- If we know $f'(x)$ everywhere, we can use some form of Riemann sum to estimate $f(x)$ as precisely as we like

- What if we don't know $f'(x)$?

$$\frac{df}{dx} = g(f, x)$$

The derivative of f depends on f, which we don't know!

$$y'(x) = \frac{dy}{dx} = x^2 \cdot y$$

$$y(x) = \int_0^x y'(x)\,dx$$

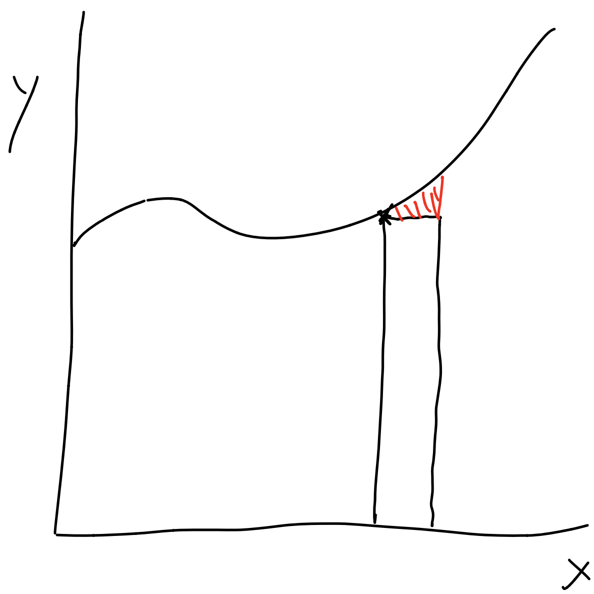
$$\approx \sum_i y'(x_i)\,\Delta x$$

But what is $y'(x_i)$?

$$y'(x_i) = x_i^2 \cdot y(x_i)$$

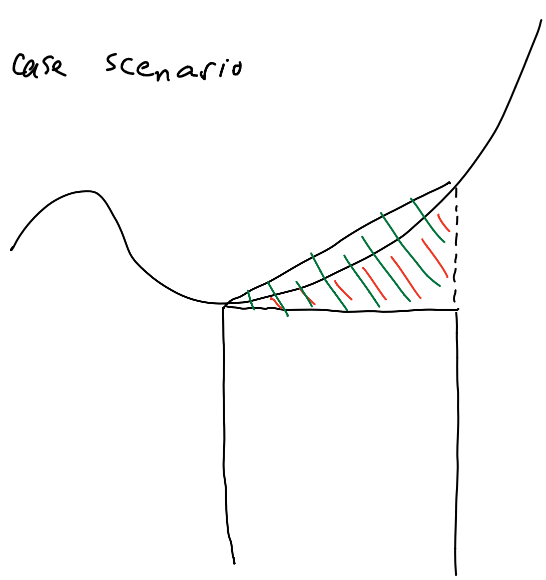
But we don't know $y(x_i)$!

