3. Numerical integration and differentiation

Motivation: Lets solve the differential equation

$$\dot{X} = -p(t)\,X \quad \rightsquigarrow \text{separation of variables}$$

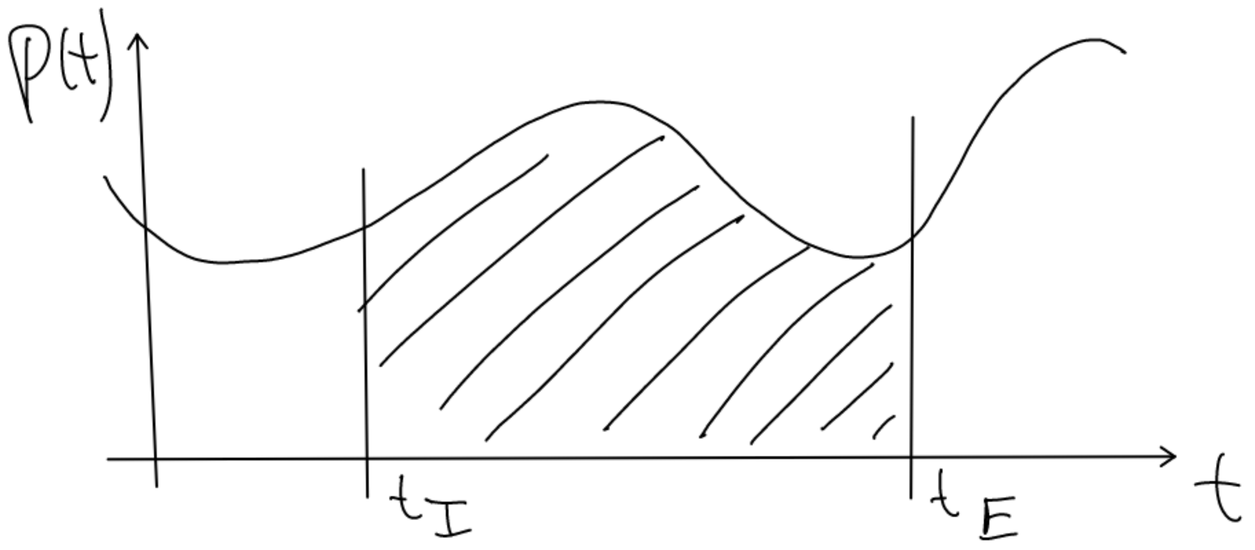
$$\ln\frac{X(t)}{X(t_0)} = \int_{X(t_0)}^{X(t)} \frac{dx}{x} = \int_{t_0}^{t} -p(t)\,dt \quad \rightsquigarrow \quad \boxed{X(t) = \exp\left[-\int_{t_0}^{t} p(t)\,dt\right] X(t_0)}$$

We already can compute the Product and exponential.

Today: Integral

3.1. Integration

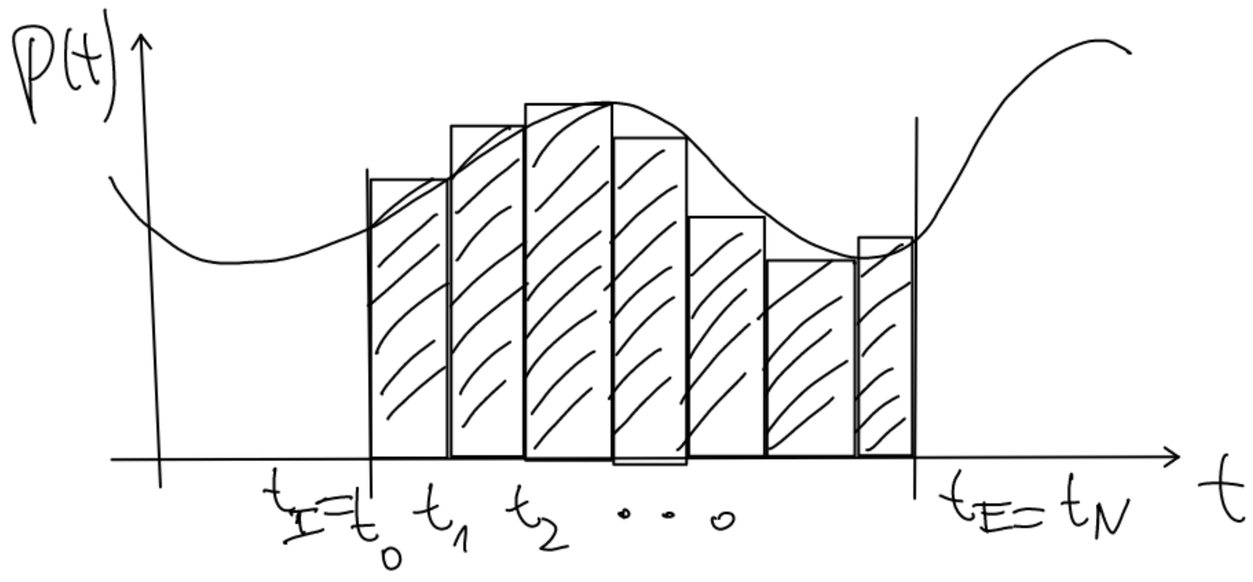
Find the area under the curve



Remember the definition:

