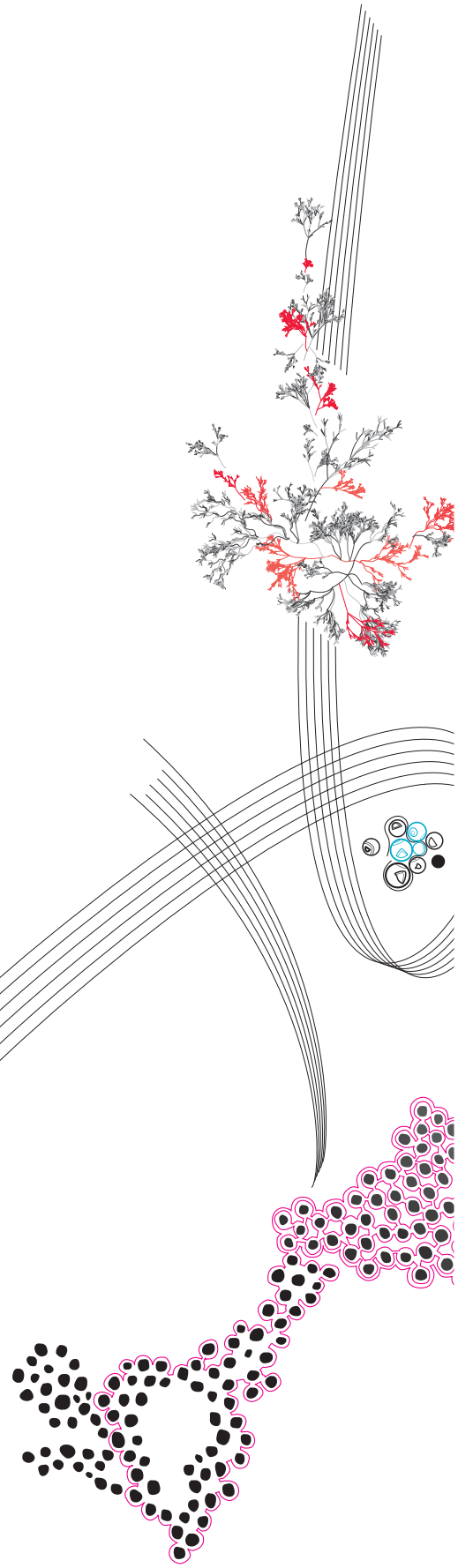


BSc Thesis Applied Mathematics

Spring Damper based Platoon Control for Merging Cars

Jesse van Werven

Supervisor: G. Meinsma

June 28, 2019

Department of Applied Mathematics
Faculty of Electrical Engineering,
Mathematics and Computer Science

UNIVERSITY OF TWENTE.

Spring Damper based Platoon Control for Merging Cars

Jesse van Werven*

June 28, 2019

Abstract

Vehicle platooning is a promising solution to the ever-growing crowdedness of highways. By eliminating humans from the driving process, not only can it be made safer, but also more efficient. Platoon structures allow for better fuel economy for the cars within the platoon, which in turn allows platoons to drive further on less fuel. This paper proposes physics based controllers for platoon control. Controllers are designed based on the physical properties of springs and dampers. The performance of these controllers is then analyzed using metrics such as safety, error convergence, and passenger jerk. Finally, controller parameter are selected based on a cost function in order to tune controller performance to performance criteria.

Keywords: platoon control, spring damper control, reference trajectory generation

1 Introduction

The world's population is growing, and with it, the number of vehicles on the road. As economies grow, more and more people can afford to drive a car to work, leading to increased congestion during morning rush hours. Moreover, a growing world population results in a growing demand for goods. As transporting goods by truck is still one of the most efficient ways for companies to supply distant markets not reachable by boat, we can also expect the amount of trucks on the highways to increase[1]. A solution to the overcrowding of our highways has been proposed, however. The concept of platooning allows multiple cars to drive safely in a formation and could result in safer roads and less fuel consumption.

The European Association of Car Manufacturers defines truck platooning as follows: "Truck platooning is the linking of two or more trucks in convoy, using connectivity technology and automated driving support systems. These vehicles automatically maintain a set, close distance between each other when they are connected for certain parts of a journey, for instance on motorways"[2].

Platooning has several advantages: not only does it lessen the burden on the driver to keep a safe distance from the next truck, allowing him to do other things, it also helps save fuel due to the aerodynamics of the platoon. It has been shown that fuel consumption is reduced by up to 16% for follower vehicles, and 8% for the lead vehicle [2]. Additionally, an increase in efficiency occurs since roads can be used more effectively. Finally, theoretically, platooning should improve safety due to the elimination of human reaction time from the stopping distance.

*Email: j.t.vanwerven@student.utwente.nl

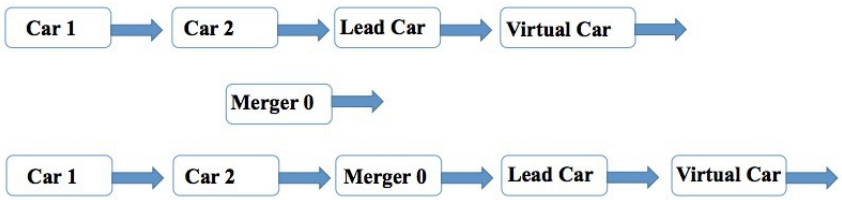


FIGURE 1: Merging maneuver for a platoon

Of course, new cars must be able to join the existing platoon. This maneuver is referred to as a merge, with the car joining the platoon being referred to as the mergers. An example of a merging maneuver is shown in figure 1

However, in order for the vehicles to drive autonomously in platoon formation, the distance between them needs to be regulated somehow.

In this paper, we control the distance between cars using a physics based spring damper controller. In this control method, the cars behave as though there is a physical spring and damper between them. This control law is expected to have desirable properties with regards to stability and error convergence.