Improving the approximation

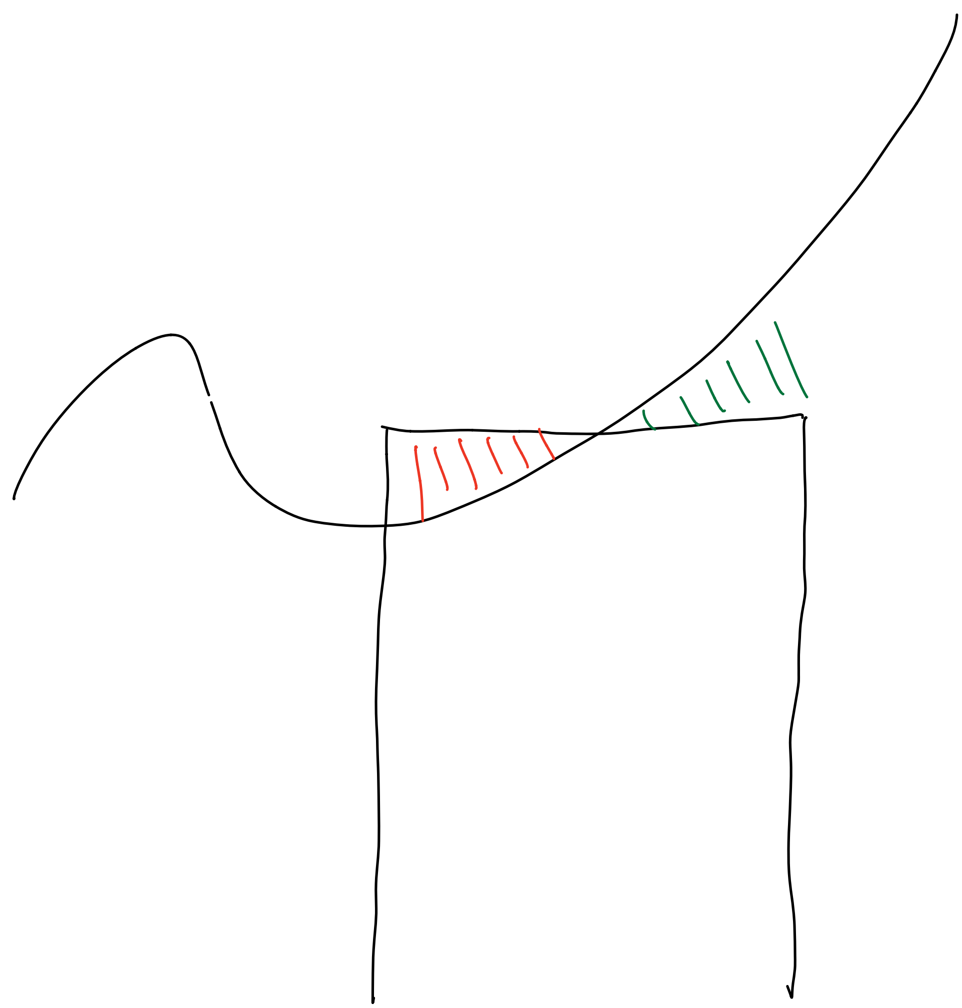


We get errors because the function deviates from our rectangle at the edge

- Worst case scenario



What if I shift each rectangle so that instead of the left edge touching the curve, the center does



$$A = y\left(x_i + \frac{\Delta x}{2}\right) \cdot \Delta x + y\left(x_i + \frac{3\Delta x}{2}\right) \cdot \Delta x + \cdots$$

$$= \sum_{k=0}^{n-1} y\left(x_i + \left(k + \tfrac{1}{2}\right)\Delta x\right) \Delta x$$

```
def integrate_midpoint(xi,xf,n):
    #First get estimate
    dx = (xf - xi) / n
    total = 0
    for i in range(n):
        x = xi + (i+1/2) * dx
        f = func(x)
        area = f * dx
        total += area
    return total
```

```python
def func(x):
    return x**2

def deriv(x):
    return 2*x


def integrate_midpoint(xi,xf,n):
    #First get estimate
    dx = (xf - xi) / n
    total = 0
    for i in range(n):
        x = xi + (i+1/2) * dx
        f = func(x)
        area = f * dx
        total += area
    return total



def integrate(xi,xf,n):
    #First get estimate
    dx = (xf - xi) / n
    total = 0
    for i in range(n):
        x = xi + i * dx
        f = func(x)
        area = f * dx
        total += area
```

One solution: an "on the fly" Riemann sum

Even if we don't know y + y' everywhere,
we usually y_0 + y'_0, our initial values

(starting position, starting speed)

This is enough info to do a left edge
Riemann sum!

