improved Euler is an *explicit* scheme like F.E.:

No (potentially nonlinear, systems of) equations to solve.

Error of Improved Euler

Like trapezoidal, improved Euler has local truncation error (LTE) $O(h^3)$.

1. Let's see this derivation.

2. Then we'll visualize how forward and improved Euler methods behave (in Matlab).

3. Finally, try applying it to the particle example from last time.