1.1 State of the Art

Platoon control has been approached from many directions. In order to sketch an idea of the current state of techniques, we first identify some of the most common strategies used by the control methods:

- Behaviour based
- Virtual Structured
- Leader Follower

Behaviour Based: In behaviour based control strategies, cars (often referred to more generally as agents) are assigned behaviours by the controller. This approach is especially common in military applications. Behaviours could be "move to object" or "stay in formation". Using behaviours in this way helps with performing multiple goal-based tasks simultaneously[3]. Control inputs can be given based on the relative importance of the behaviours the agent is assigned[17]. The primary advantage of this strategy is that it allows for multiple objectives to be attempted simultaneously, in parallel. Moreover, these objectives have a physical meaning, which could make it easier to analyze and evaluate the performance of the agent. Additionally, since it is distributed, i.e. each agent is individually assigned some behaviour to execute, it requires less communication[17]. Disadvantages include needing to derive a model of the platoon formation and dynamics, as well as potentially unstable formations. [17].

Virtual Structure: In the virtual structure method, a formation is first designed. Then, this formation is given some trajectory to follow. Agent trajectories are then derived from the trajectory of the virtual structure. Due to this approach being centralized, errors are prone to being propagated and can lead to unstable systems[17].

Leader Follower: In the leader follower approach, a platoon leader is chosen. The followers then attempt to match their trajectory to that of the leader. This structure leads to a more intuitive and more easily implementable controller, as a large share of regular control techniques can be applied when this control design paradigm is chosen. The primary disadvantge is that the method assumes the lead car will be able to maintain trajectory. If this is not the case, then any errors will be propagated throughout the platoon[17].

Some common platoon configurations include column, echelon, wedge and line formations. The echelon formation is used in military and agricultural applications; the wedge formation is niche to military applications[3].

It is important to note that a lot of controllers are synthesized from a state space model through designing a transfer function, rather than using a physically meaningful structure or concept. However, in this paper, the physics based background of the controller is the main focus, so we will not discuss these methods.
Additionally, some controllers use the Model Predictive Control structure, with knowledge of several cars' states to predict the best possible control input [12].

Some leader-follower controllers implement some form of physics based controller. For example, a fluid, newton, and friction force has been proposed to model a lateral and longitudinal platoon in [6]

Additionally, though not necessarily from a physics based perspective, many controllers also implement a PID algorithm for platoon control[11][7][9]. PID controllers have also been used to merge into platoons and split out of platoons[9]

Some PID platoon controllers tune their parameters using Nyquist gains [11]. Genetic Algorithms are also used in platoon control[10], and in general to tune PID parameters [14][15]. However, not a lot of research has been done concerning how to merge into platoons with physics-based controllers, especially in terms of choosing parameters for double-sided springs. When the springs are one-sided, critical damping has been proposed as a parameter choice for lateral and longitudinal control [8]

Of special interest in this paper is whether there can, through some control method, be chosen a set of parameters that make the merging maneuvre and the platoon control have desirable properties in, for example, error convergence, safety, or fuel consumption.

FIGURE 2: An image of different platoon structures. From left to right, the image shows the column, line, echelon, and wedge structure. Image taken from [13]

The method outlined in this paper focuses on providing a stable controller for platoon con-

trol using the physics inspired method of spring-damper control. Controller performance with respect to a merging manoeuvre is investigated. Additionally, we will compare different implementations in a few standard scenarios which are deemed relevant for evaluating the controllers. Finally, the different implementations will also be tested on their performance in specific aspects, such as the jerk felt by the passengers and the energy used by the platoon during the merge.

2 Platoon Properties

In this section we describe properties of the platoon, and explain what a merge is and how it takes place.

2.1 Platoon Definitions

We will first define what a platoon is, and certain important properties that we shall need.

Definition 2.1. Platoon A platoon P consists of n cars $c^{(1)}, c^{(2)}, ..., c^{(n)}$ where each $c^{(i)}, 1 < i < n$ in front of car $c^{(i-1)}$ and behind $c^{(i+1)}$. The cars $c^{(1)}$ and $c^{(n)}$ are the last and lead car, respectively, of the platoon P. All cars in P are assumed to be driving laterally. The positive direction is to the right, as indicated in figure 3.

In a similar fashion, we define M to be the group of all merging cars during the simulation. This group has no purpose other than being a collection of merging cars.
A merging car can join the platoon once it is within a distance d_{merge} of the platoon. Furthermore, the platoon identifies there is a merging car when the merging car is within a distance d_{identify} of the platoon. The merging car joins the platoon by inserting itself in between the 2 platoon cars that are closest to it at the moment the merge is initiated, as shown in figure 3

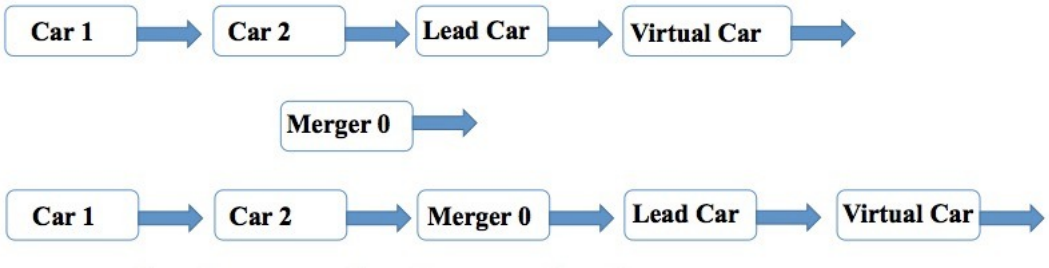


FIGURE 3: A car merging into a platoon.

A car is assumed to be in a platoon P unless otherwise indicated by the subscript m. In this case $c_m^{(j)}$ is a the j^{th} merging car.

Definition 2.2. Given a platoon P of n cars, $c^{(1)}, c^{(2)}, ..., c^{(n)}$, $|P|$ is defined to be the number of cars in the platoon, n. Similarly, for the group of all merging cars, M, consisting of $c_m^{(1)}, c_m^{(2)}, ..., c_m^{(n)}$, the number of all merging cars during the simulation is defined as $|M|$.