Scenario: Don't know $y(x)$, know $y'(x) = f(x,y)$

$\left[y'(x) = x^2 \cdot y(x) \right]$

Need both x + $y(x)$

We know $y(x_i)$ + $y'(x_i)$

We want $y(x_f)$

Use rectangles with width Δx

$$y(x_i + \Delta x) = y(x_i) + \underbrace{\int_{x_i}^{x_i+\Delta x} y'(x)\,dx}$$

Approximate using
a rectangle with
$h = y'(x_i)$ +
width $= \Delta x$

$$y(x_i+\Delta x) \approx y(x_i) + y'(x_i)\Delta x$$

Now we know $y(x_i+\Delta x)$, how do we get $y(x_i+2\Delta x)$?

$$y(x_i+2\Delta x) = y(x_i+\Delta x) + \int_{x_i+\Delta x}^{x_i+2\Delta x} y'(x)\,dx$$

$$y(x_i+2\Delta x) \approx y(x_i+\Delta x) + y'(x_i+\Delta x)\Delta x$$

$$y'(x_i+\Delta x) = f(x_i+\Delta x, y(x_i+\Delta x))$$

$$y(x_i+3\Delta x) \approx y(x_i+2\Delta x) + y'(x_i+2\Delta x)\Delta x$$

→ Start with $y(x_i)$, $y'(x_i)$

- use $y'(x_i)$ to estimate $y(x_i+\Delta x)$
- use $y(x_i+\Delta x)$ to estimate $y'(x_i+\Delta x)$
- use $y'(x_i+\Delta x)$ to estimate $y(x_i+2\Delta x)$

...

$$y(x_n) \approx y(x_{n-1}) + y'(x_{n-1})\Delta x$$

$$x_n = x_i + n\Delta x$$

$$x_{n-1} = x_i + (n-1)\Delta x$$

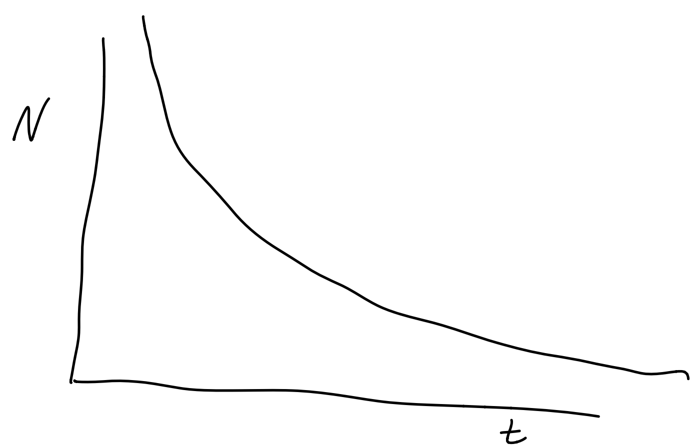
Example: Radioactive Decay

$N(t)$ = amount of material left [kg]

