Project Report “Analysis and Design of Hybrid Control Systems”

Sebastian van de Hoef

June 10, 2014

1 Introduction

The application to be considered in this project is platooning of vehicles from a high level coordination perspective. Platooning of heavy duty vehicles is a way to significantly reduce fuel reduction. It exploits the effect of air-drag reduction when vehicles drive with small inter-vehicle distance.

The, possibly most obvious, control challenge is related to controlling the inter-vehicle distance and performing maneuvers such as reordering, split, merge etc. Here, we however consider the problem of formation of platoons. Unless a platoon has the same start and destination, platoons have to form on the way.

While the effect of platooning takes place in the order of meters of inter-vehicle distance, traveled distances are in the range of hundreds of kilometer. Therefore it makes sense to consider membership in a platoon as a discrete variable. The vehicle position can also be approximated discretely. Here, however, we want to consider the vehicle position as a continuous quantity. Compared to the distance traveled, the distances needed to adopt speed to other vehicles are relatively short, so, we will assume that the velocity is directly controlled.

2 Modeling

We will refer to individual vehicles as agents. We will index them $1, 2, \dots, K$, so K is the number of agents. In a road network the actual road can be conveniently modeled as a discrete variable, as well. In order to not overly complicate things, we will consider all agents to travel on one, infinitely long road. So we get for the the continuous dynamics

$$\dot{d}_i = u_i, \tag{1}$$

where $d_i, u_i \in \mathbb{R}$. d_i is the position of the ith agent and u_i is the speed of the ith agent. We collect all d_i into the stack vector $d = [d_1, \dots, d_K]^\mathsf{T}$ and similarly for the inputs $u = [u_1, \dots, u_K]^\mathsf{T}$.

That a subset of agents form a platoon during a time interval, means that they stay “close”, i.e., if the difference between their d_i remains “small”. We will here assume that the length of a vehicle is small compared to the initial differences in d_i. Therefore we will say that agents i and j platoon if $d_i = d_j$. This implies that if a subset of agents forms a platoon, the agents cannot differ much in u_i. We will consider that

$$\dot{u}_i = 0, \tag{2}$$

i.e, it can only change at jumps. Jumps happen if two agents form a platoon. Furthermore, we consider that each agent moves either at high speed or low speed, i.e.,

$$u_i \in \{u_{min}, u_{max}\},\ u_{max} > u_{min},\ i = 1, \dots, K. \tag{3}$$

Then, if a subset of agents forms a platoon, all the u_i of the agents in the platoon have to jump to the same value, this is, either to u_{min} or to u_{max}. This modeling can be motivated by the fact that the range of speed of heavy duty vehicles is fairly limited and largely subject to disturbances due to the slope of the road and traffic. Therefore, one can think of u_{min} as the nominal speed and u_{max} as going as fast as possible.

2.1 Two Agents

To begin with, we will study the case with only two agents. Obviously, if the agents are initially separated, they can only from a platoon, if their initial velocities are such that their distance is decreases. We index the agent such that $d_1(0) \geq d_2(0)$. Then a jump into a platoon can only happen if $u_1(0) = u_{min}$ and $u_2(0) = u_{max}$. As soon as the two agents meet ($d_1 = d_2$) they should jump to the same velocity in order to maintain $d_1 = d_2$.

The jump condition $d_1 = d_2$ is, however, problematic. While the set $\{d_1, d_2 \in \mathbb{R} : d_1 = d_2\}$ is closed and thus fulfills the hybrid basic conditions, it is not straightforward to define the flow set such that it is closed but solutions have to jump at $d_1 = d_2$. A remedy is to inflate the jump set in some way. If then the jump set and the flow set intersect at their boundaries, the system will flow out of the flow set into the jump set and hence jump. For the simple case considered here, where we can assume that $d_1(0) \geq d_2(0)$ and that this order is maintained, we can define

$$C = \{d \in \mathbb{R}^2 : d_2 \leq d_1\},\ D = \{d \in \mathbb{R}^2 : d_2 \geq d_1\}. \tag{4}$$

An alternative would be to let the system jump when the difference in position $d_1 - d_2$ is less than a parameter ϵ. But then, we loose the property that the system jumps into "platooning mode" at $d_1 = d_2$. This can be resolved by letting the system jump to $d_1 = d_2$, which is, however, not in the flow map. This can be addressed by introducing a logic variable, which disables jumps and enables flow as soon as the two agents are platooning. We will consider the first option here.

