test_su_media_mobile_e_media_pesata

April 10, 2021

[1]:
```
import numpy as np
import matplotlib.pyplot as plt

def media_mobile(array: np.ndarray, finestra: int):
    shape = array.shape
    res = np.zeros(shape)

    for i in range(shape[0]):
        n = min(i+1, finestra)
        for j in range(n):
            res[i] += array[i - j]
        res[i] /= n
    return res

def media_pesata(array, k):
    shape = array.shape
    res = np.zeros(shape)

    res[0] = array[0]
    for i in range(shape[0]):
        if i == 0:
            continue
        res[i] = k * res[i-1] + (1-k) * array[i]
        # if i < 2: continue
        # res[i] += (res[i-1] - res[i-2]) * (1-k)
    return res

def testInContext(fps, seconds, noise, path_generator, k, finestra):
    style_path = 'ko'
    style_data = 'ro'
    style_mm = 'co'
    style_mp = 'mo'

    size = fps * seconds

    time = np.linspace(0, seconds, size)
    path = np.stack(list(map(path_generator, time)))

    noise = np.random.normal(0, noise, path.size)
    noise = noise.reshape(path.shape)

    data = path + noise

    mm = media_mobile(data, finestra)
    mp = media_pesata(data, k)

    if path.ndim > 2: return

    if path.ndim == 1:
        plt.plot(time, path, style_path, time, data, style_data, time, mm,␣
↪style_mm, time, mp, style_mp, ms=1)

    if path.ndim == 2:
        x = lambda arr: arr[:,0]
        y = lambda arr: arr[:,1]
        plt.plot(x(path), y(path), style_path, x(data), y(data), style_data,␣
↪x(mm), y(mm), style_mm, x(mp), y(mp), style_mp, ms=1)

    plt.show()
    return
```

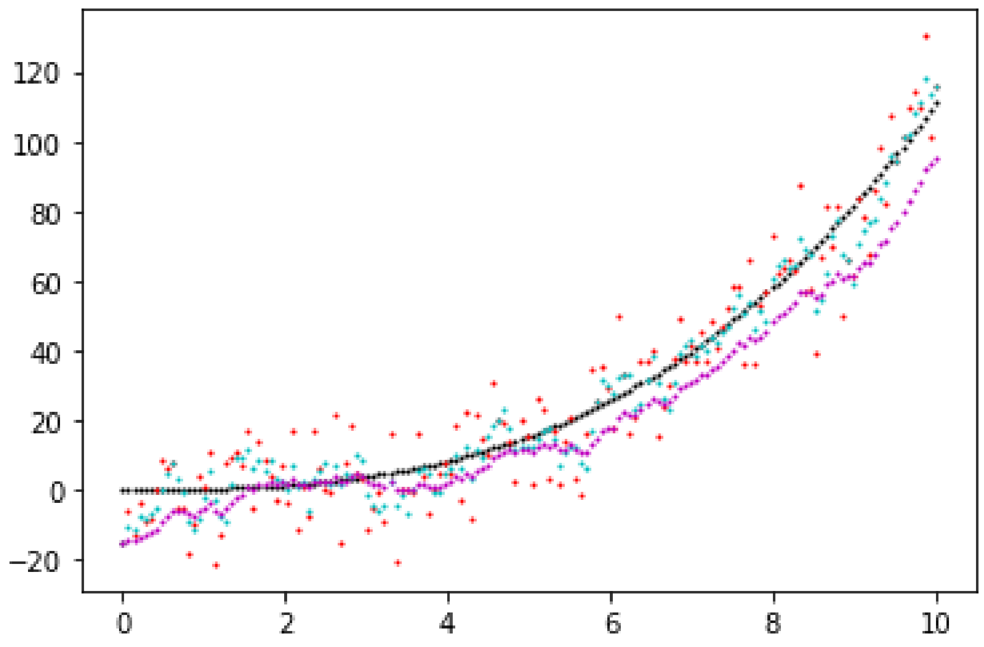
[2]:
```
fps = 15
sec = 10
noise = 10
k = .9
finestra = 3

t = lambda gen: testInContext(fps, sec, noise, gen, k, finestra)


print("NERO = REALE, ROSSO = CON RUMORE, CELESTE = MEDIA MOBILE, ROSSO = SOMMA␣
↪PESATA")
```

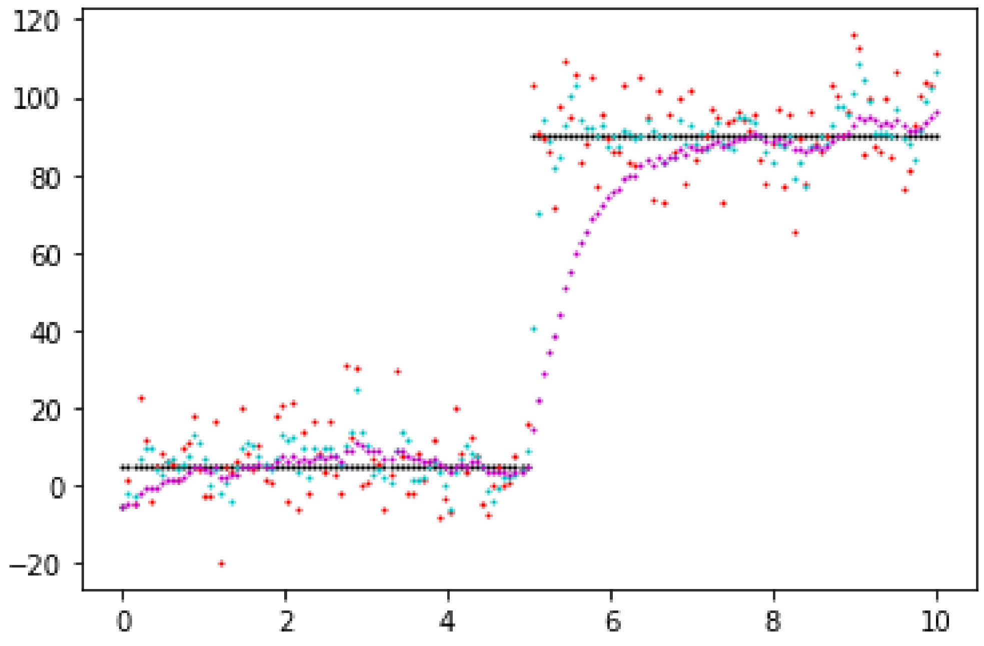
```
NERO = REALE, ROSSO = CON RUMORE, CELESTE = MEDIA MOBILE, ROSSO = SOMMA PESATA
```

