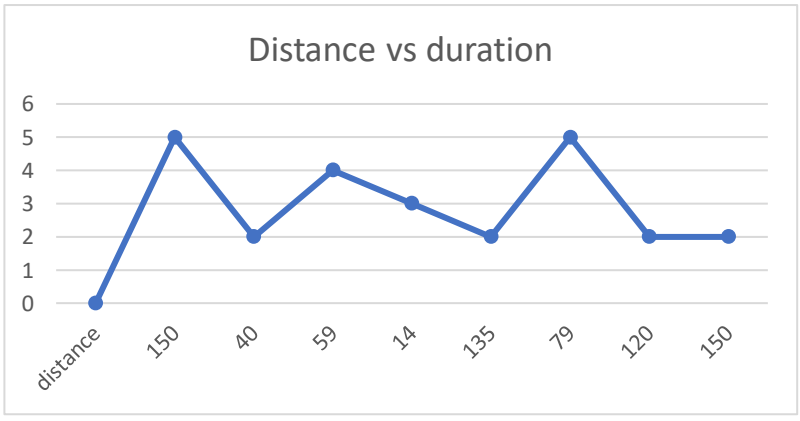
The program works as follows: user inputs the initial velocity, distance, yellow light duration and the intersection width. The program uses the given values to calculate the acceleration for two cases: car will manage to pass the intersection before the red light or it has to stop.

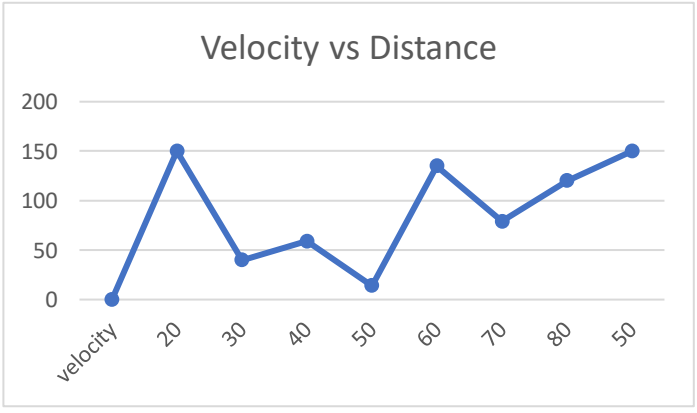
The acceleration is being calculated using $X = X_0 + V_0 t + \frac{at^2}{2}$ formula.

After a few experiments I observed that there are no cases of car passing and I had only two cases when the car would manage to stop. The rest of the experiment shows that the driver has neither the option of passing nor stopping. The above made me do another calculation: if we take the best-case scenario for passing (with maximal initial velocity and time, min. distance) I get v0 = 80 km/s (22m/s), distance from intersection = 10m, duration = 5m, width = 5m, I get an acceleration that is outside of [1,3] (given in the problem statement).

Based on the mentioned observation I conclude that there is no way the car will pass with the given parameters which means that they must change. Since the yellow light duration, intersection width and probably the initial velocity are constant in real life we have to change the parameters for the distance. Meaning the driver should make a decision of stopping or passing sooner than reaching 150m from the intersection and start increasing the velocity in case he wants to pass.

Below are the described experiments with the program's decisions and some graphs for visualization.





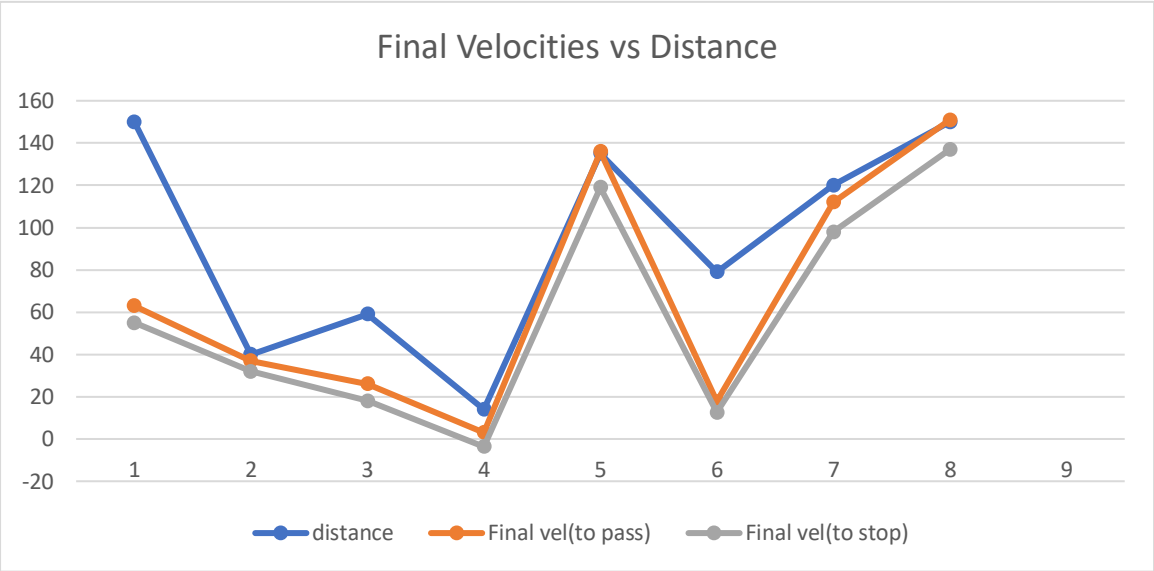
velocity	distance	duration	width	decision
20	150	5	20	NP NS
30	40	2	5	NP NS
40	59	4	15	NP S
50	14	3	10	NP NS
60	135	2	17	NP NS
70	79	5	13	NP S
80	120	2	14	NP NS
50	150	2	14	NP NS

For the second part of the project I have used $V = V_0 + at$ equation to calculate the final velocity at the end of the intersection (in case of passing) or at the beginning of the intersection (in case of stopping).

Since there are some requirements on the value of the final velocity, the program now also checks if the velocity is acceptable along with the acceleration.

I have used the same data as above to understand the situation as it is easier to connect to the real-life velocity than to acceleration values. Interestingly enough although in the first part we had 2 cases of NO PASS/STOP, here we have only one as the velocity requirement was not met. Below are charts showing the Final Velocities (to pass or stop) vs Distance.

velocity	distance	duration	width	decision	Final vel(to pass)	Final vel(to stop)
20	150	5	20	NP NS	63	55
30	40	2	5	NP NS	37	32
40	59	4	15	NP S	26	18
50	14	3	10	NP NS	3	-3.6667
60	135	2	17	NP NS	136	119
70	79	5	13	NP NS	17.8	12.6
80	120	2	14	NP NS	112	98
50	150	2	14	NP NS	151	137



For the third part of the project I have implemented the program the same way as it is for the first part with addition of the second car. The program will work the following way: if the first car stops the second car has no choice but to stop and if it won't manage(a>3 && a<1) to stop the result will be a car crush.

If the first car passes the second car has two choices: stop or pass = the program will decide based on the acceleration. It is possible that the second car will not be able neither to pass nor stop with the given parameters which means that either the car crush will happen or it will be stuck in the "middle" of the intersection.

I have conducted several experiments with the datapoints given above and I always observed the case with "no stop and no pass", hence no need to graph the results. This means that at the distance from the intersection where the cars make decision is small and their initial velocity will not be enough to pass or stop.

NOTE: Everywhere throughout the project the cars are considered as mathematical points, thus, the widths of cars are ignored.