Finally, we have to figure out the jump map. As mentioned before, as soon as $d_1 = d_2$, the agents have to jump to the same velocity. The most straightforward modeling is that the position does not change at a jump. Therefore

$$\begin{aligned} d_i^+ &= d_i \text{ for } i = 1, 2 \\ (u_1, u_2) &\in \{(u_{min}, u_{min}), (u_{max}, u_{max})\}. \end{aligned} \tag{5}$$

It is easy to see, that these two maps are out semi-continuous, and thus fulfill the hybrid basic conditions.

There is, however, still a problem with the model at this stage. Since $u_1 = u_2$ after the jump, the system will remain at $d_1 = d_2$ and thus the system will remain inside the jump set. Therefore, an eventually discrete solution is admitted, where the speed jumps arbitrarily between u_{min} and u_{max} for both agents in the same way. A reasonable controller should let the system to keep on flowing with the same and constant velocity or change velocity relatively seldom once the two agents have started platooning. So, we have to introduce come kind of memory that the jump has taken place. Therefore we introduce a logic variable p, which jumps from 0 to 1 if the jump has taken place. So, according to the above discussion, we have for p the dynamics

$$\dot{p} = 0,\ p^+ = 1. \tag{6}$$

For the interesting solutions, this is, where the trucks do not platoon initially, we have $p(0) = 0$.

Furthermore, we have to adjust the flow and jump set, to accommodate the new parameter. The system should be able to jump if and only if $p = 0$. Therefore we get

$$C = \{d \in \mathbb{R}^2, p \in \mathbb{R} : d_2 \leq d_1\},\ D = \{d \in \mathbb{R}^2, p \in \mathbb{R} : d_2 \geq d_1, p = 0\}. \tag{7}$$

In the dimension of p, the whole real axis is included in C and a point in D. Therefore, the sets are still closed. D and C overlap at $p = 0, d_1 = d_2$, which is not a problem, as we will only have $p = 0$, before the first jump, when the two agents have different velocities and continued flowing is not admissible because the system would flow of the flow set otherwise. As soon as the jump occurs, u_1 and u_2 will jump to the same value and thus maintain $d_1 = d_2$. After that jump, no more jumps will be possible, since $p = 1$.

Finally, to write the system on standard form, we introduce the state vector

$$x = \begin{bmatrix} d \\ u \\ p \end{bmatrix}. \tag{8}$$

We summarize the data of the system:

$$\begin{aligned}
C &= \{d \in \mathbb{R}^2, p \in \mathbb{R} : d_2 \leq d_1\} \\
F(x) &= \begin{bmatrix} u_1 \\ u_2 \\ 0 \\ 0 \\ 0 \end{bmatrix} \\
D &= \{d \in \mathbb{R}^2, p \in \mathbb{R} : d_2 \geq d_1, p = 0\} \\
G(x) &= \begin{bmatrix} d_1 \\ d_2 \\ \{\begin{bmatrix} u_{min} \\ u_{min} \end{bmatrix}, \begin{bmatrix} u_{max} \\ u_{max} \end{bmatrix}\} \\ 1 \end{bmatrix}.
\end{aligned} \tag{9}$$