Definition 2.3. Let $c^{(i)} \in P$. Then the position of $c^{(i)}$ at time t is given by $x^{(i)}$. This is position is actually the time dependent $x^{(i)}(t)$, however, the time dependency is omitted since it is implied in the definition. Similarly, the velocity and acceleration at time t is given by $v^{(i)}$ and $a^{(i)}$, respectively. For the sake of notational brevity, we will assume that cars denoted by $c_m^{(j)}$ are mergers. For $c_m^{(j)} \in M$, the velocity and acceleration at time t is given by $v_m^{(j)}$ and $a_m^{(j)}$.

Thus, P consists of a single file of cars with positions $x^{(i)}$ driving at speeds $v^{(i)}$ with accelerations $a^{(i)}$, for $1 \leq i \leq n$. M is a set of cars that will eventually merge into the platoon.

2.2 Errors

We construct the following errors for each car $c^{(i)}$, $c^{(i+1)} \in P, 1 < i < n$, given a reference distance between the cars $c^{(i)}, c^{(i+1)}$ of $r(t)$:

$$e^{(i,i+1)}(t) = x^{(i+1)}(t) - r(t) - x^{(i)}(t) \tag{1}$$

$$e^{(i+1,i)}(t) = -e^{(i,i+1)}(t). \tag{2}$$

The second equality holds because the forward error between car i and $i+1$ is the backward error between car $i+1$ and i, only in the opposite direction. To avoid lengthy notation, we shall refer to $e^{(i,i+1)}(t)$ more conveniently as the forward error $e^{(i)}$. This immediately implies the backward error of car i is $-e^{(i-1)}$, where $-e^{(i-1)}$ is then the forward error of car $i-1$.
For car $c^{(1)}$ only the first equation is applicable, as it is the last car in the platoon, so it only has an error with the next car:

$$e^{(1)} = x^{(2)} - r(t) - x^{(1)}. \tag{3}$$

For the lead car, $c^{(n)}$ we define $e^{(n)}$ as follows:

$$e^{(n)}(t) = x^{(n+1)} - r(t) - x^{(n)}. \tag{4}$$

Here, $x^{(n+1)}$ is the position of the virtual lead car, a car that is not part of the platoon and follows an independent trajectory. This is the trajectory we would like the lead car to follow.

We therefore have a number of cars driving in sequence, with the distances between each car being controlled, and the lead car attempting to follow the virtual lead car.

In order to control the distances between the cars, we use a spring-damper inspired controller, which will be introduced in the next section.

2.3 Model of the cars

The cars used for the simulation are masses that are governed by Newton's laws of motion. Once they are moving, they continue their motion unaltered until a force is exerted on them, after which the change in their behaviour can be analyzed using Newton's second law of motion. We also trivially have the following relations for each car:

$$\dot{x}^{(i)} = v^{(i)} \tag{5}$$

$$\dot{v}^{(i)} = \ddot{x}^{(i)} = a^{(i)} \tag{6}$$

$$\dot{a}^{(i)} = \dddot{x}^{(i)} = j^{(i)}. \tag{7}$$

Here $j(t)$ denotes the jerk of the car. The car's acceleration is governed by Newton's second law of motion, as given below in (8). The equations above can be analyzed for each car by determining the net force on a car, after which $x^{(i)}, \dot{x}^{(i)}, \ddot{x}^{(i)}$, and $\dddot{x}^{(i)}$ can be found through integration or differentiation.

$$F_{\text{net}} = m\ddot{x}^{(i)}. \tag{8}$$

The cars have a maximum acceleration and a maximum deceleration of $10m/s/s$ and $-10m/s/s$ respectively.

2.3.1 Friction force

In our model we have also included friction force, as this force is always present on the cars, and could pose a problem for control strategies if not compensated for properly. Since the cars are driving through air, a drag force is applied to them. The force of air friction on the cars, also known as the drag force, is given by:

$$F_D = \frac{1}{2}\rho v^2 C_D A. \tag{9}$$

where ρ is the mass density of the fluid, which in this case is air. C_D is the coefficient of drag for the car, A is the car's cross sectional area. Finally, v is the speed of the car relative to the speed of the air. We will assume v is the velocity of the car, in other words, the air is stationary for the duration of the simulation.

2.4 Assumptions

In making the simulation, it is assumed that the cars can only move laterally (need to make picture). By extension, this means when a merging car wants to join the platoon, the cars in the platoon can only create space through slowing down or speeding up.

The net force on a car can be adjusted through controlling the car's engine force, which is assumed to be possible without input lag. However, mergers that are not within a distance d_{merge} of the platoon are not controllable.

During the simulation, the acceleration a, velocity v, and position x, and by extension the errors between the cars, e, are all known perfectly.

Finally, during a sufficiently small time step, the acceleration is constant for all cars $c_i \in P \cup M$. This assumptions allows for the discretization of the model into finite time steps, which can then be analyzed numerically using the kinematic equations for constant acceleration.

2.5 Discretization

The system is discretized in the following steps: First, in a small time-step, we calculate the input force for each car based on the controller algorithm. Note that we assumed that during a small time-step we may regard the acceleration as constant. Therefore, we may use the kinematic equations of motion.

The input force is used to calculate the net force on the car. The net force is translated to an acceleration over the time period δ_t through $a = \frac{F_{\text{net}}}{m}$. If the car's calculated force is larger than $ma_{\max}$, then the force is set to $ma_{\max}$. The velocity update is performed through multiplying the acceleration of the car with the time-step, as given below in (10). Here v_f is the final velocity of the car at the end of the time-step t, and v_0 the velocity of the car at the start of the time-step, if the car has an acceleration of a.

$$v_f = v_0 + at. \tag{10}$$

Finally, given a time-step t, and letting Δ_x be the lateral displacement during t, v_0 the velocity of a car at the beginning of the time-step, and a the acceleration during this timestep, then the lateral displacement is calculated through the kinematic equation

$$\Delta_x = v_0 t + \frac{1}{2} a t^2. \tag{11}$$

These equations are used to update the car's position, velocity, and acceleration.

