```
import numpy as np
# import matplotlib.pyplot as plt
from scienceplots import plt
# from numba import jit
from tqdm import tqdm
plt.style.use(["nature", "science"])
# Period = 100

# Exponential Signal Generation
# @jit
def signal_array(delay=1, cut=10, end=110, period = 100, amp1=1, amp2=0.01):
    arr = np.zeros(int(end-cut)*100)
    for i00 in range(len(arr)):
        # every 1 period
        num = np.ceil(i00/period) # number of excited function
        num = int(num+cut)
        for k00 in range(num):
            time_in_this_period = i00 + cut * period - k00 * period
            if (time_in_this_period/delay) < 100:
                arr[i00] += amp1 * np.exp(-time_in_this_period/delay)
        # every 2 period
        num2 = np.ceil(i00/(2*period)) # number of excited function
        num2 = int(num2+cut/2)
        for k01 in range(num2):
            time_in_this_period2 = i00 + cut * period - k01 * period * 2
            if (time_in_this_period2/delay) < 100:
                arr[i00] += amp2 * np.exp(-time_in_this_period2/delay)
    freq = np.fft.fft(arr)
    # plt.plot(np.arange(len(arr)), arr)
    # plt.show()
    return freq[int(period/2)]/freq[int(period)]*amp1/amp2

delay_arr = np.linspace(1,100,100)
data = []
for delay_x in tqdm(delay_arr):
    data.append(signal_array(delay=delay_x))
# fig, ax = plt.subplots(1,2, figsize=(12,4))
# ax[1].scatter(np.arange(len(data))/100, data, marker="x",c="red",
label="data")
# ratio = np.linspace(0,1,100)
#
ax[1].plot(ratio,np.sqrt(1+(2*np.pi/(2*ratio))**2)/np.sqrt(1+(2*np.pi/(ratio
))**2),label="theory")
# ax[1].set_xlabel(r"$f_0\tau$")
```

```
# ax[1].set_ylabel(r"$signal distortion rate$")

# plt.legend()
# # plt.savefig("test_code.png", dpi=600)
# plt.show()

# fig, ax = plt.subplots(1,2, figsize=(12,4))
plt.scatter(np.arange(len(data))/200, data, marker="x",c="red",
label="data")
ratio = np.linspace(0,0.5,100)
plt.plot(ratio,1/2*np.sqrt(1+((4*np.pi*ratio))**2)/np.sqrt(1+((2*np.pi*ratio
))**2),label="theory")
plt.xlabel(r"$f_1\tau$")
plt.ylabel(r"Signal Distortion Rate")
plt.legend()
plt.savefig("test_code.png", dpi=600)
plt.show()
```