$$\int_{t_I}^{t_E} dt\, p(t) = \lim_{\Delta t \to 0} \sum_{i=1}^{N} p(t_i)\,\Delta t$$

$$t_{i+1} - t_i = \Delta t$$



for example:

$$\int_a^b dt\,\sin(t) = -\cos(t)\Big|_a^b$$

$$= -\cos(b) + \cos(a)$$

$$\varepsilon = \left|1 + \frac{I}{\cos(b) - \cos(a)}\right|$$

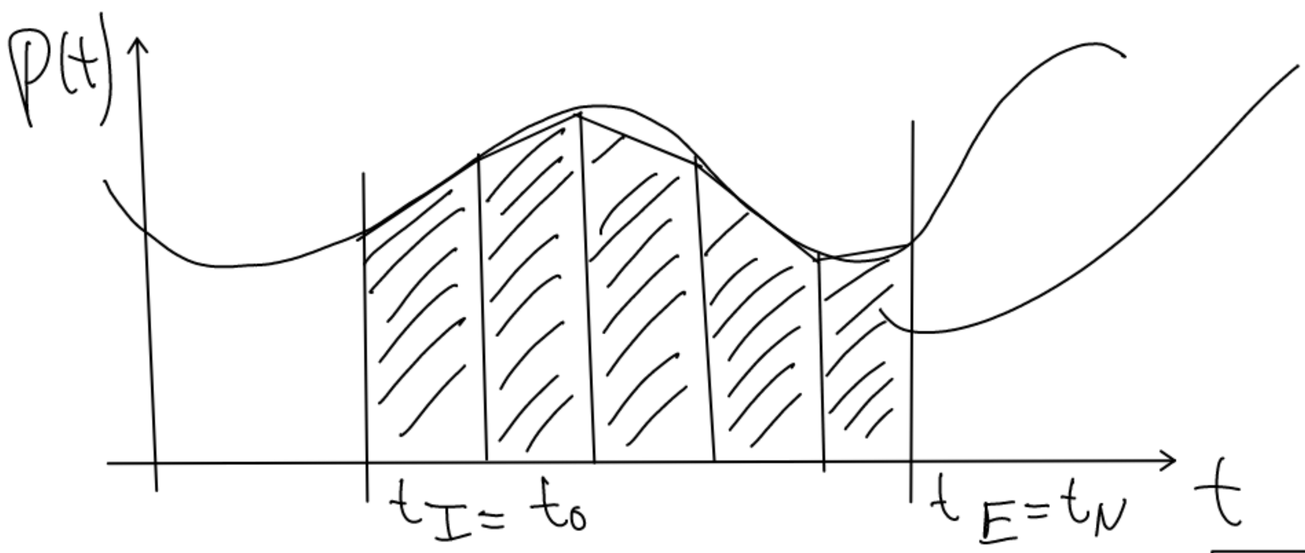
```
def intSimple(p, a, b, N):
    I = 0  # value of integral
    dt = (b-a)/N
    for j in range(N+1):
        t = a + dt*j
        I = I + p(t)*dt
    return I
```

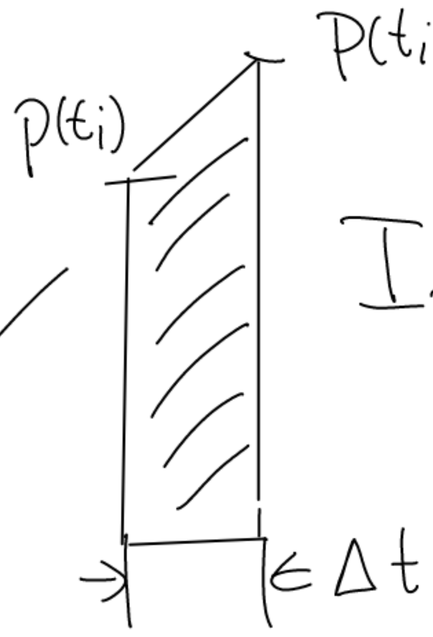
Problem: • for small error large N

$\rightsquigarrow$ many function calls $\rightsquigarrow$ very slow

Question: How to get smaller errors with smaller N?

Trapezoid method:





$$I_T = \sum_{i=0}^{N-1} \frac{p(t_i) + p(t_{i+1})}{2} \Delta t$$

Can be simplified to

$$I_T = \left[\frac{p(t_I) + p(t_E)}{2} + \sum_{i=1}^{N-1} p(t_i) \right] \Delta t$$

We learn: be careful at the boundaries.

