

numpy as np
matplotlib.pyplot as plt

$t = \text{np.arange}(\text{start}=0, \text{stop}=4 * \text{np.pi}, \text{step}=0.1)$

$\text{linspace}(\text{start}, \text{stop}, \text{num}=300)$

$x = \text{np.sqrt}(t)$

$y = t^{**}(1/3)$

$$K=3$$

$$2K=6$$

$$K^2=9$$

$$\frac{K(K-1)}{2}$$

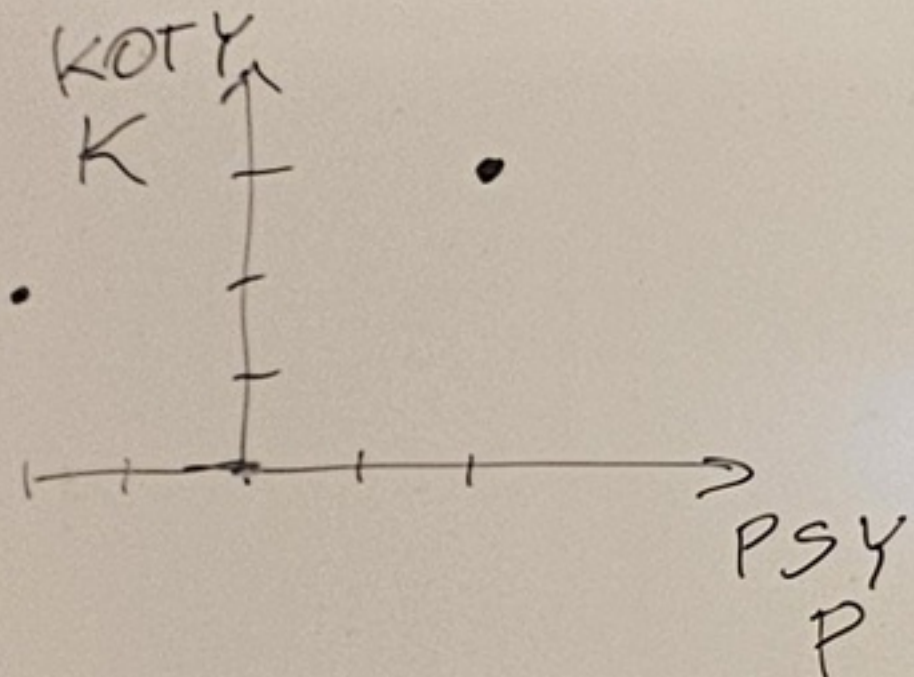
$\text{fig}, \text{ax} = \text{plt.subplots}(\text{nrows}=1, \text{ncols}=1, \text{subplot_kw} = \{ \text{'projection': '3d'} \})$

$\text{ax.plot}(t, x, y)$

$\text{ax.set_xlabel}(\text{'t'})$
y label
z label

$\text{plt.show}()$

, fontsize=12



$$\begin{bmatrix} 2 \\ 3 \end{bmatrix} \quad (2, 3)$$

$$a: \quad \underbrace{2P + 3K}$$

$$b: \quad -P + K$$

$$a + b = 2P + 3K + (-P) + K = P + 4K$$

$$a \cdot b = (2P + 3K)(-P + K) = -2P^2 + 2PK - 3KP + 3K^2$$

$\{KP = PK\}$

$$= -2P^2 - PK + 3K^2$$

$$\{P = 1\} = -2 - K + 3K^2$$

$$K = \sqrt{-1} = j$$

$$\{K^2 = -1P = -1\} = -5 - K$$

range

$$\text{np.arange}(0, 10^{+1}, \text{step}=1)$$

$$\text{np.linspace}(0, 10, \text{num}=11)$$

↓ w domu

własna implementacja

def linspace(.....):

:

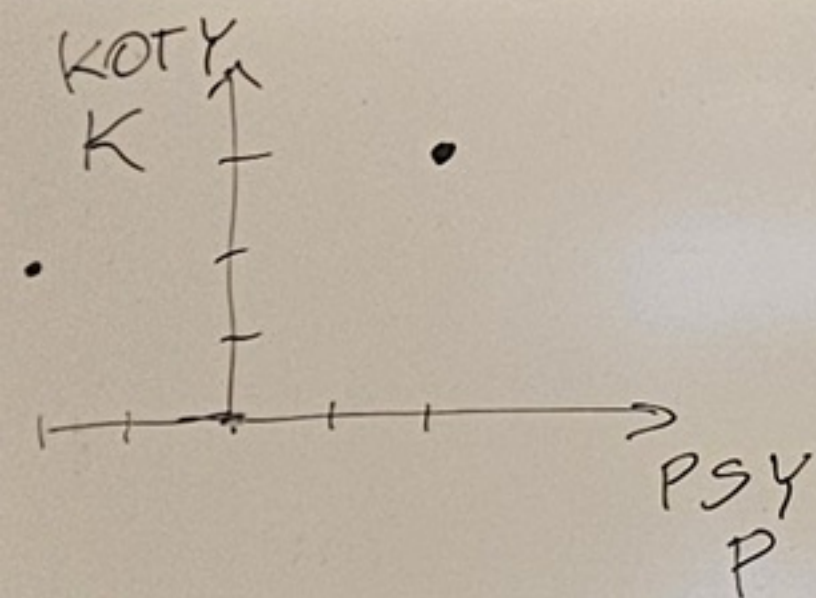
w domu

import math

czego nie ma w numpy,
szukaj w math

K_1	•	•	
K_2	•	•	
	P_1	P_2	P_3

K_1	•	•
K_2	•	•
	K_1	K_2



0 1 1 0

$$\begin{bmatrix} 2 \\ 3 \end{bmatrix} \quad (2, 3)$$

$$a: \underbrace{2P + 3K}$$

$$b: -P + K$$

$$a + b = 2P + 3K + (-P) + K = P + 4K$$

$$a \cdot b = (2P + 3K)(-P + K) = -2P^2 + 2PK - 3KP + 3K^2$$

$\{KP = PK\}$

$$= -2P^2 - PK + 3K^2$$

$$\{P = 1\} = -2 - K + 3K^2$$

$$\{K^2 = -1P = -1\} = -5 - K$$

$$(1 + 4j)(-2 - 2j) = 6 - 10j$$

$$\boxed{j^2 = -1}$$

$$(a + bj) \cdot (c + dj) = (ac - bd) + (ad + bc) \cdot j$$

cz. rzeczyw.

cz. urojona

$$a = 1 + 4j$$

$$b = -2 - 2j$$

$$a * b$$

np. real(a)

np. imag(b)

a. real

b. imag

$$\begin{bmatrix} a \\ b \end{bmatrix} \cdot \begin{bmatrix} c \\ d \end{bmatrix} =$$

