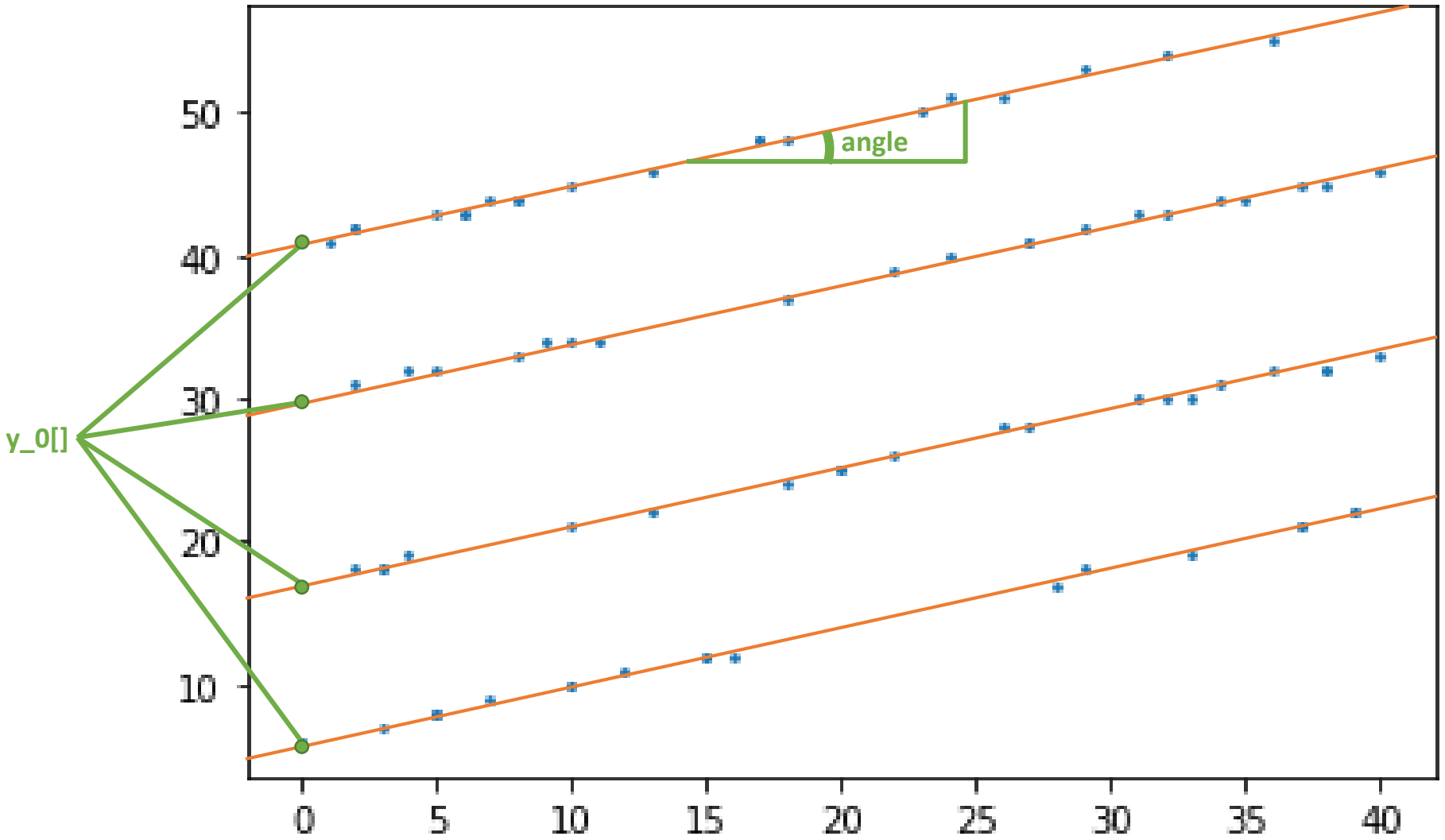
- A list of 2D points are clustered around N parallel ideal lines
- We want to find those ideal lines (the ones the best approximate the position of the given points)
- The space between to lines vary from a min 8 to a max of 20 units
- The number of lines for each input set is between 5 and 10
- The angle of the lines for each input set is between -60 and 60 degree (increasing counterclockwise)
- The points are not positioned exactly on the ideal lines but rather noisily scattered around those lines
- 15 input set are provided as examples in Json format
  - the points coordinate are given as list of x and list of y
  - the level of noise increases at every given input set (input_0 having the lowest value of noise and input_14 being the noisiest set)
- for reference purpose 7 ideal outputs are also provided (one for every other input set: result_0 being the ideal output for input_0 result_2 for input_2 and so on) - the algorithm does not have to return necessarily that exact given result but rather approximate as much as possible the ideal output

Assignment:

- Provide an algorithm that detect the N ideal lines for a given input set by returning:
  - **y_0** list (list o y_intercept of each line)
  - the **angle** that the lines share
- Identify what are possible configuration parameters for varying precision, accuracy and performance of the algorithm
- Provide a document that illustrates the solution
  - Discuss all the assumptions made and the limits of the proposed algorithm
- Bonus point:
  - Provide as an extra output of the algorithm a «measure» for the level of noise of every given input set

- Preferably the solution will be developed in Python
- No external module/libraries should be used to provide high level functionality for data processing (only for loading/dumping json file and eventually for visualization purpose, even though data visualization is not required)
- The result will be evaluated in term of correctness, efficiency (the amount of operations required), the code structure and the case analysis.


angle
y_0[]
50
40
30
20
10
0
5
10
15
20
25
30
35
40