

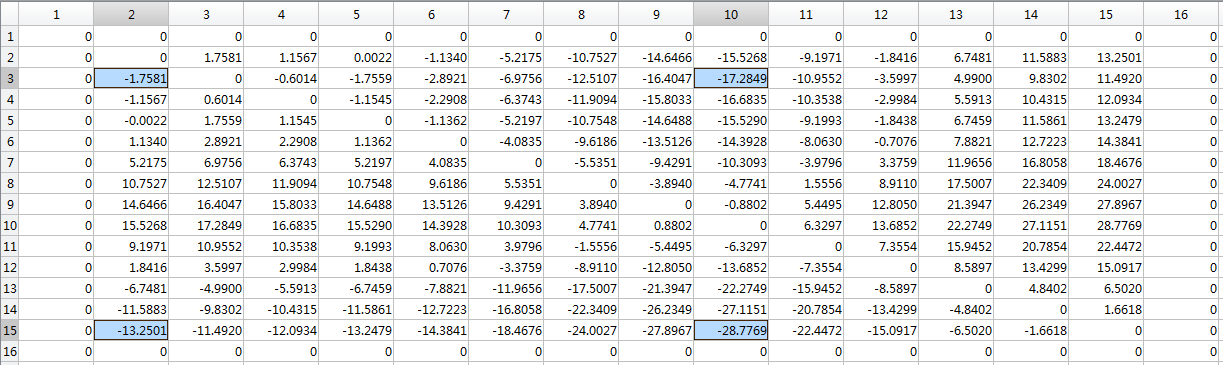
Trajectory of 220th agent.

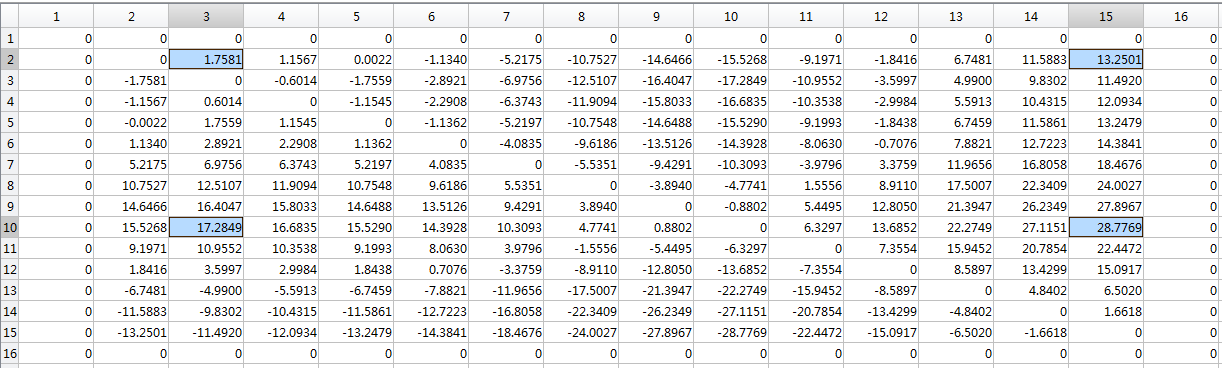


Absoluta angle of every point (vector O)



Absolute angle of every three point (vector A)