[3]:
```
polinomial = lambda x: (x ** 3 + x ** 2 + x) * .1
t(polinomial)
```



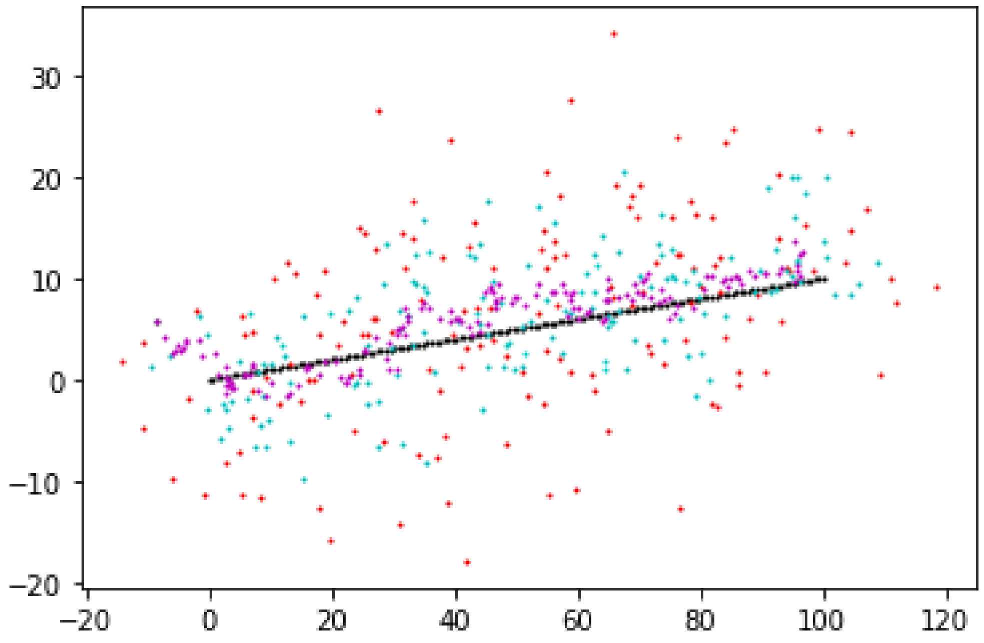
```
[4]: sharp = lambda x: 5 if x < 5 else 90

     t(sharp)
```



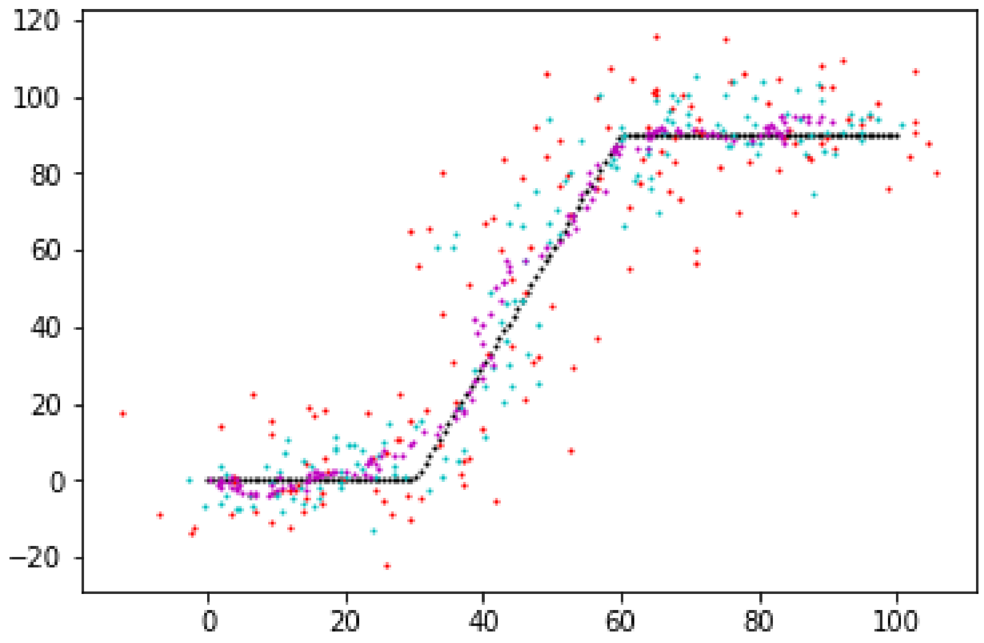
[5]:
```
def linear2d(x):
    return [10* x, x]

t(linear2d)
```



[6]:
```
def sharp2d(x):
    if x < 3:
        return [10 * x, 0 * x]
    if x < 6:
        return [10 * (x - 3) + 30,30 * (x - 3) + 0]
    return [10 * (x - 6) + 60, 0 * (x - 6) + 90]

t(sharp2d)
```


120
100
80
60
40
20
0
−20
0
20
40
60
80
100