$$= \begin{bmatrix} ac - bd \\ ad + bc \end{bmatrix}$$

$t = \text{np.arange}.$

$f_zesp = \text{np.exp}(-1j * t)$

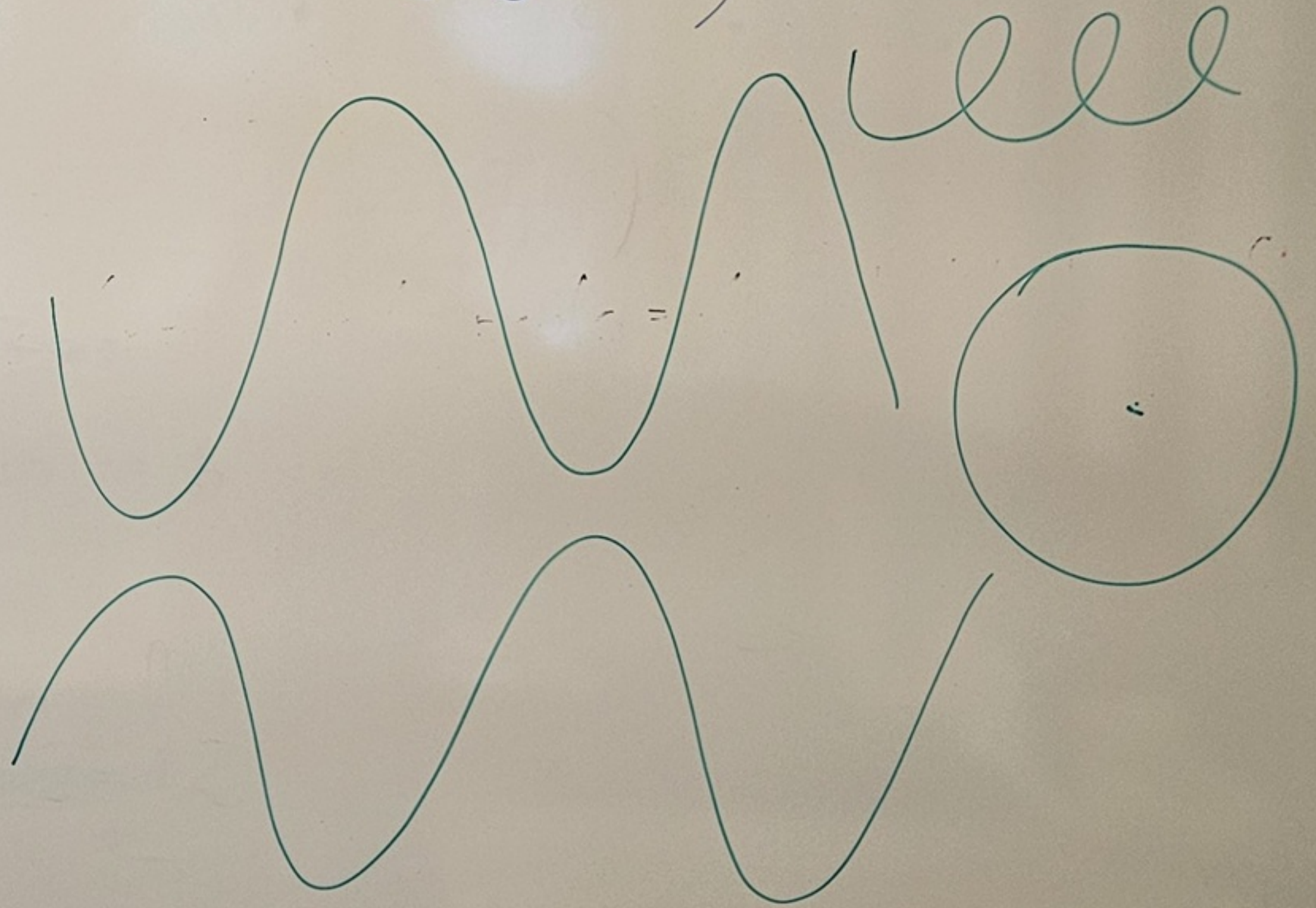
$x = \text{np.real}(f_zesp)$

$y = \text{np.imag}(f_zesp)$

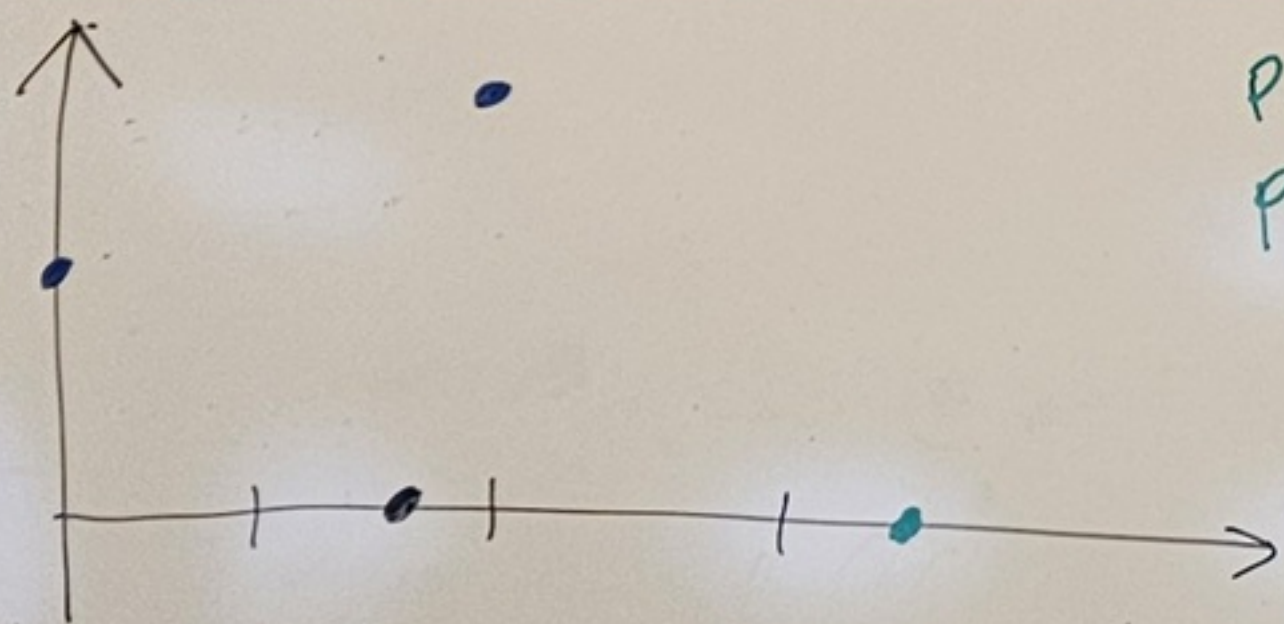
$$e^{-jt}$$

$$\text{stop} 2 ** (-1j * t)$$

$$\begin{bmatrix} 4 & 1 \\ -3 & 2 \end{bmatrix}$$



Samodzielność kodowania



Program
prototypowy

ustawić zakres na osiach x
y

x lim
y lim

wsp-x = [4, -1, 0, 3]

wsp-y = [-1, 0, 3, 2]

kolor = ['red', 'gray', 'tab:orange', 'black']

punkty_lk = [(4, -1, 'red'),
(-1, 0, 'gray'),
...]

Σ

0

0

3

3

—

punkty_ls = [{'x': 4, 'y': -1, 'kolor': 'red'},
{ 'x': -1, 'y': 0, 'kolor': 'gray'},
...]

konwersja punkty_lk
↓
punkty_ls

Wieżien:
wartość
ucieka do $\pm\infty$

Płak

max_iter = 20

NIESKONCZONOSC = 100

for

for/while

if $x > \text{NIESKONCZ}$:
break;