3 Extension to Multiple Agents

We will first settle the simple part, when extending to K agents. We have for agent i the position d_i and the velocity u_i. We introduce the stack-vectors $d = [d_1, \dots, d_K]^\intercal$ and $u = [u_1, \dots, u_K]^\intercal$. We have for the flow map

$$\begin{aligned} \dot{d} &= u \\ \dot{u} &= 0. \end{aligned} \tag{10}$$

Clearly, the position cannot jump, so for the jump map we get

$$d^+ = d. \tag{11}$$

For the remaining part, there are many ways how the model for two agents can be extended to multiple agents. Here, we will consider that as soon as two agents meet, they form a platoon, which means that their positions d_i will be the same from that point in time on. This implies that u_i for each agent in a platoon must be the same. Again, we will index the agents such that for the initial conditions $d_1(0) \geq d_2(0) \geq \cdots \geq d_K(0)$. There should be a jump as soon as $d_i, d_j, i, j \in \{1, \dots, K\}, i \neq j$ changes from $d_i \neq d_j$ to $d_i = d_j$. So for the flow set, we get the condition $d_1 \geq d_2 \geq \cdots \geq d_K$:

$$C = \{d, u, p \in \mathbb{R}^K : d_i \geq d_j, i < j, (i, j \in \{1, \dots, K\})\}. \tag{12}$$

Then, as in the two agent case, there shouldn't be any further jumps due to the system staying at $d_i = d_j$. So, we have, also here, to introduce logic variables that disable these jumps. Therefore, we introduce for each agent $i = 1, \dots, K$ a logic variable p_i, that stores the index of the "platoon leader", this is, the agent with the smallest index in the platoon. A single agent is in this sense considered as a platoon of size 1. A jump happens as soon as an agent j with lower index than p_i reaches the same position d_j. We define as well the stack vector $d = [d_1, \dots, d_K]^\intercal$. Since p is a logical variable, it does not change continuously, so $\dot{p} = 0$. We get for the jump set

$$D = \{d, u, p \in \mathbb{R}^K : d_i \leq d_j, i < j, (i, j \in \{1, \dots, K\}), p_i \leq p_j - 1\}. \tag{13}$$

Note that, according to the previous description, we should have $d_i = d_j$. Since the $d_i < d_j$ for $i < j$ is not included in the flow set, we can add this part to the jump set for a more robust detection of the jump in the implementation. Furthermore, we write $p_i \leq p_j - 1$ instead of $p_i < p_j$ in order to have D closed. Since p only changes during jumps and assumes only integer values, the conditions are equivalent as far as the behavior of the system is concerned.

Finally, we have to specify the jump map. Here, the logic how the system behaves as whole is defined. We already discussed that $d^+ = d$. We defined p_i to be the index of the agent with smallest index in the platoon. So, in mathematical terms p_i has to fulfill

$$p_i = \min(j) \text{ s.t. } p_i = p_j. \tag{14}$$

Still, we assume that $u_i \in \{u_{min}, u_{max}\}$. If we do not allow for additional jumps, for all agents to eventually form one platoon, the platoon including agent 1 should always jump to u_{min} and the platoon including agent K should jump to u_{max} until the grand platoon has formed. For simplicity, we will only merge two platoons at each jump. If several platoons meet at the same continuous time instance there will be several jumps in sequence merging two platoons at a time. The agents not involved in the two merging platoons do not change their state during the jump. From the definition of the jump set, we can find the two platoons that "caused" the jump:

$$i, j : i < j \wedge d_i \leq d_j \wedge p_i \leq p_j - 1. \tag{15}$$

We define a set of the agents included in these two platoons:

$$\mathcal{N} = \{k : p_k = i \vee p_k = j\}. \tag{16}$$

With that, we can define the jump map for u and p:

$$\begin{aligned} &\text{For } k \in \{1, \dots, K\} \\ &(u_k^+, p_k^+) = \begin{cases} (u_k, p_k) & \text{if } k \notin \mathcal{N} \\ (\bar{u}, p_i) & \text{if } k \in \mathcal{N} \wedge (1 \in \mathcal{N} \wedge K \in \mathcal{N} \vee 1 \notin \mathcal{N} \wedge K \notin \mathcal{N}) \\ (u_{min}, p_i) & \text{if } k \in \mathcal{N} \wedge 1 \in \mathcal{N} \wedge K \notin \mathcal{N} \\ (u_{max}, p_i) & \text{if } k \in \mathcal{N} \wedge 1 \notin \mathcal{N} \wedge K \in \mathcal{N} \end{cases} \\ &\text{with } \bar{u} \in \{u_{min}, u_{max}\}. \end{aligned} \tag{17}$$

