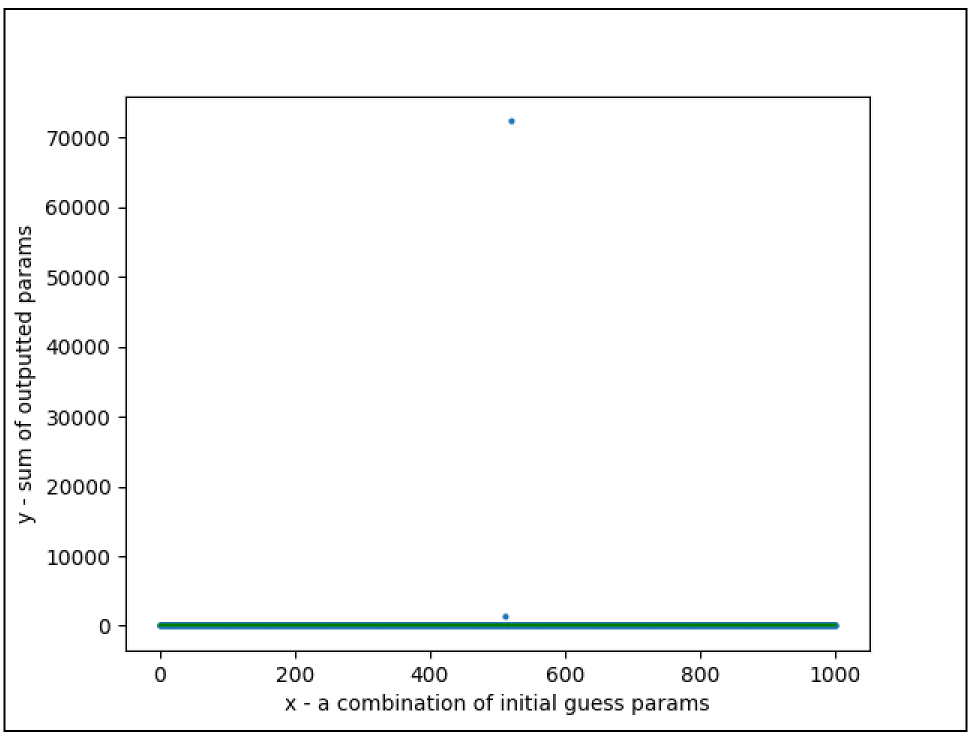
Let's consider we have a linear function **y = 6x + 3 + noise**, where scale of noise is 2.0. We will use some optimisation algorithms. Aim is to fix a certain method (algorithm) and change initial guessed parameters. Lets analyze 1000 combinations of these parameters (from 0,0,0 to 9,9,9), where **b** is free term, **k** - coefficient near x, and **n** - noise. We have a 3-level loop to implement a combination of these 1000 parameters. We will evaluate our result by comparing a line y = 10 with the sum of outputted params (in the best case this sum also 10 (6+3+1)).

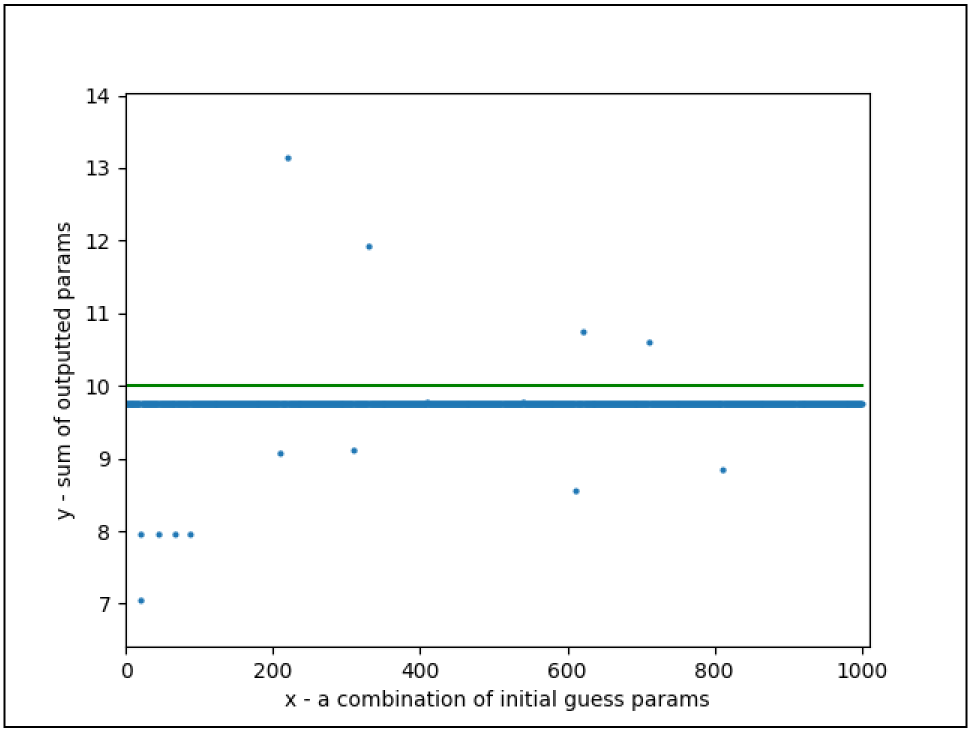
```
b_plus_k = []
for b in range(10):
    for k in range(10):
        for n in range(10):
            results = minimize(MLERegression, np.array([b, k, n]), method='Nelder-Mead', options={'disp': True},
                               jac=fprime, hess=fprime)
            b_plus_k.append(sum(results.x))
```