wieżien

$$x = \frac{1}{2}$$

$$x = 1$$

$$x = 2$$

$$\underline{x = 0} \quad \underline{x \leftarrow 2x}$$

$$x = 1$$

$$x = 0$$

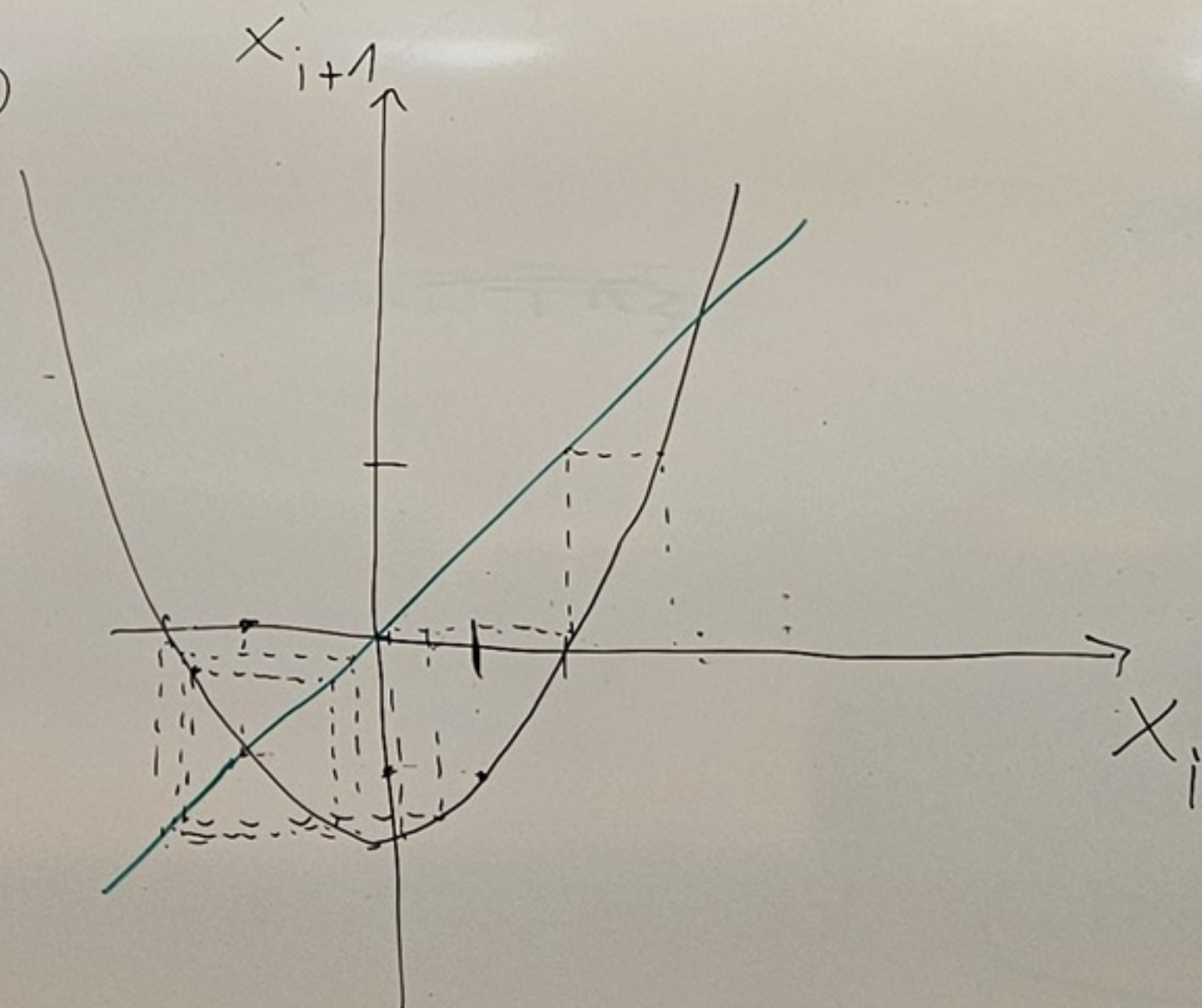
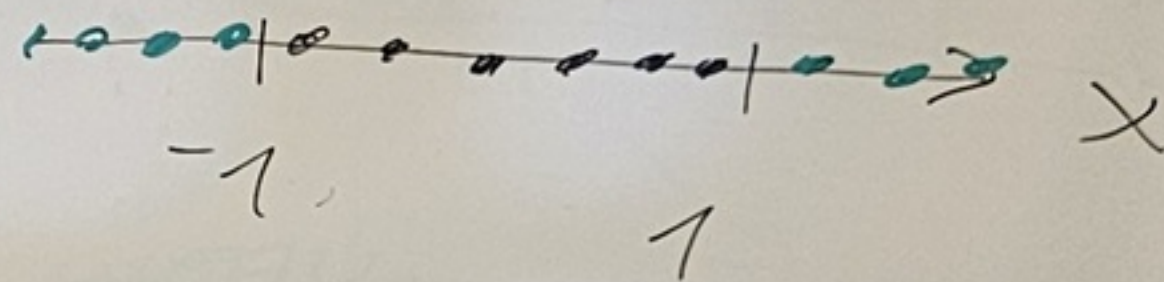
$$x = -1$$