There are four cases. The first case applies to agents which are not involved in a merge. For them, the state remains the same. In the other three cases, the agent belongs to one of the two platoons that merge in the jump. The p_k^+ is set to the index of the "platoon leader", i.e., the agent in $\mathcal{N}$ with smallest index. In the third case, the first but not the Kth agent are part of the new platoon, so u_k^+ is set to u_{min}. In the fourth case, the Kth but not the first agent are part of the new platoon, so u_k^+ is set to u_{max}. In the second case, neither the first nor the kth agents are part of the new platoon. Therefore, the map for u_k^+ is set valued, this is, mapped to either u_{min} or u_{max}. The variable $\bar{u}$ is merely introduced for better readability.

Here, it is less straightforward to see that the mapping is outer semi-continuous. The jump map either leaves the state unchanged or maps the state to a constant depending on p. But the elements of p live on a discrete domain which makes the only sequences approaching these values be the values itself. Therefore the jump map is outer semi-continuous.

4 Simulations

In this section, I present some simulations. After the theoretical work for the two agent case, I simulated the system. The simulation results are presented in section 4.1. Once I verified that this works as expected, I continued working on the multi-agent case and and simulated the result. These simulations are presented in section 4.2.

4.1 Two Agent Case

In order to verify, that the modeling for the two agent case behaves as expected, and to get acquainted with HyEQ toolbox, I did some simulations for this case. A small adjustment on the jump map had to be made. When the solver triggered the jump, the system had already flown past $d_1 = d_2$, such that at jump time $d_2 > d_1$. The flow set is, however, defined on $d_1 \geq d_2$, so the solver stopped after the jump. A simple but practical solution to this problem is to change the jump map to

$$G(x) = \begin{bmatrix} d_1 \\ d_1 \\ \{\begin{bmatrix} u_{min} \\ u_{min} \end{bmatrix}, \begin{bmatrix} u_{max} \\ u_{max} \end{bmatrix}\} \\ 1 \end{bmatrix}, \tag{18}$$

so that d_1^+,d_2^+ are set to $d_1^+ = d_2^+ = d_1$. The the system jumps right on the boundary of the flow set. Another solution would be to change the definition of the flow set such that there is no condition on d_1, d_2 once $p = 1$.