Approximate better than linear

$\rightsquigarrow$ i.e. quadratic by parabolas

Simpson's method

$$I_S = \sum_{i=1}^{\frac{N-1}{2}} \int_{t_{2i-1}}^{t_{2i+1}} p_i(t)\, dt \quad \text{with parabola } g(x)$$

going through the 3 points

$(t_{k-1}, p(t_{k-1}))$, $(t_k, p(t_k))$, and $(t_{k+1}, p(t_{k+1}))$

$$p_k(t) = \frac{(t-t_k)(t-t_{k+1})}{(t_{k-1}-t_k)(t_{k-1}-t_{k+1})} f(t_{k-1}) + \frac{(t-t_{k-1})(t-t_{k+1})}{(t_k-t_{k-1})(t_k-t_{k+1})} f(t_k)$$
$$+ \frac{(t-t_{k-1})(t-t_k)}{(t_{k+1}-t_{k-1})(t_{k+1}-t_k)} f(t_{k+1})$$

↝ reorder in powers of t

$$P_k(t) = t^2 \cdot A + t \cdot B + C$$

$$\int_{t_{k-1}}^{t_{k+1}} P_k(t)\,dt = \frac{1}{3}A\left(t_{k+1}^3 - t_{k-1}^3\right) + \frac{1}{2}B\left(t_{k+1}^2 - t_{k-1}^2\right) + C\left(t_{k+1} - t_{k-1}\right)$$

and after some algebra

$$I_{Si} = \frac{\Delta t}{3}\sum_{k=\text{odd}}^{N-1}\left[p(t_{k-1}) + 4p(t_k) + p(t_{k+1})\right]$$

↳ The algorithm described here needs an even number N. If we have odd N the relation has to be adapted for the last slice

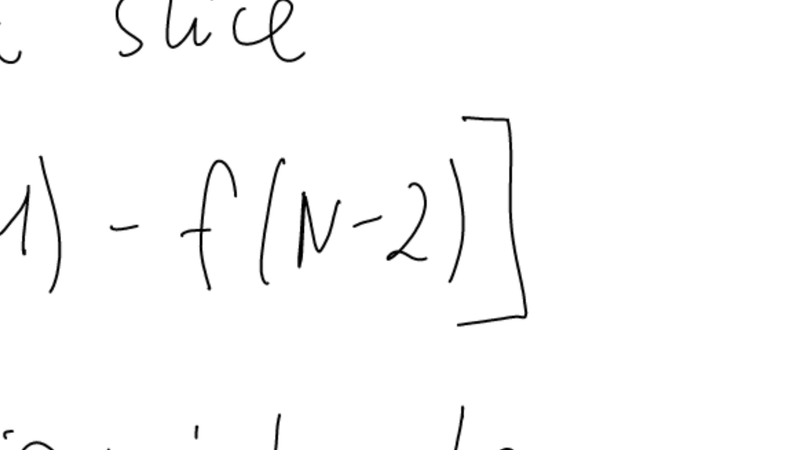
$$I_{last} = \frac{\Delta t}{12}\left[5p(N) + 8f(N-1) - f(N-2)\right]$$

Other methods, too. Check i.e. SciPy.integrate

3.2. Differentiation

Again from the definition

$$\dot{p}(t) = \lim_{\Delta t \to 0} \frac{p(t+\Delta t) - p(t)}{\Delta t}$$



which we can just evaluate for finit Δt.

↳ Prone to numerical errors for too small Δt !

Look at Taylor series expansion:

$$p(t_i+\Delta t) = p_{i+1} = p_i + \Delta t\,\dot{p}_i + \frac{\Delta t^2}{2}\ddot{p}_i + \dots$$

$$p_{i-1} = p_i - \Delta t\,\dot{p}_i + \frac{\Delta t^2}{2}\ddot{p}_i + \dots$$

$$\frac{p_{i+1}-p_i}{\Delta t} = \dot{p}_i + \frac{\Delta t}{2}\ddot{p}_i$$

error grows linearly with Δt

$$\frac{p_{i+1}-p_{i-1}}{2\Delta t} = \dot{p}_i + \frac{\Delta t^2}{6}p_i^{(3)}$$

now quadratically much better for $\Delta t \ll 1$

with similar computational cost.

This can be pushed to higher orders, i.e.

@ 4th order the "five-point" derivative formula

$$\dot{p}_i = \frac{1}{12\Delta t}\left(p_{i-2} - 8p_{i-1} + 8p_{i+1} - p_{i+2}\right)$$

Remarks:

- higher order ⇒ smaller error but more comp. effort to obtain function values
- Problematic at the boundary of function
- similar eqs. can be derived for higher derivatives, like

$$\ddot{p}_i = \frac{p_{i-1} - 2p_i + p_{i+1}}{\Delta t^2} + \mathcal{O}(\Delta t^3)$$

in scipy from scipy.misc import derivative