$$x = 0$$

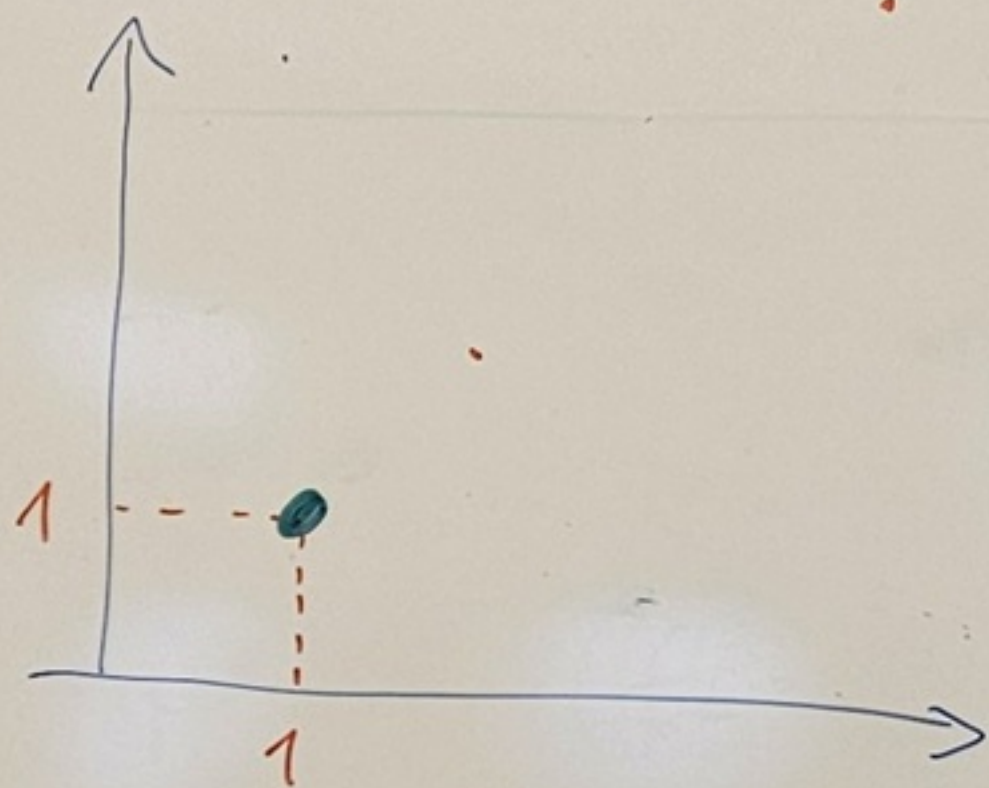
przyp. testowy

$$\underline{x \leftarrow \frac{1}{2}x}$$

$$x \leftarrow x^2 - 1$$



def pokaz.wieczniow(fun,



```
punkt = np.zeros(max_iter, dtype=complex)
```

```
wsp_x = np.real(punkt)
      -y
      imag
      plot
```

```
plot ( markersize=10 )
      linestyle='dashed'
```


$$\text{punkty} = \text{np.random.uniform}(1, 0, n_punktow) + 1j * \text{np.random.uniform}(0, 1, n_punktow)$$

$$n_punktow = 10$$

$$\text{punkt} \leftarrow \text{punkt} * 2$$

$$\text{punkt}[i+1] = \text{punkt}[i] \cdot 0.3 + \text{punkt}[0] * \text{punkt}[i]$$