**Here is the result for the Nelder Mead algorithm.**

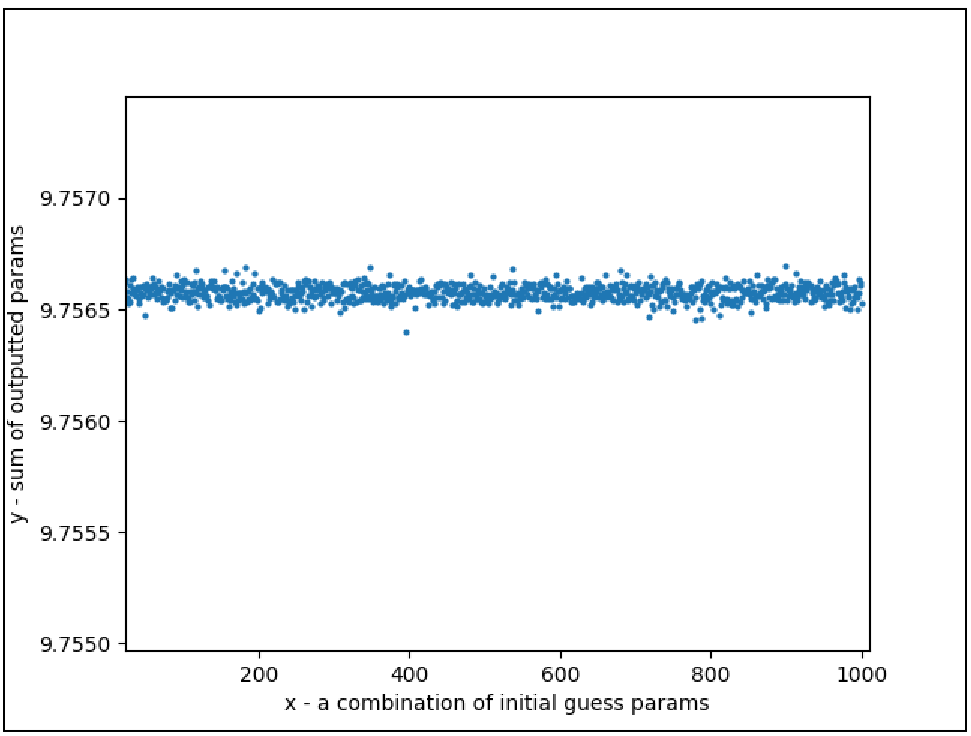
On the general, not zoomed graph we can observe two anomalies: at x=520, y=72377 and x=310, y=46