2.6 Verifying the physical simulation

Before any controller simulations are investigated, it remains to show that the discretization as described in the previous section yields results which are sufficiently close to the continuous equivalent. We perform a simulation where a car drives for a certain time with a certain speed and a certain acceleration, and then we see if this matches with the analytic result.
In order to verify the simulation results, we assume that the jerk is constant

$$a(t) = 0.1\, t. \tag{12}$$

Integrating this twice results in the displacement being given by

$$x(t) = 0.1 \frac{1}{6} t^3. \tag{13}$$

Running the simulation for 1000 seconds, with a time-step of 0.01, we obtain as final x position: $x_{\text{final}} = 16666916.66$ to 2 decimal places. The analytical result is 16666666.67, also rounded to 2 decimal places. This means after 15 minutes, the simulated distance is about 300 meters from the analytic result. This is deemed acceptable. Therefore, all simulation results are obtained with a step-size of 0.01 unless otherwise stated.

2.6.1 Discretization problems

The car's energy as a function of time should be non-increasing if the forces applied to it are spring and damper forces. Therefore, ideally, a trajectory of sinusoidal form will maintain constant amplitude. However, in the simulation this is not the case: the amplitude of the sinusoid is dampened over time. This is presumable because the total energy contribution does not stay perfectly constant due to sampling. However, this dampening effect is lessened when the time-step is decreased, leading to the conclusion that this is indeed (for a large part[1]) due to discretization of a continuous time system. This phenomenon is observed in figure 4

[1] There are possibly additional factors at play, such as machine epsilon

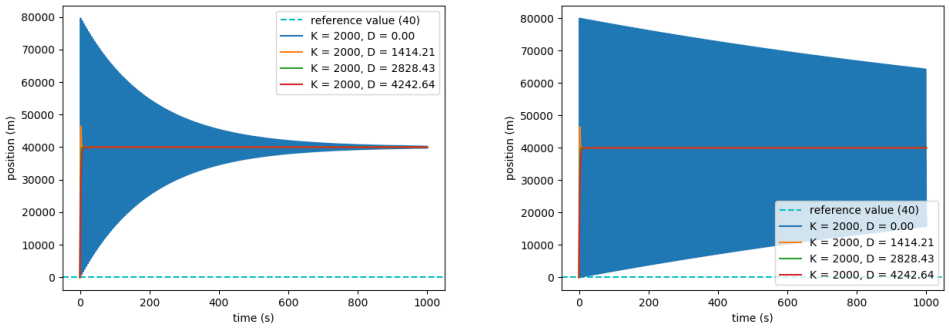


FIGURE 4: A simulation run over 1000 seconds with $K = 20$, $D = 0$, with one car being controlled by a spring force, a critically damped force, and an overdamped force. As can be seen from this image, $K = 20$ does not produce a pure harmonic. However, as the time-step gets smaller, the approximation of a pure harmonic gets better.

2.7 Controller metrics

In order to analyze the controller performance, we introduce the following metrics:

- Jerk experienced by passengers
- Time spent within an unsafe distance from the next car

2.7.1 Jerk experienced by passengers

Jerk has been linked to motion sickness, and it is generally a design goal of motion planning and motion control to minimize the jerk felt by passengers [18]. Jerk has been linked to discomfort while driving, and therefore it is interesting to look at minimizing this. This metric is therefore also kept track of during the simulation for each car.

2.7.2 Time spent within an unsafe distance from the next car

The international convention of traffic safety in vienna has determined, amongst other things, a safe distance between autonomous cars. However, this distance is given as the requirement that "The driver of a vehicle moving behind another vehicle shall keep at a sufficient distance from that other vehicle to avoid collision if the vehicle in front should suddenly slow down or stop"[2]. Several papers have proposed a finalized solution for this [16][19]. However, we have decided to use the checker from [16], because this checker has been proven with the use of a formalized checker. We implement the safe distance check as shown in the appendixB:

This checker is not identical to the one used in the paper: a small modification has been made for when the cars are not in the precondition. In this case, following the papers methods one would obtain infinitely large safe distances. Instead of this, when the precondition is not true, we return safe distance 1 as the safe distance, which is always guaranteed to be safe. Therefore, the simulation may have some false negatives, where the car is deemed unsafe but is actually still safe. More time could be spent investigating this.

[2]Safe distance formulation is found on page 15 of https://www.unece.org/fileadmin/DAM/trans/conventn/crt1968e.pdf

3 Spring Damper Control

3.1 A Physical Example

Consider a cart with a mass on it, connected to a wall by a spring, as shown in figure 5. The position of the cart is denoted by x, the mass of the cart by m, and the spring constant by K. Suppose the cart, initially at rest, is attached to a spring, while it is at $x = x_0$. We now assume, without loss of generality, that the spring is centered at $x = x_{\text{rest}}$. The dynamics of this cart are

$$m\ddot{x} = -K(x - x_{\text{rest}}). \tag{14}$$

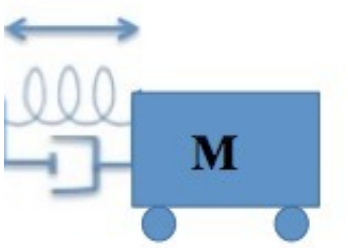


FIGURE 5: A cart with mass M connected to a spring and a damper

Example 3.1. Suppose the cart in figure 5, initially at rest, has initial position $x = 2$, and that the spring is centered around $x = 40$. As can be seen in figure 6, the line corresponding to $K = 20$, $D = 0$ starts at $x = 2$. The position of the cart then gradually increases to around $x = 76$, after which it again returns to its initial position. Then we have the previous system, as in equation (14), with some initial conditions for the cart:

$$m\ddot{x} = -K(x - 40), \tag{15}$$

subject to $\dot{x}(0) = 0,\ x(0) = 2$.