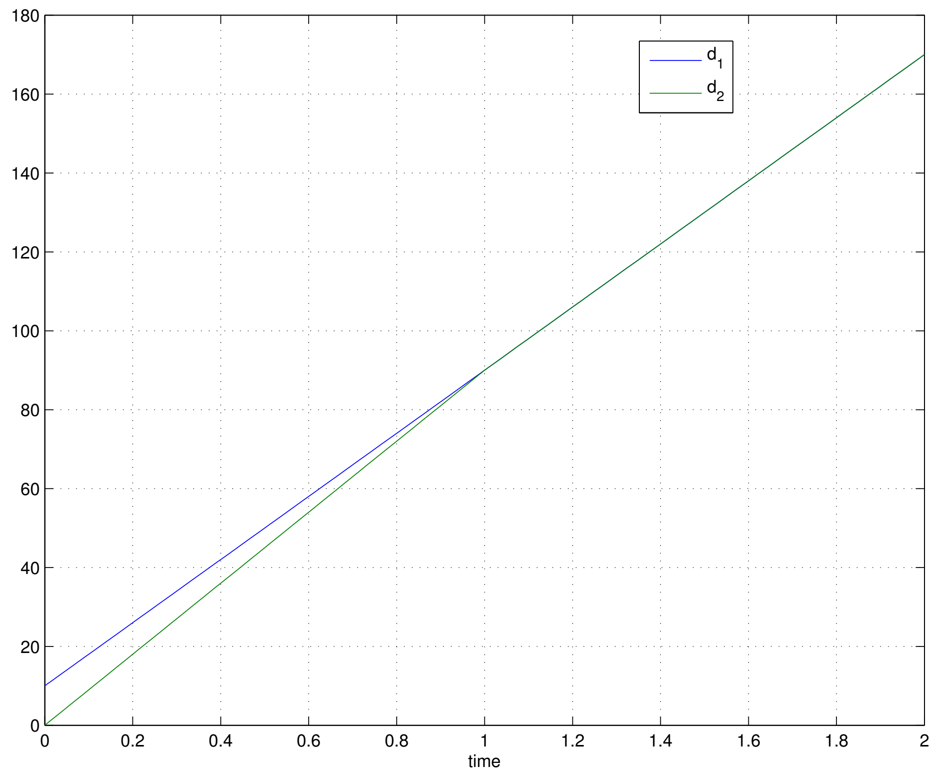

Figure 1: Trajectories of d_1, d_2 for the case of two agents.

Furthermore, we have to get rid the set valuedness of $G(x)$ for the simulation. I decided to jump to u_{min} for both agents. So we get

$$G(x) = \begin{bmatrix} d_1 \\ d_1 \\ u_{min} \\ u_{min} \\ 1 \end{bmatrix}. \tag{19}$$

In figure 1 you can see a plot of d_1, d_2 over time. I chose $u_{min} = 80$, $u_{max} = 90$, which would be realistic speeds in $\frac{km}{h}$ for trucks Europe. The initial condition where chosen to be:

$$\begin{aligned} d_1(0) &= 10 \\ d_2(0) &= 0 \\ u_1(0) &= u_{min} \\ u_2(0) &= u_{max} \\ p(0) &= 0. \end{aligned} \tag{20}$$

You can see that, as expected, the two trajectories have a constant slope and get closer to each other until they meet at time $t = 1$. There the systems jumps and from there on $d_1 = d_2$.

4.2 Multi Agent Case

In this section, I present simulations for the extension to multiple agents as discussed in section 3.

Also here, I adjusted the jump map to set d_k for $k \in \mathcal{N}$ to be the same in order to end up in the flow set after the jump and have the simulation continue. If the jump occurred exactly on the boundary of the flow set it would remain there and this reset would not be necessary.

I choose random initial conditions for d. The elements of $u(0)$ except for $u_1(0), u_K(0)$ are set randomly with equal probability to u_{min} or u_{max}. In order to get convergence to a single platoon $u_1(0) = u_{min}$, $u_K(0) = u_{max}$. I consider all agents being initially separated. Therefore each agents forms a platoon of size one, which implies that $p_i(0) = i$.