$$\frac{dN}{dt} = -\frac{1}{\tau} N(t)$$

- material decays faster when there is more of it

- As N decreases, $\frac{dN}{dt}$ also decreases

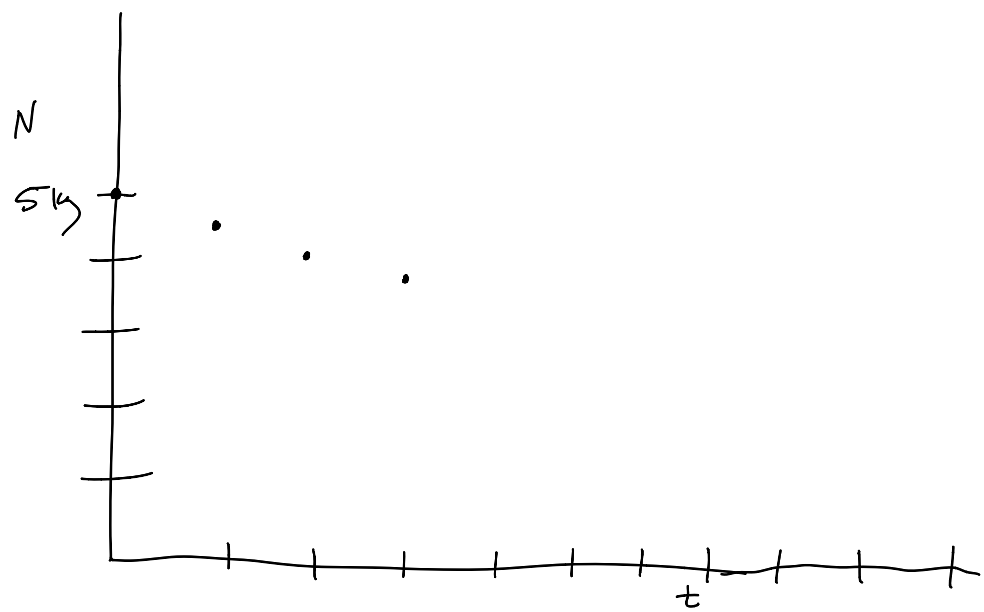


$N(t=0) = 5\ \mathrm{kg}$

$\tau = 100\ \mathrm{sec}$

$N'(t=0) = -\frac{1}{\tau} N(t=0) = \frac{-1}{20} \frac{\mathrm{kg}}{\mathrm{s}}$

$\Delta t = 10\ \mathrm{s}$



$N(0) = 5\ \mathrm{kg}$

$N(10\mathrm{s}) \approx N(0) + N'(5\mathrm{kg})(10\mathrm{s}) = 5\ \mathrm{kg} - \left(\frac{1}{20} \frac{\mathrm{kg}}{\mathrm{s}}\right)(10\mathrm{s})$

$= 5\ \mathrm{kg} - 0.5\ \mathrm{kg} = 4.5\ \mathrm{kg}$

$N(20\mathrm{s}) \approx N(10\mathrm{s}) + N'(4.5\mathrm{kg})(10\mathrm{s})$

$= 4.5\ \mathrm{kg} - \frac{4.5}{100} \frac{\mathrm{kg}}{\mathrm{s}} \cdot 10\mathrm{s}$

$= 4.05\ \mathrm{kg}$

$N(30s) \approx N(20s) + N'(4.05\,kg)(10\,s)$

$= 4.05\,kg - \frac{4.05}{100}\frac{kg}{s} \cdot 10\,s$

$= 3.645\,kg$

. . .

How to code

Need:

- function for derivative
- $x_i, x_f, \Delta t / n$

$y_n = y(x_i + n\Delta x)$

$y'_n = y'(x_i + n\Delta x)$

Know y_0, y'_0

$y_1 = y_0 + y'_0 \Delta x$

$y'_1 = y'(x_1, y_1)$

$y_2 = y_1 + y'_1 \Delta x$

$y_2 = y_0 + y'_0 \Delta x + y'_1 \Delta x$

$$y_3 = y_2 + y_2' \Delta x$$

$$y_3 = y_0 + y_0' \Delta x + y_1' \Delta x + y_2' \Delta x$$

$y = y_0$

$yprime = yp0$

$x = x_0$

$y = y + yprime * \Delta x$

$x = x + \Delta x$

$yprime = deriv(x, y)$

$y = y + yprime * \Delta x$

$x = x + \Delta x$

$yprime = deriv(x, y)$

What are we doing?
-Varying x from x0 to xf in steps of delta x
-Calculating yprime of x
-Use add the rectangle area yprime*dx to y

We can do this in a loop!

```
import numpy as np

def deriv(N,t,tau):
    return -N / tau

tau = 100
t0 = 0
tf = 10
dt = 1

N = 5
for t in np.arange(t0,tf+dt,dt):
    Nprime = deriv(N,t,tau)
    N = N + Nprime * dt
print(N)
```