The solution of this initial value problem is a sinusoid with amplitude 38, and a shift of 40 up. Thus the cart is always oscillating between $x = 78$ and $x = 2$. It is interesting to note that there is a relationship between the mass of the cart and the spring constant. If we want to reach the value of 40 within a few seconds, then the spring constant K must be chosen of the same magnitude (or greater) than the mass of the car.

If the oscillatory behaviour was dampened, then the cart would eventually become stationary at $x = 40$. This can be done by applying a damping force on the cart, proportional to its velocity. Then the cart system would look like this:

$$m\ddot{x} = -Kx - D\dot{x}. \tag{16}$$

Dividing by m and rearranging, we obtain

$$\ddot{x} + \frac{D}{m}\dot{x} + \frac{K}{m}x = 0. \tag{17}$$

Analyzing the characteristic equation $r^2 + \frac{d}{m}r + \frac{k}{m} = 0$, we can obtain the following:

$$r_{1,2} = -\frac{\frac{D}{m} + -\sqrt{(\frac{D}{m})^2 - 4\frac{K}{m}}}{2}. \tag{18}$$

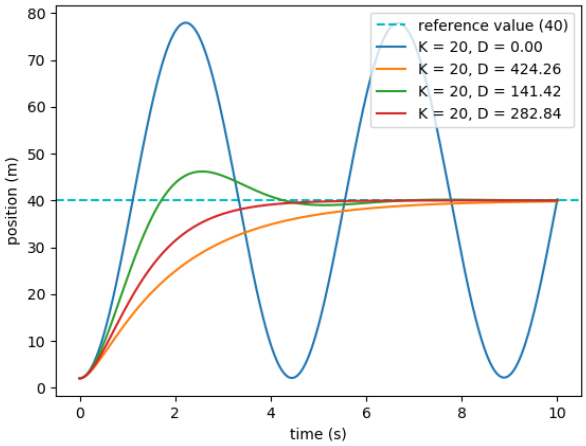


FIGURE 6: The effects of varying D while keeping K constant for a car with mass 1000 kg. In this simulation the position of a mass is being controlled by a spring and a damper at different values for K and D. The damping value D is given to 2 decimal points.

Underdamped: Analyzing this equation, we see that if $S = (-\frac{D}{m})^2 - 4\frac{K}{m}$ is smaller than zero, then S imaginary, so the solution will be a pair of complex conjugates. This system is often referred to as underdamped, as the poles of this system are partially imaginary.

Overdamped: If $S > 0$, the solution decays exponentially with time, as described below:

$$x(t) = c_1 e^{r_1 t} + c_2 e^{r_2 t}. \tag{19}$$

One root is less negative than the other, so this exponential decays slower. Therefore, this root dominates the decay process. This process is called overdamped, and can be seen in figure 3, where the orange line depicts an overdamped system. It is interesting to note that this system does not pass the equilibrium line, as it does not contain an oscillatory component.

Critically Damped: Finally, if $S = 0$, we have a critically damped system, and our solution has the form of

$$x(t) = c_1 e^{r_1 t} + c_2 t e^{r_2 t}. \tag{20}$$

Here the roots r_1 and r_2 are equal, and thus the decay process is not dominated by a slower exponential. Some desirable properties can be seen in figure 6. For example, it reaches the equilibrium in a smooth manner, without overshoot, and after it does so, it stays at the equilibrium. [5] Replicating such behaviour for larger systems could be desirable.

3.2 Controller Formalization

We formalize the spring damper based controller explained in the previous section and motivate why it is useful for platoon control.

Let the error e be defined as $e := r - y_m$, r being the reference signal, and y_m the measured signal. In the controller this force is proportional to its displacement from its rest length,

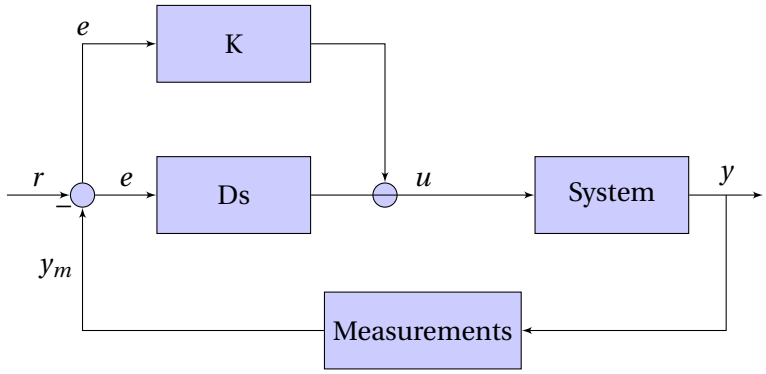


FIGURE 7: A block diagram for a general spring damper controller. A reference signal r is chosen for the system, the difference between y_m and r is defined as the system error e. This error is both differentiated and multiplied by D, and also multiplied by K. The input for the system is the force u, which is chosen as $u = Ke + D\dot{e}$

which is an error of 0, i.e. when $r = y_m$. This is the ideal control goal. The spring force of the controller can be written as follows:

$$\mathrm{F}_{\text{spring}} = -K(y_m - r) = Ke. \tag{21}$$

This results in oscillatory behaviour around the desired value, where $e = 0$. However, in order to make the oscillations decrease in magnitude, we draw another physics-inspired parallel to dampers. Let D be the damping coefficient, and $\dot{e}$ be $\frac{de}{dt}$, the time derivative of the difference between the measured signal and the reference signal.

$$\mathrm{F}_{\text{damper}} = -D\dot{e}. \tag{22}$$

The damper opposes the change in e. Referring back to figure 6, it can be seen that the amplitude of the oscillations gets damped. More than that, however, the controller uses the derivative to approximate the change in e, and also to predict the controller behaviour. By constructing an input force u_d defined as above, we mimic this behaviour.