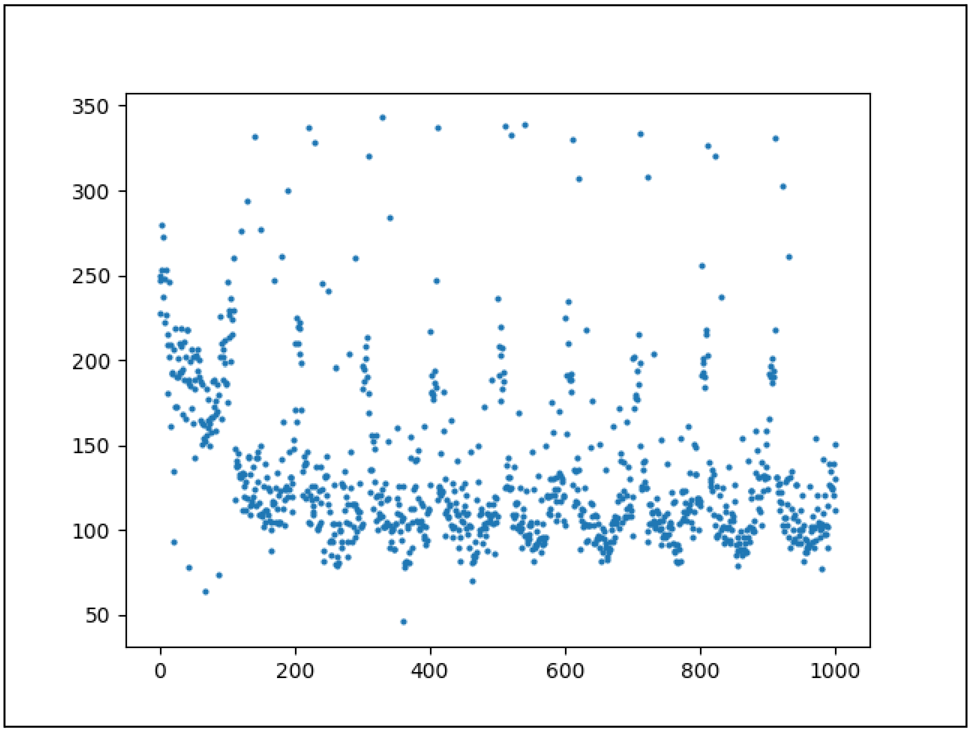

On the next, zoomed graph, we can see some errors at random points, but the general result is approximately what we've expected.




A bit more zoom and we definitely can say that most of the results are distributed between 9.7564 and 9.7567 which is a good index.

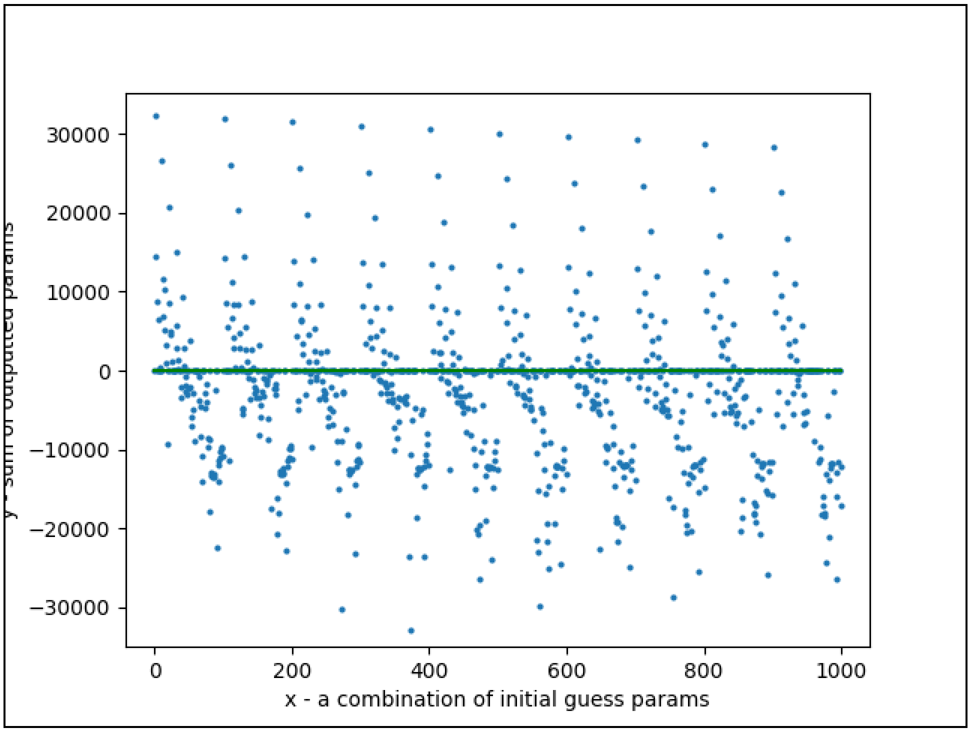



The last plot represents a number of iterations for finding k and b in dependency of initial guessed parameters. A trend can be described as a spread between 75 and 140. And a detrend between 0 to 150 ~. The initial parameters of 3,2,0 ~ had a maximum number of iterations - 340. Minimum is 3,6,0 - 30.

