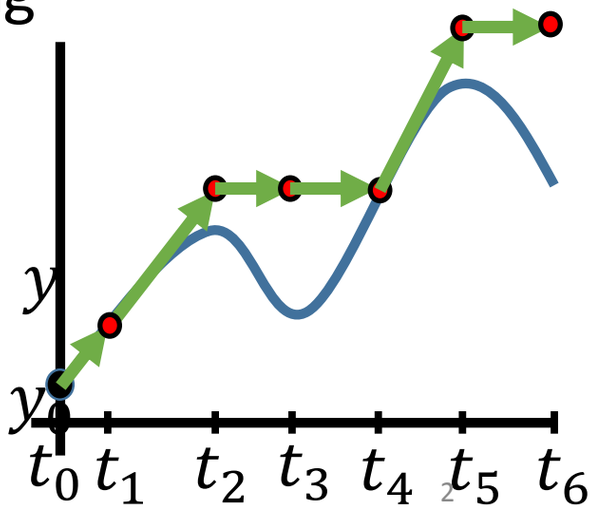
Last Time – ODEs and Forward Euler

Considered 1st order ordinary differential equations (ODE) in the form

$$y'(t) = f(t, y(t)), \text{ with } y(t_0) = y_0.$$

We applied **forward Euler** to approximate the value of y over time, stepping from one discrete time instant to the next using

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



Notation Reminder

Our discrete data is index by integers, i: t_i, y_i, x_i etc.

- $y(t_n)$ is the *exact* value of the true function y at time t_n.
- y_n is the approximate/discrete data at step n, i.e. time t_n, so $y_n \approx y(t_n)$.

So for example:

- $f(t_n, y_n)$ means the f evaluated on the numerical data, y_n.
- $f(t_n, y(t_n))$ means f evaluated on the exact value of y at time t_n.

Form of Ordinary Differential Equations

Note: The ordinary differential equation may not always be given in the desired form

$$y'(t) = f\big(t, y(t)\big).$$

e.g. $\frac{y'(t)-7t}{3t^2y(t)} = 1$ is still a valid ODE. It can be easily rearranged into the form above by isolating y', as in $y'(t) = \underbrace{3t^2y(t) + 7t}.$

Dynamics function, $f(t, y(t))$.

Forward Euler – Example #1

Consider the simple IVP $y'(t) = 2y(t)$, with initial conditions at $t_0 = 1$ of $y(t_0) = 3$.

a) Write down the Forward Euler recurrence.

b) Use forward Euler to approximate y at time $t = 4$ with $h = 1$ **and** $h = 1/2$.

c) Compare against the true (closed-form) solution, $y(t) = 3e^{2(t-t_0)}$.

See sample Matlab code on Piazza; try adjusting h.

Recall: *Systems* of Differential Equations

A model may have *multiple* variables of interest.

E.g. *x* and *y* coordinates of a moving object over time.

This gives a *system* of differential equations, such as

$$x'(t) = f_x\big(t, x(t), y(t)\big), \text{with } x(t_0) = x_0,$$
$$y'(t) = f_y\big(t, x(t), y(t)\big), \text{with } y(t_0) = y_0.$$

Two dynamics functions. *Two* initial conditions.

Forward Euler - Systems of Equations

For *systems* of 1st order ODEs, we apply Forward Euler to each row in exactly the same way.

$$\begin{bmatrix} y'(t) \\ z'(t) \end{bmatrix} = \underbrace{\begin{bmatrix} z(t) \\ tz(t) - ay(t) + \sin(t) \end{bmatrix}}_{\text{(Vector) Dynamics Function, } \boldsymbol{f}} \text{ with } \underbrace{\begin{bmatrix} y(0) \\ z(0) \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}}_{\text{Initial Conditions}}$$

Applying Forward Euler gives:

$$\begin{bmatrix} y_{n+1} \\ z_{n+1} \end{bmatrix} = \begin{bmatrix} y_n \\ z_n \end{bmatrix} + h \begin{bmatrix} z_n \\ t_n z_n - a y_n + \sin(t_n) \end{bmatrix} \text{ with } \begin{bmatrix} y_0 \\ z_0 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