Finally, choosing the control input force as follows

$$u = Ke + D\dot{e}. \tag{23}$$

This makes the error converge to 0 under the above assumption. More so, however, it also forces the system to mimic the physical behaviour of a spring and a damper. Since the springs and dampers between cars, when tuned correctly, can provide a safe and robust system, this makes it ideal for platoon control, as explained in 4.2.2

3.3 Properties of Spring Damper Control

Using this type of controller offers some advantages. First of all, the approach is very localised. This means that each local controller will be interested in the error with their neigbours (i.e. the car in front of it and the car behind it). It is generally true that global error convergence suffers when the controllers are localized. However global control requires expensive, inaccurate sensors such as satellites, whereas local control can be implemented

much more easily with on-board sensors.[7]

In order to illustrate another aspect of this localized control, consider the following: suppose we control a platoon of 5 cars, initially at the positions [0, 2, 4, 6, 8], where each entry in the vector corresponds to the position of a car in the platoon. Suppose we desire them to be at [2, 4, 6, 8, 10]. This means we move the leader, which is at position 8, to position 10, and set all the reference distances between the follower vehicles to be 2 meters. On top of being a very simple control structure, ideally, the controller also should, immediately, through virtue of its design, keep the cars from colliding with one another.

However, as mentioned before, the downside of this control structure is that if the leader cannot follow its reference properly, none of the followers will follow their reference either. Regardless, it is expected that the error will be controlled quickly as well as smoothly by the spring damper controller, as long as the lead car is not obstructed during the journey.

4 Methods and Models

4.1 Two Sided Spring Damper Controller

4.1.1 System Dynamics for Two Sided Controller

Consider a platoon P with $|P| = n$. Then the system being controlled consists of n horizontally aligned masses connected by n springs and n dampers**??**. Ignoring friction, we obtain:

$$\mathrm{F_{net}} = \mathrm{F_{engine}} = m\ddot{x}. \tag{24}$$

The variable x denotes the displacement of the spring from its rest length. Newton's third law can be applied to each mass, leading to a system of equations of the following form for the engine force of car i

$$\mathrm{F_{engine\ i}} = -k_{i-1}e^{(i-1)} - d_{i-1}e^{(i-1)} + k_i e^{(i)} + d_i e^{(i)}. \tag{25}$$

Note that the error $e^{(i)}$ is the forward error of car i, and thus for $i = 0$, $e^{(i)} = 0$, leading to the following equation for the engine force of car 1:

$$\mathrm{F_{engine\ 1}} = k_1 e^{(1)} + d_1 \dot{e}^{(1)}. \tag{26}$$

The engine force is applied purely through the controller in such a way as to make the cars behave as though there are real, mechanical strings between them. This is illustrated in figure 8

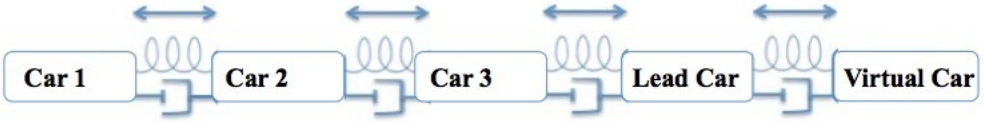


FIGURE 8: Visualization of the platoon under the double sided spring control law.

4.1.2 Merging car dynamics for Two Sided Controller

Consider a platoon P, and a group of mergers M. Let $c_m^{(j)} \in M$. The platoon observes a merging car when it is within a distance d_{id}. Now suppose the two cars closest to the merger are $c_p^{(i)}$ and $c_p^{(i+1)}$. Once the merging car is within a distance d_{merge}, these cars are selected by the controller, and the spring and damper between $c_p^{(i)}$ and $c_p^{(i+1)}$ is removed. A spring and damper is reattached between $c_p^{(i)}$ and $c_m^{(j)}$, and another between $c_m^{(j)}$ and $c_p^{(i+1)}$. Once these springs and dampers have been reattached, the merging car is effectively part of P, and becomes $c_p^{(i+1)}$. All other cars in the platoon after $c_p^{(i+1)}$ have their index shifted by 1, i.e. $c_p^{(i+1)} \longrightarrow c_p^{(i+2)}$.

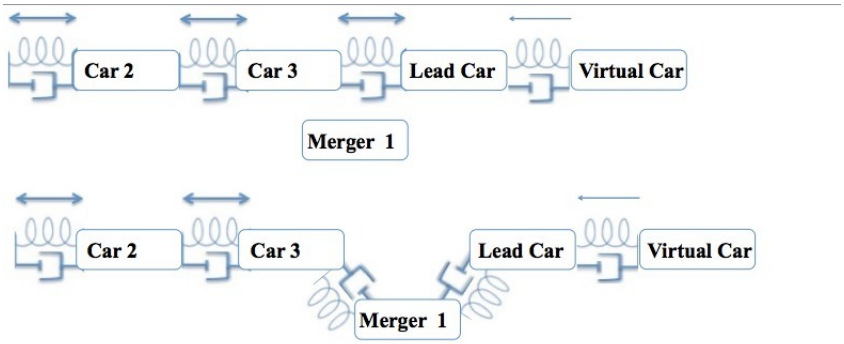


FIGURE 9: How the springs reattach during the merging procedure for a two-sided spring damper controller

As such, the control structure of the platoon is highly localized: each car in the platoon only "sees" the cars in front and behind it. The controller is constructed in such a way that the cars also only attempt to achieve local error convergence: each car wants the errors between itself and the cars in front and behind it to be minimal, and only the lead car follows the reference trajectory given by the virtual lead car. The advantage of such a structure is that there is a high local stability, since each car $c_p^{(i)}$ controls its distance between the car previous to it, $c_p^{(i-1)}$, and the car in front of it, $c_p^{(i+1)}$.

4.1.3 Two Sided Controller performance

As we can see from the figure, the controller performance depends very heavily on the choice of K_i and D_i. In section 3, we already saw that the mass of the car and the spring force were both contributing factors in the determination of the period of the solution. As K increases, the time taken for the controller to move the cart to the equilibrium point decreased. We see this in the simulation also:
In this simulation we have a platoon with initial positions [12, 22, 32], velocities of [50, 40, 50], and initial acceleration of 0. The virtual lead car starts at position $x = 32$ and travels with speed $70m/s$ for the duration of the run. The virtual lead car is followed by the platoon car that is also at $x = 32$, and is not part of the platoon.