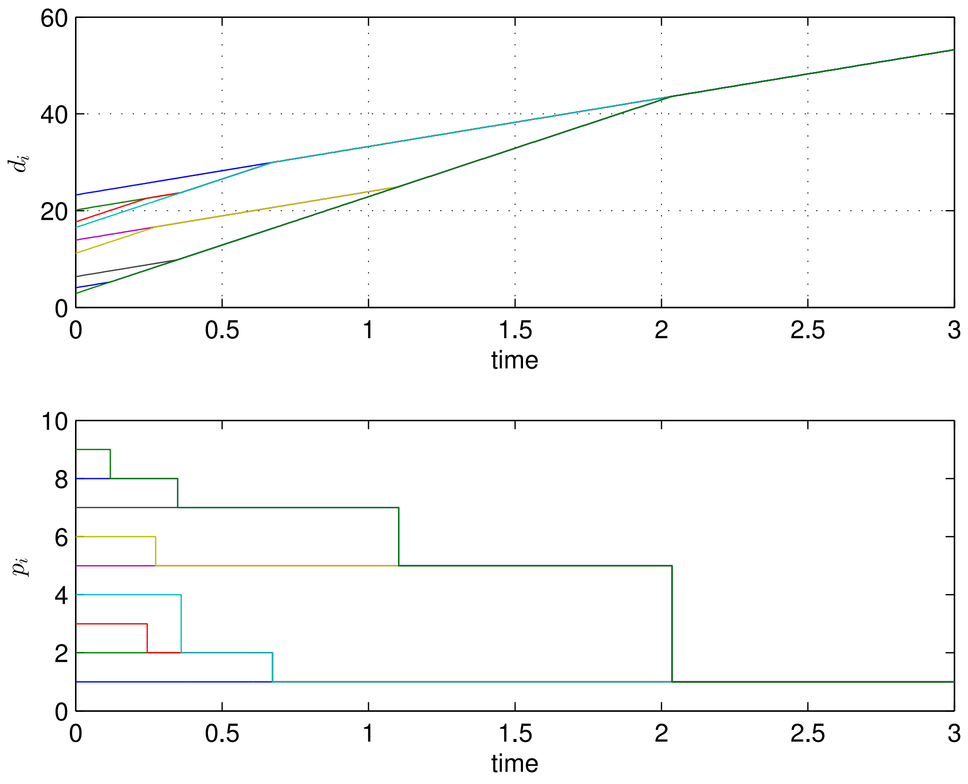

Figure 2: Trajectories of the elements of d and p for the case of 9 agents, first simulation.

Instead of having a deterministic $g(x)$, I decided to randomly select a value from $G(x)$. This means that, if two platoons not including agent both 1 and K or including both agent 1 and K join, u_i^+ for these agents is randomly set to u_{min} or u_{max}.

u_{min} was set to 10 and u_{max} to 20 in order to better see the system trajectories in the plots with many agents. I set the number of agents K to 9.

In figure 2 and 3 you can see the trajectories of d and p for two different simulation runs. The trajectories for u are omitted since they can be easily seen from the the slope of p.

We can see, that as soon as to trajectories of elements of d cross, the agents jump to same velocity, i.e., the same value for u_i and the p_is jump to the smallest index of the agents with the same d_i. The selected velocity at jumps for the other agents is random, except for the platoons containing agent 1 and $K = 9$, select u_{min}, u_{max} respectively until they meet. This happens due to initial conditions at time 2.1 in the simulation shown in Figure 2 and a little later at time 2.2 in the simulation shown in Figure 3. At this point all agents jump to the same state vector, which can be interpreted as them forming one large platoon. The velocity randomly jumps to u_{min} in the first simulation and to u_{max} in the second simulation. No more jumps occur after this point. Thus, the system behaves exactly as expected.

5 Convergence for the Multi-Agent Case

In this section, we analyze the convergence to one platoon, when the agents are initially separated. This is, we study if the system converges to the set

$$\{d, u, p \in \mathbb{R}^K : d_i = d_j, p_i = p_j, (i, j \in \{1, \dots, K\})\} \tag{21}$$

from initial conditions in the set

$$\begin{aligned}(d(0), u(0), p(0)) \in \{(d, u, p) \in \mathbb{R}^K : d_i \geq d_j, i < j, (i, j \in \{1, \dots, K\}), p_k = k, (k \in \{1, \dots, K\}),\\ u_1 = u_{min}, u_K = u_{max}, u_l \in \{u_{min}, u_{max}\}, (l \in \{2, \dots, K-1\})\}.\end{aligned} \tag{22}$$