Forward Euler – Example #2

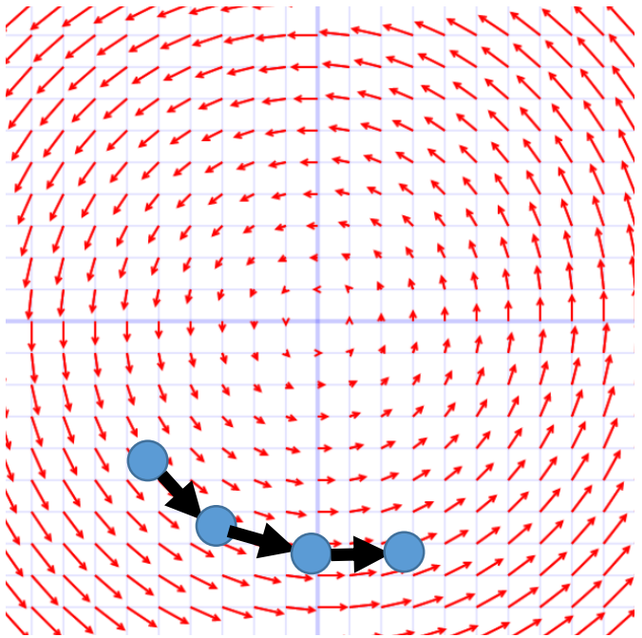
Consider a particle with coordinates $(x(t), y(t))$ satisfying the ODE system

$$x'(t) = -y(t)$$
$$y'(t) = x(t)$$

with $x(t_0) = 2, y(t_0) = 0,$ and $t_0 = 0.$

a) Write down the recurrence for Forward Euler applied to this problem, with time step $h = 2.$

b) Apply forward Euler up to $t = 6.$



A particle moving in a rotational wind field.

Further Examples

The course notes contain further examples of forward Euler in action.

Example 4.5: $y'(t) = y(t)\left(2.5t - t^2\sqrt{y(t)}\right)$

Forward Euler becomes:

$$y_{n+1} = y_n + hy_n(2.5t_n - t_n^2\sqrt{y_n}).$$

Example 4.6: A 4-parameter (a, b, α, β) predator-prey population model:

$$x'(t) = x(t)\left(a - \alpha y(t)\right)$$
$$y'(t) = y(t)(-b + \beta x(t))$$

Deriving Forward Euler

Forward Euler is a fairly intuitive method: compute the slope, and follow it to the next point in time.

Can we derive it a bit more carefully?

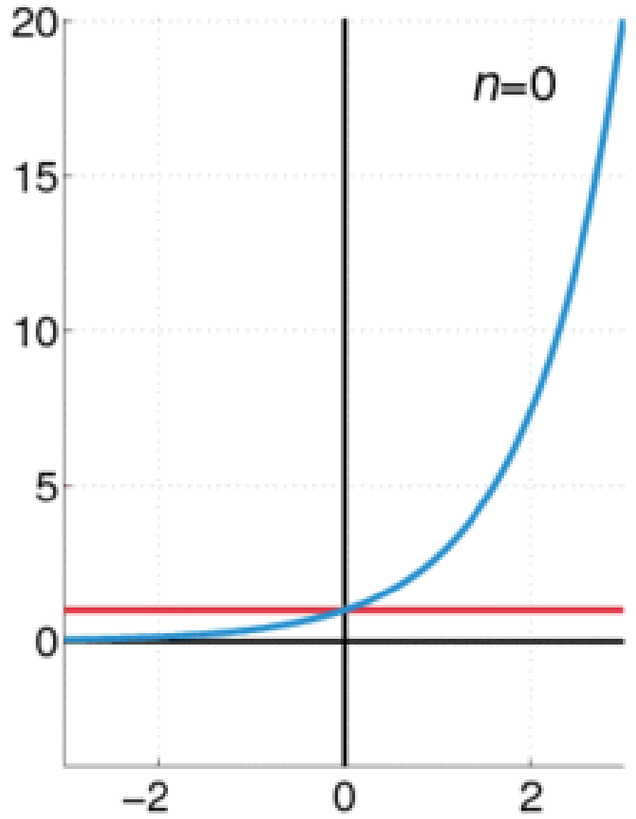
We'll need a key tool from calculus, today and going forward…

Reminder: Taylor series

Our key tool in this unit is the Taylor series/expansion.

A Taylor series lets us approximate a function in some neighbourhood using a weighted sum of its derivatives.