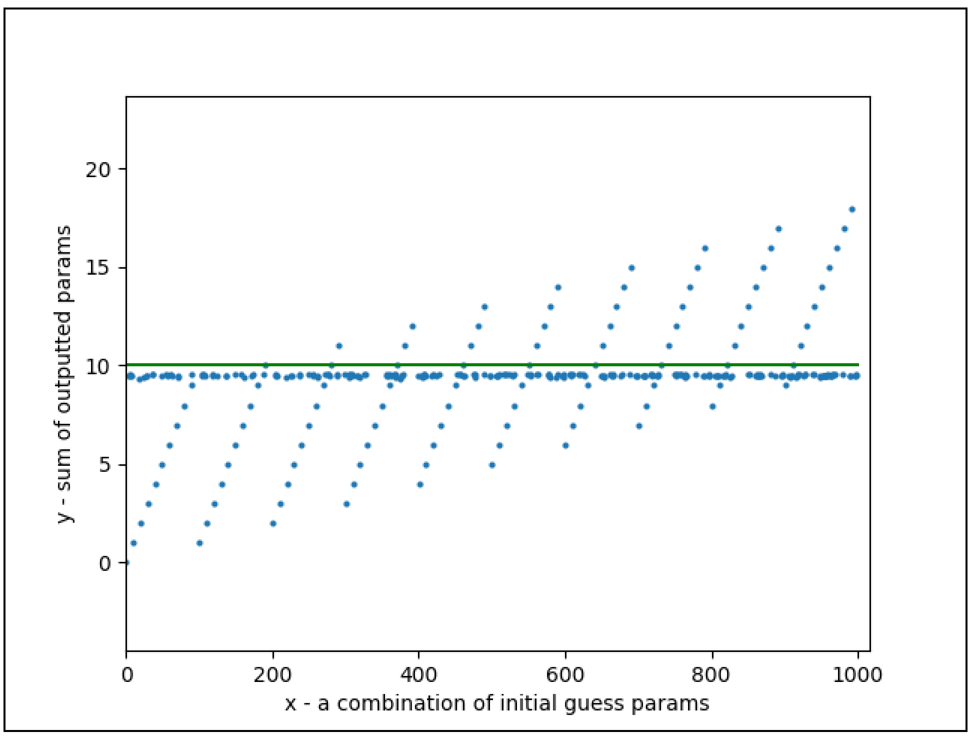



**Let's use another method called 'BFGS':**

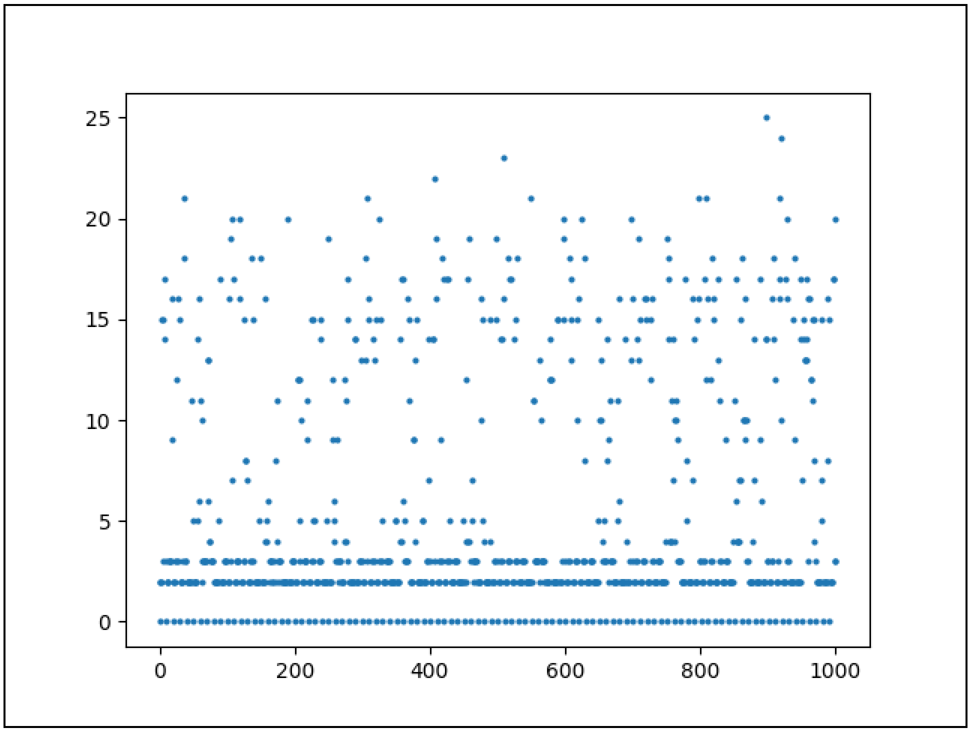
On the general view a huge spread between -30000+ and 30000+ can be seen. This distribution definitely obeys a certain pattern which repeats every 100 xs (every initial b parameter of our function).




If we zoom in a bit, despite ‘wrong’ results, we can notice those, which are close to what we expected y=10, about 9.75, like in the previous method.




In the following picture the number of iterations can be seen. In the most common case, there are 2 or 3 per one result.




**We can conclude** that the Nelder Mead method is more efficient about results, but takes much more iteration to get it, while BFGS is not precise but fast.