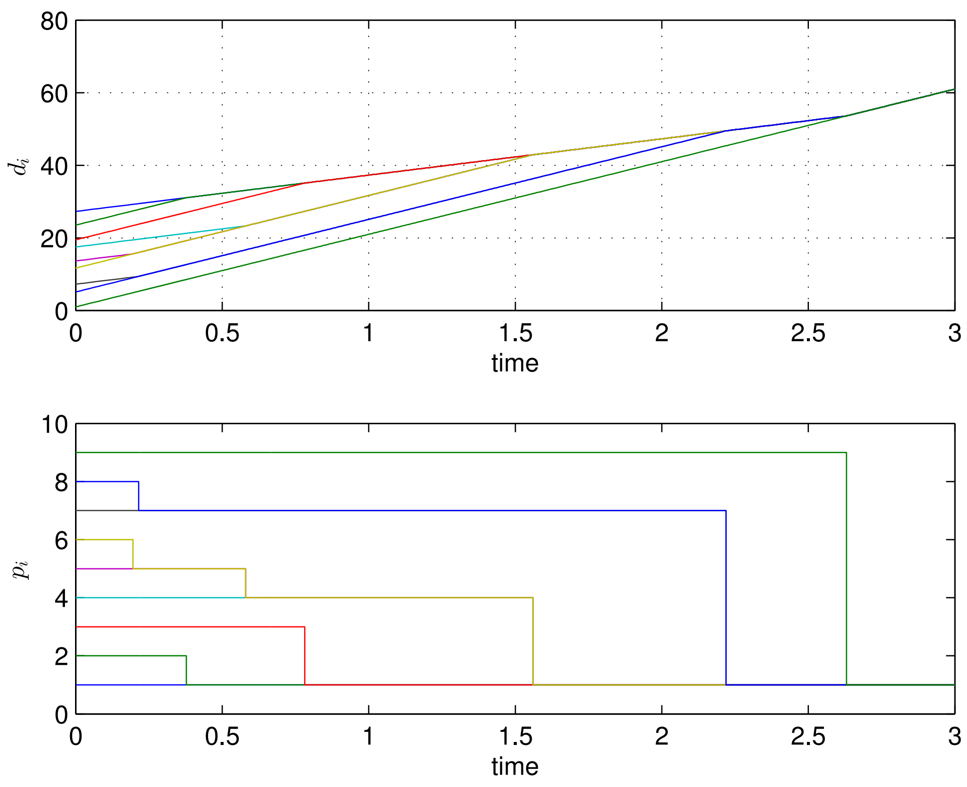

Figure 3: Trajectories of the elements of d and p for the case of 9 agents, second simulation.

Due to the simple system dynamics during flows, we can explicitly calculate the trajectories during flows. We have that p and u are constant, and $d_i(t)$ are lines with slope u_{min} or u_{max}, depending on the state u_i. At jumps $d^+ = d$. Therefore, the trajectories of d_i are piecewise linear functions.

The first thing to notice is that the system is eventually continuous. From the definition of the jump set (13), the system can only jump if $p_i \neq p_j$ for $i \neq j$. After the jump, all agents with $p_k = i$ or $p_k = j$ have state p_i. So, p_k remains the same for the agents with $p_k = i$ but decreases for the agents with $p_k = j$, which is at least one agent, that is agent j, that has $p_j = j$. But since there is only a finite number of agents and p can only assume a finite number of values, the number of jumps are finite.

Furthermore, we have that solutions that start in the flow set and fulfill $(d_i(0) > d_j(0) \Leftrightarrow p_i(0) < p_j(0))$, are complete. We have C compact, and continuous solutions during flow time, so, the only way a solution might end is that it flows to the boundary of C or jumps out of it. We know from the simulations that the system indeed flows up to the boundary. Therefore, we need to discuss if the system can always jump when it reaches the boundary of the flow set and what happens when it jumps.

We have

$$C \cap D = \{d, u, p \in \mathbb{R}^K : d_i = d_j, i < j, (i, j \in \{1, \ldots, K\}), p_i \leq p_j - 1\} \subset \partial C. \tag{23}$$

This shows us, that jumps can only happen on parts of the boundary of C. Since u is limited to two different values $u_{min}, u_{max} : u_{min} < u_{max}$, the only way the system can flow out of the flow set is if two agents i, j flow up to $d_i = d_j$ and have different state $u_i \neq u_j$. Otherwise, $\dot{d}_i = \dot{d}_j$, so they will stay on the boundary of C. But if the system is at the point where it could leave the flow set, it will jump. In this situation, the condition $d_i = d_j$ is fulfilled.