More terms in the series generally give better estimates.



Higher degree Taylor expansions do a better job approximating the true function in blue.

Reminder: Taylor series

Standard form:

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \ldots$$

If we apply to $y(t_{n+1})$, expanding around $y(t_n)$, since $h = t_{n+1} - t_n$, we get:

$$y(t_{n+1}) = y(t_n + h) = y(t_n) + h \cdot y'(t_n) + \frac{h^2}{2} y''(t_n) + \frac{h^3}{6} y'''(t_n) + \cdots$$

Deriving Forward Euler

Forward Euler is a fairly intuitive method: compute the slope, and follow it to the next point in time.

Can we derive it a bit more carefully?

1) “Finite difference” view.
2) Taylor series view.

Error of Timestepping Schemes

To be useful, we need to know something about the *error* we incur with a given time-stepping scheme (e.g., forward Euler).

Absolute/global error at some step n is : $|y_n - y(t_n)|$

Discrete estimate from timestepping

Exact value from (unknown) closed-form solution.

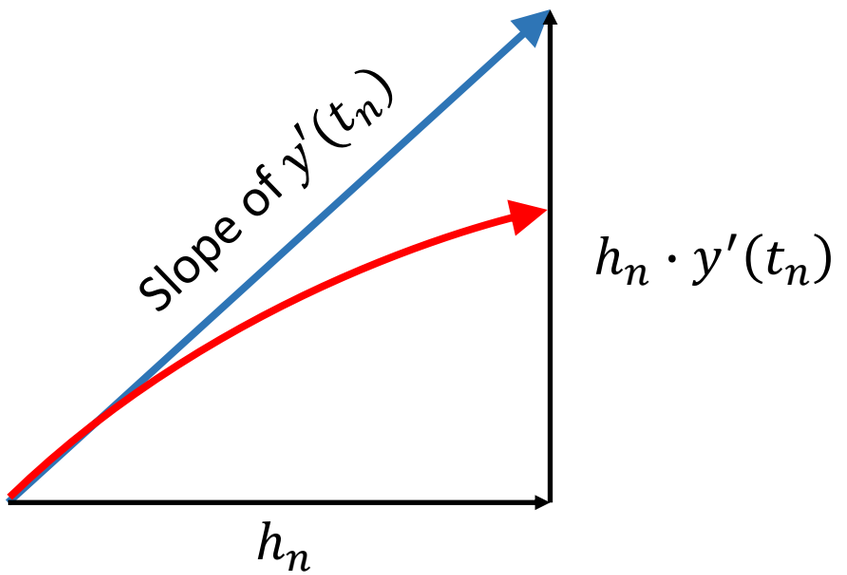
But we can't measure error ***exactly*** without knowing $y(t)$!

Understanding forward Euler error

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

Smaller step size $h \rightarrow$ more frequent estimates of the slope $\rightarrow$ less error in approximate solution!

Let's try to understand this error.



Recall: Taylor expansions

A Taylor series expansions of function y *near* $y(t_n)$ is

$$y(t_{n+1}) = y(t_n + h) = y(t_n) + h \cdot y'(t_n) + \frac{h^2}{2} y''(t_n) + \frac{h^3}{6} y'''(t_n) + \cdots$$

To derive Forward Euler we used a *first order* Taylor expansion:

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

Exact quantities!

This led to forward Euler looking like:

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

Approximate/discrete quantities!

Finite Differences via Taylor expansion

We also used a (forward) finite difference to derive FE:

$$y'(t_n) \approx \frac{y_{n+1} - y_n}{h}$$

This can be derived from Taylor expansion too!

$$y(t_{n+1}) = y(t_n) + hy'(t_n) + O(h^2)$$

Rearranging gives:

$$y'(t_n) = \frac{y(t_{n+1}) - y(t_n)}{h} + O(h)$$

So the error in this derivative approximation is $O(h)$.

Aside: Multivariable Taylor expansions

If we have a (dynamics) function of multiple variables, we can likewise Taylor expand in the same way for each variable…

$$f\left(x + h_x, y + h_y\right) = f(x, y) + h_x \frac{\partial f}{\partial x} + h_y \frac{\partial f}{\partial y} + O(h_x^2) + O(h_y^2)$$

Now, let's see how to derive the (truncation) error in Forward Euler…