Python Seminar 2018

Ana Bulović
Jorin Diemer
Jens Hahn

Plan for today

1. ODE
2. Numerical integration
3. numpy.odeint

Ordinary differential equations

$$\frac{\mathrm{d}\vec{x}}{\mathrm{d}t} = \vec{f}(\vec{x}, t)$$

- **Description of the change of a species**
- **Integration needed**

Form an ODE

$$\frac{d\vec{x}}{dt} = \sum \mathrm{Rates}_{\mathrm{production}} - \sum \mathrm{Rates}_{\mathrm{loss}}$$

- **ODE is a sum of producing and consuming terms**
- **e.g. Michaelis-Menten term for enzyme reactions**

ODE properties

- **Deterministic & continuous description**
- **Only valid for large quantities in biology**
- **Analytic solutions? (TdV, VdK…)**

Numerical solution

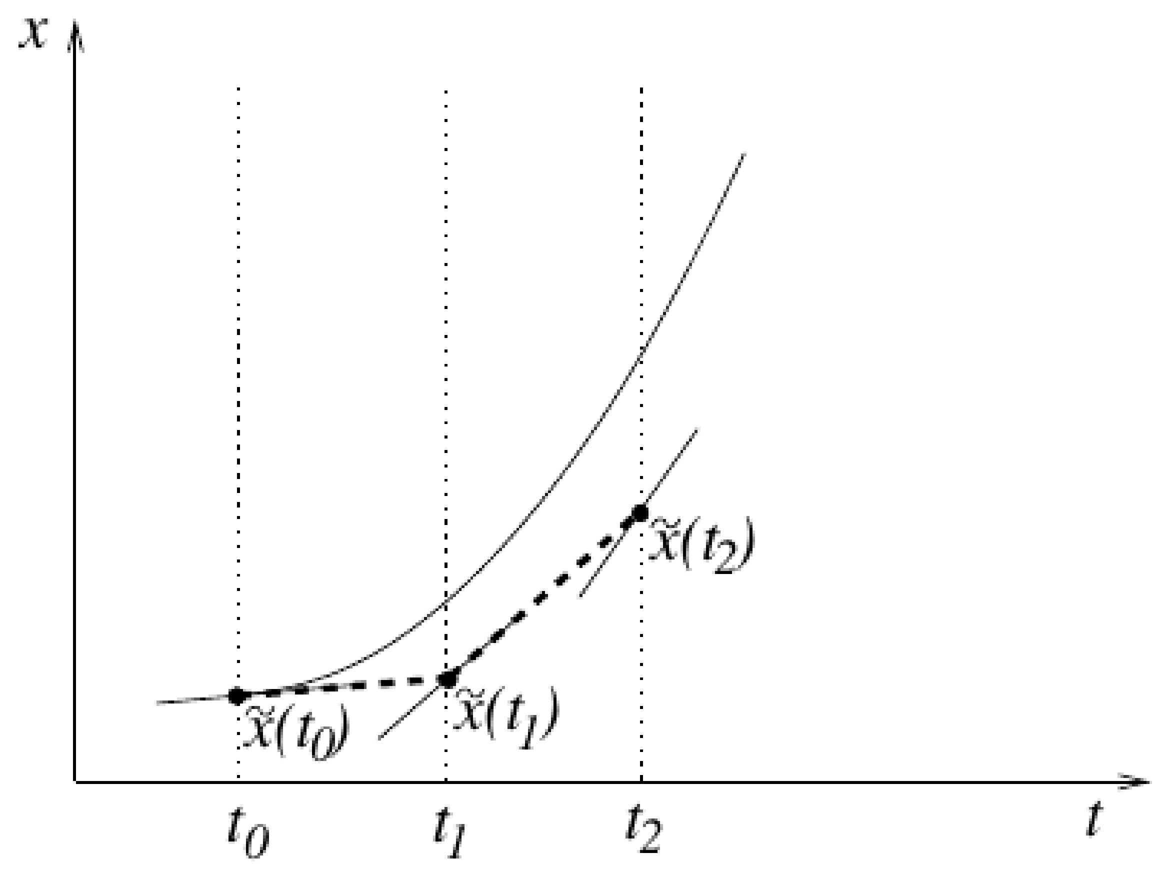
Discretisation:

$$\Delta\vec{x} = \vec{f}(\vec{x}, t) \times \Delta t$$

$$\vec{x}_{i+1} - \vec{x}_i = \vec{f}(\vec{x}_i, t) \times \Delta t$$

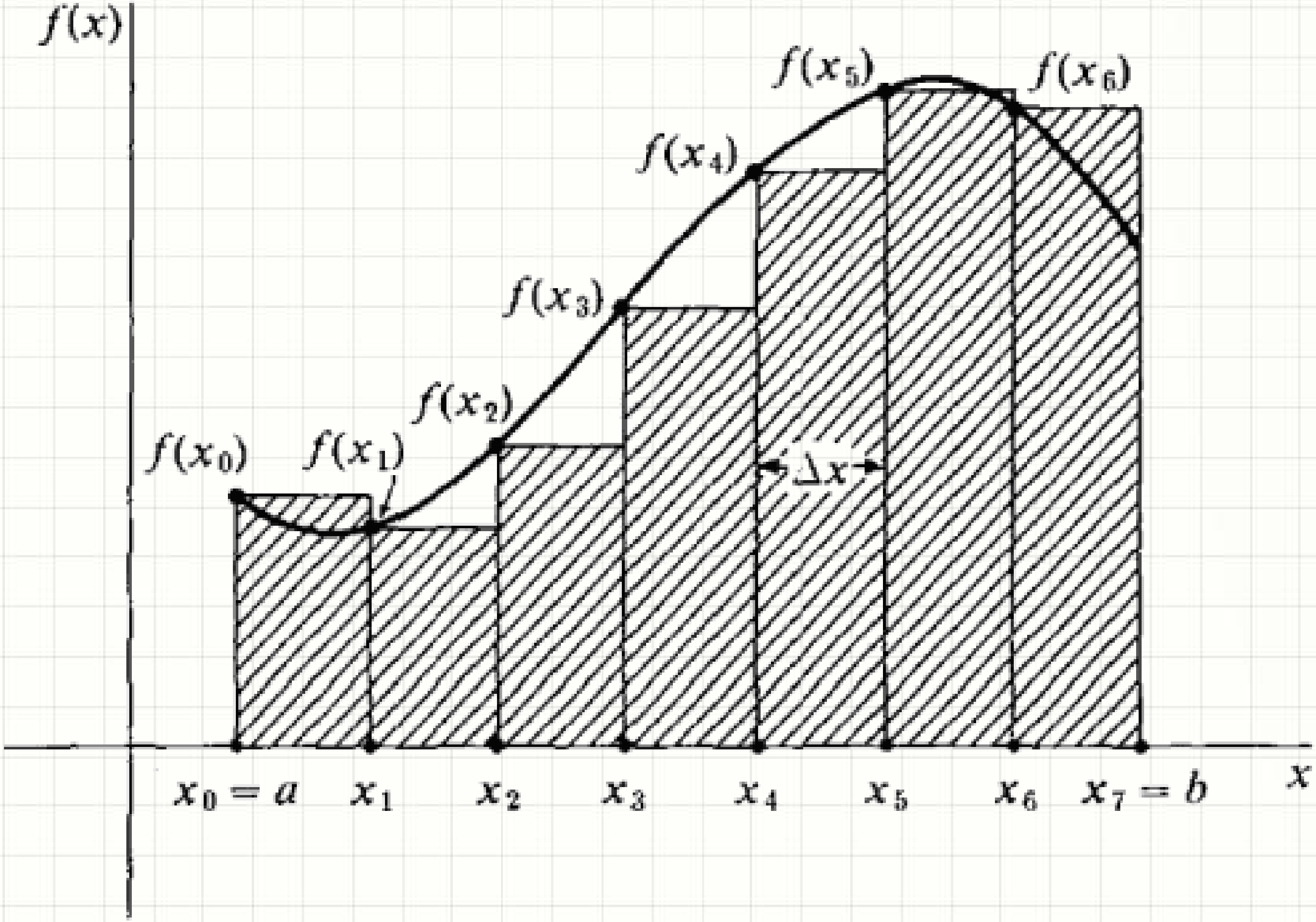
Euler‘s method

$$\Phi(t, x, h) = x + h \cdot f(t, x)$$



Euler‘s method

$$\Phi(t, x, h) = x + h \cdot f(t, x)$$



ODE 101

- **Time grid**

- **Method to identify next point x(i+1)**

- **<u>Methods</u>:**
  - **Euler’s method (explicit, single step)**
  - **Heun’s method (explicit, single step)**
  - **Runge-Kutta methods (explicit, single step)**
  - **Adams-Bashfort (explicit, linear multistep)**
  - **Adams-Moulton (implicit, linear multistep)**
  - **Backwards differentiation formulas (implicit, linear multistep)**

Important notes

- **Adaptive step size control**
- **Stiff systems (implicit methods)**

Solve an ODE numerically

1. Time grid

2. Initial conditions

```
X(0) = 42
Y(0) = 14
```

3. Derivative function

```
def f(x,t):
        …
        return dxdt
```