In figure 10, we can see that the error converges rather slowly, and after about 100 seconds it is still not 0. This is because the controller has a spring constant of $K = 20$ which is much smaller than the mass of the cars (which is 1000). The velocity of each cars approaches the

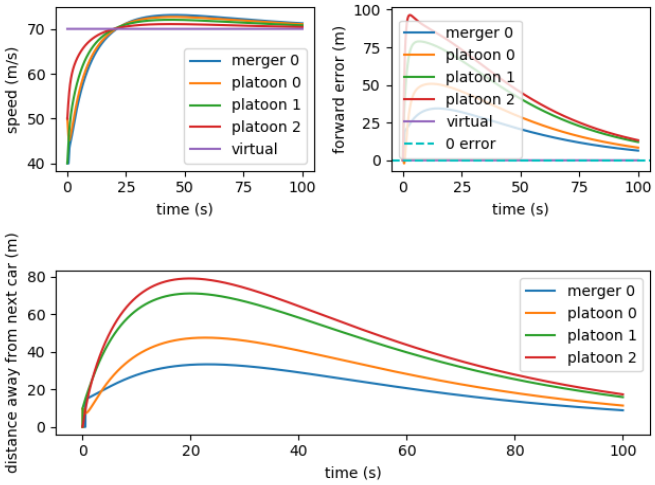


FIGURE 10: A simulation run over 100 seconds with $K = 20$, $D = 1000$, and starting positions for the platoon of [12, 22, 32], velocities of [50, 40, 50], and initial acceleration of 0. The virtual lead car starts at position $x = 32$ and travels with speed $70m/s$ for the duration of the run. The merging car starts with at position $x = 2$ with a horizontal velocity of $40m/s$ merges at $t = 0.42s$. It can be seen that for K much smaller than m (the mass of the car), the controller converges very slowly. However, after 25 seconds, most cars have a speed roughly around the desired speed, which is that of the virtual lead car.

virtual lead car's velocity after about 25 seconds. The middle car has a velocity of 40, but this is adjusted quickly enough that the car does not collide with other cars. This is a rather slow convergence for platooning purposes. However, the controller is very safe and the distance between the cars is never negative (which would imply a collision).

Additionally, as can be seen in figure 11 the jerk of the cars over time spikes very high very early. This could be a by-product of the simulation, since each car starts at an acceleration of 0, but due to the absence of time delay, the next acceleration could be 10, making the jerk $j = \frac{da}{dt} = \frac{0-10}{\text{timestep}}$ with time-step = 0.01. This would produce a jerk around -1000. Additionally, the peaks around 2000 could be from the fact that, again, the car can change its acceleration from $a = -10$ (its maximum deceleration) to $a = 10$ (its maximum acceleration) within a time-step, resulting in a derivative of around 2000. Finally, the distance between the cars appears, at least from the simulation results, to be safe within a few seconds.

Now we run the same simulation, with the same parameters, only with $K = 200$. In this simulation we have a platoon with initial positions [12, 22, 32], velocities of [50, 40, 50], and initial acceleration of 0. The virtual lead car starts at position $x = 32$ and travels with speed $70m/s$ for the duration of the run.

The simulation results are shown in figure 12. The safe distance metric shows that this choice of K and D is inferior to the previous choice. This might be because the controller lets the cars oscillate harder, but does not dampen the oscillations properly, thus resulting in a large portion of the cars coming too close to the cars near them. However, there is an eternal trade-off between the parameters and the metrics: a large damping force can result in slower convergence to steady state, as the damping force opposes any controller actions taken by the

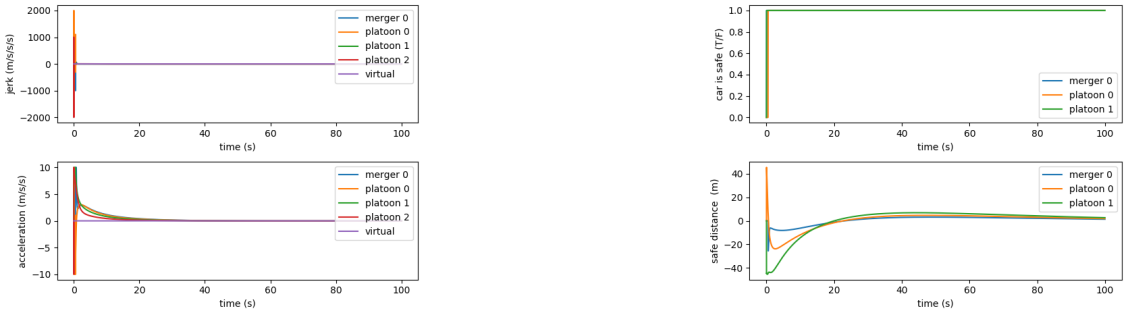


FIGURE 11: A simulation run over 100 seconds with $K = 20$, $D = 1000$, and starting positions for the platoon of [12, 22, 32], velocities of [50, 40, 50], and initial acceleration of 0. The virtual lead car starts at position $x = 32$ and travels with speed $70m/s$ for the duration of the run. The jerk has some very large peaks, due to the discontinuous nature of the error once a merger enters the platoon.

spring. A large spring force can result in high jerks due to the sudden accelerations but also results in fast error convergence and thus also velocity convergence.