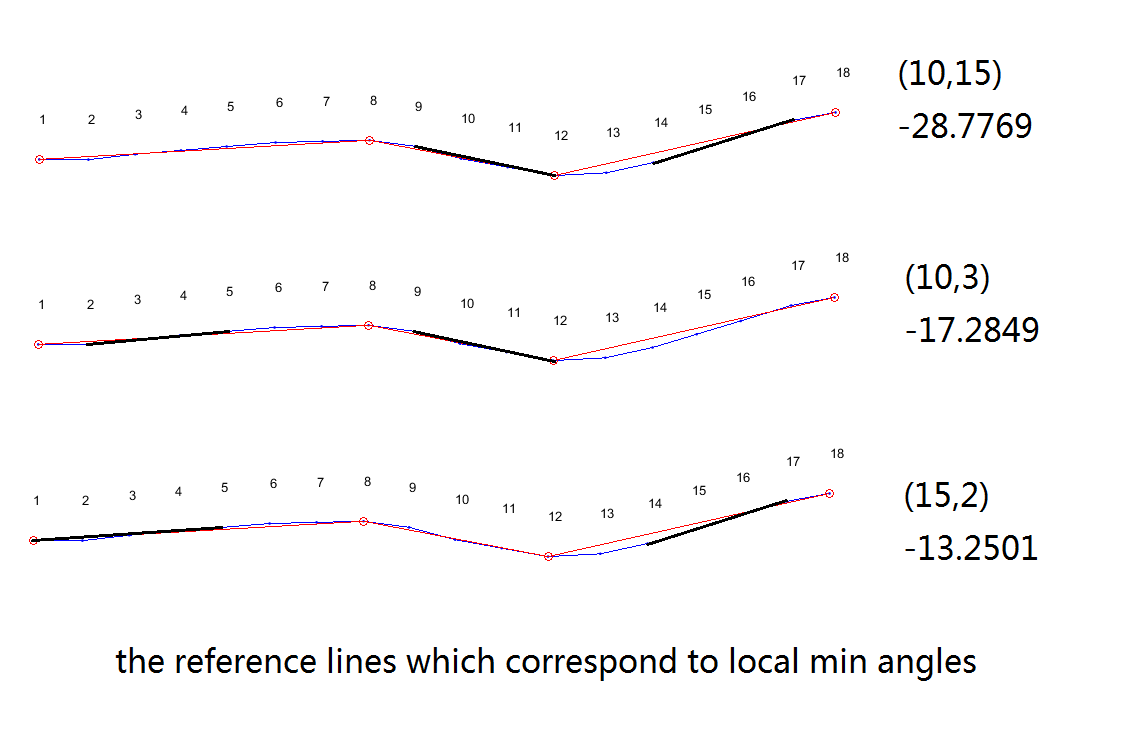


I add this figure to show how the codes work to find local max value exactly. (Old figure shows the local min value, they will bring same result.)

Matrix U (I make the 0 margin to find local max(min) value conveniently). Notice the local max(min) value with blue, correspond to row and line 2,3,10,15. As the existence of 0 margin, the real row and line should be [1,2,9,14].

I don’t understand why some of these were included. Clearly, (10,15) is the largest magnitude. But why is (2,3) here? It is not the smallest or largest magnitude. And I also don’t understand why (10,3) and (2,15) are there. Are you solving for x0,y0,x1,y1 such that the sum of the four elements (x0,y0)+(x0,y1)+(x1,y0)+(x1,y1) is maximum? If so, I don’t understand the logic behind it.

Also, why four boxes in the matrix and not some other number?



**The central idea here is to find all possible key angles and then filter them.**

As this figure shown, the largest magnitude only illustrates the first key angle difference. But it’s clear that there are three key differences. So all the local max value need to be found. The code traverses every element in matrix, to check whether this element is bigger than four adjacent elements (not just single largest magnitude). If yes, take it as local max value, and note the row and line as the positions of key angles.

Unfortunately, the second one in this figure picks up line 3 as key line while the third one picks up line 2. This outcome is caused by different direction of line 10 and 15. So we get one needless special line here. Then we need next filter method to make sure everyone in vector G is essential.

(Sometimes this method will pick up only valuable key angles with no redundancy, but we need to consider general situation. So I make this example to show why we need a filter then.)

Then [1,2,9,14]’s corresponding absolute angle in Vector A will be the vector S (S ⊂ A):

S = [-3.6395,-5.3976,11.8873,-16.8896].

I think you have understood with above part. And the following part is what I neglect in paper.

Filter Process:

Next we need to make sure the every difference between two neighbor elements in S is bigger than 6 degree threshold (with enough difference to represent fitting-line’s angle), and delete others (e.g. S2). Then we get the vector G (G ⊂ S) :

G = [-3.6395 ,11.8873,-16.8896].

(PS: if we make threshold smaller, we will get more elaborate fitting-lines. Otherwise, we will get less precise result.)

Classify process:

After getting vector G, I use it to represent fitting angle. Traverse Vector O from first element, if current element is between G1 and G2, compare the difference between current angle and G1/G2. If it’s closer to G1, then plus 1 to the counter of first fitting-line team T1. Otherwise plus 1 to T2. If current element is between G2 and G3, calculate team counter as the same process.

Then we got T = [8, 4, 6], so the subgoal vector P = [8,12,18]. I use amount of team to avoid stagger around subgoal.

I think the classify process is not very important, and illustration will be complex and tedious. So I skip this part in paper.

Actually the process is with respect to angle all the time.

1. Find local max(min) angle;
2. Filter invalid angles and get the represent angles for fitting;
3. Classify points to different team of fitting-line by their absolute angle.

I think this explanation will be helpful for you to understand the whole fitting process.