To verify that $p_i \leq p_j - 1$ holds, when the system is about to leave the flow set, we need to consider the jump map. That two agents i, j flow up to $d_i = d_j$ and have different state $u_i < u_j$ means that an arbitrarily small amount of continuous time before, we had $d_i > d_j$. But then we assumed that initially $p_i(0) \leq p_j(0) - 1$. Then the system jumps to the same state $p_i^+ = p_j^+ = p_i$ and the same value $u_i^+ = u_j^+$. But then u_i, u_j as well as p_i, p_j will jump to the same value in all future jumps, which means that this pair of agents exactly stays on the boundary of C but stays outside of D. Since $p_i^+ = p_j^+ = p_i$ the property $d_i > d_j \Leftrightarrow p_i \leq p_j - 1$ still holds after the jump. So, by induction, we can conclude that the solutions can be always be continued for the assumed initial conditions.

In order to show convergence, we first note that the fact that the system remains inside the flow set implies,

$$d_1 \geq d_j \text{ for } j \in \{2, \dots, K\}, d_K \leq d_j \text{ for } j \in \{1, \dots, K-1\}. \tag{24}$$

In particular $d_1 \geq d_K$. But from the jump map we can see that unless $d_1 = d_K$, we have $u_1 = u_{min}$ and $u_K = u_{max}$. Then,

$$\frac{\mathrm{d}(d_1 - d_K)}{\mathrm{d}t} = \dot{d_1} - \dot{d_K} = u_{min} - u_{max} < 0. \tag{25}$$

Since $d_1 - d_K \geq 0$, the system reaches $d_1 = d_K$ in finite time. So with (24), we can conclude that the system reaches $d_1 = d_2 = \dots = d_K$ in finite time. This time t_0 can be simply calculated to be

$$\begin{aligned} \frac{\mathrm{d}(d_1 - d_K)}{\mathrm{d}t} = u_{min} - u_{max} \Leftrightarrow (d_1 - d_K) = (d_1(0) - d_K(0)) + (u_{min} - u_{max})t_0 \\ (d_1 - d_K) = (d_1(0) - d_K(0)) + (u_{min} - u_{max})t_0 = 0 \Leftrightarrow t_0 = \frac{d_1(0) - d_K(0)}{u_{max} - u_{min}}. \end{aligned} \tag{26}$$

At this point, all agents $i \in \{1, \dots, K\}$ jump to $p_i = 1$ and all agents jump to either u_{min} or u_{max}. This implies that after t_0, no more jumps can occur and $\frac{\mathrm{d}(d_1 - d_K)}{\mathrm{d}t} = 0$. So, for $t > t_0$, the system remains at $d_1 = d_2 = \dots = d_K$.

6 Conclusion

In this project, I have modeled high level platoon formation as a hybrid system. In order to keep the complexity low, I assumed simple dynamics for the individual vehicles. Starting with the simple case of only two vehicles, it was possible to model the system including a hybrid controller as a multi-agent system. The controller was designed to let all vehicles form one large platoon. On the way to convergence to the large platoon, vehicles "build up" the grand platoon by merging smaller platoons starting from individual vehicles.

I showed that the system indeed behaves as expected, i.e., converges into one large platoon in finite time, given appropriate initial conditions. Furthermore, I provided simulations, which illustrate the behavior of the system. Only minor adjustments had to be made in order to successfully simulate the system with the HyEQ toolbox.

There are multiple extensions to this work possible. The modeling of trucks as single integrators is, in my opinion, a reasonable abstraction in order to study high level coordination schemes. It is assumed here, that there exist low level controllers that keep the desired speed and take care of keeping the inter-vehicle distance once the platoons are formed. I might be interesting to consider more complicated vehicle dynamics, such as second order systems and model the controllers for speed and inter-vehicle distance controller, explicitly. Also, the limitation to only two set-points for the speeds are somewhat restrictive and could be relaxed in future work.