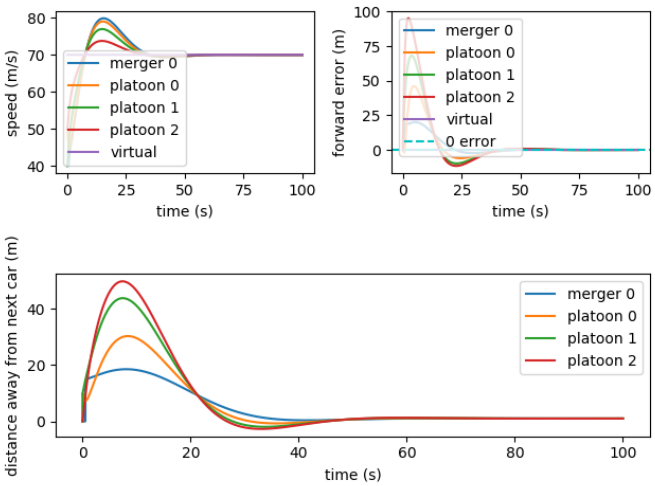


FIGURE 12: A simulation run over 100 seconds with $K = 200$, $D = 1000$, and starting positions for the platoon of [12, 22, 32], velocities of [50, 40, 50], and initial acceleration of 0. The virtual lead car starts at position $x = 32$ and travels with speed $70m/s$ for the duration of the run. The merging car starts with at position $x = 2$ with a horizontal velocity of $40m/s$ merges at $t = 0.42s$. It can be seen that with a higher K, the error converges more quickly. However, still after 25 seconds, most cars have a speed roughly around the desired speed, which is that of the virtual lead car. It is interesting to note that some cars have a negative distance between them and the car in front of them, this means that there was a collision between the cars

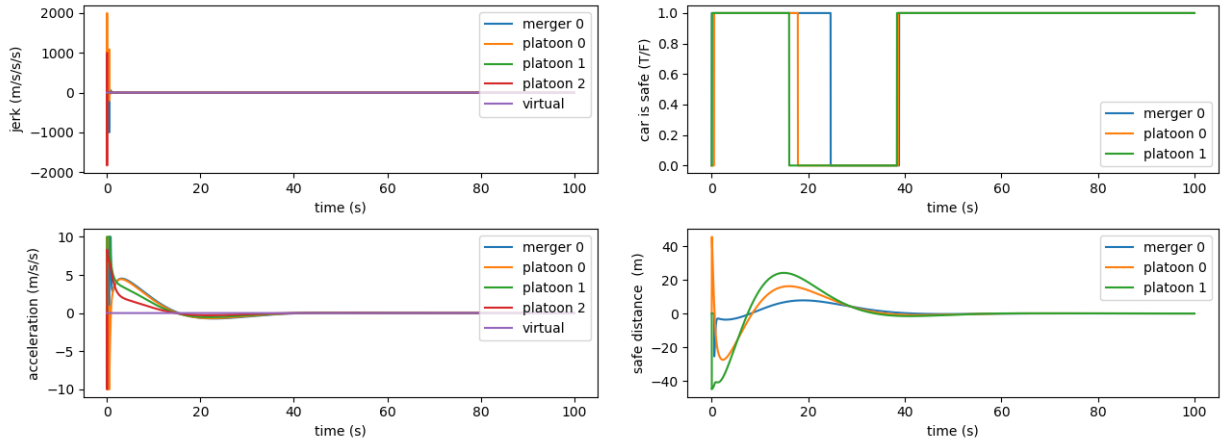


FIGURE 13: A simulation run over 100 seconds with $K = 200$, $D = 1000$, and starting positions for the platoon of [12, 22, 32], velocities of [50, 40, 50], and initial acceleration of 0. The virtual lead car starts at position $x = 32$ and travels with speed $70m/s$ for the duration of the run. As explained in figure 12, the distance between cars is sometimes negative. This is also reflected in the safety metric, which becomes false for about 20 seconds for all cars. The jerk again has some very large peaks, due to the discontinuity produced in the acceleration once a merger enters the platoon.

4.2 One-sided Spring Damper control

4.2.1 Platoon Dynamics

In this control method, the only difference is that we only consider the error with the car in front to be an input to the car being controlled. This eliminates the problem of double sided springs when contributions from both side are equal and opposite, and thus the car does not move. An obvious downside is that cars do not look behind and therefore the assumption that each car will converge locally is very important. The cars are controlled according to the following control law:

$$\mathrm{F}_{\text{engine, 1}} = k_1 e^{(1)} + d_1 e^{(1)} \tag{27}$$

4.2.2 Merging car dynamics one sided controller

Consider a platoon P, and a group of mergers M. Let $c_m^{(j)} \in M$. The platoon observes a merging car when it is within a distance d_{id}. Now suppose the two cars closest to the merger are $c_p^{(i)}$ and $c_p^{(i+1)}$. Once the merging car is within a distance d_{merge}, these cars are selected by the controller, and the spring and damper between $c_p^{(i)}$ and $c_p^{(i+1)}$ is removed. A spring and damper is reattached between $c_p^{(i)}$ and $c_m^{(j)}$, but this time, no spring nor damper between $c_m^{(j)}$ and $c_p^{(i+1)}$. Once these springs and dampers have been reattached, the merging car is effectively part of P, and becomes $c_p^{(i+1)}$. All other cars in the platoon after $c_p^{(i+1)}$ have their index shifted by 1, i.e. $c_p^{(i+1)} \longrightarrow c_p^{(i+2)}$.

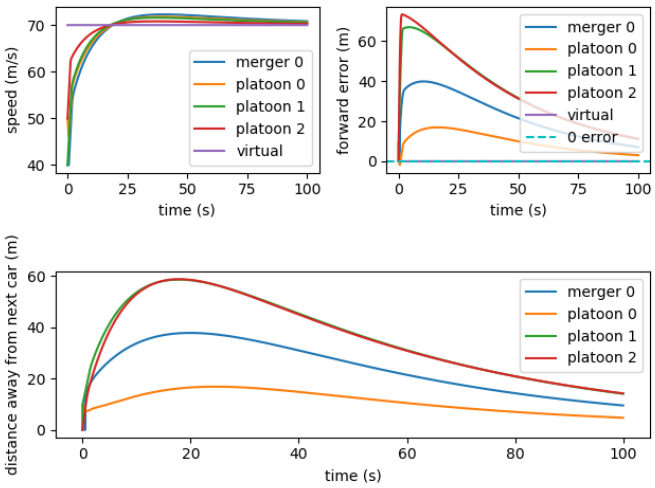


FIGURE 14: A simulation run over 100 seconds with $K = 20$, $D = 1000$, and starting positions for the platoon of [12, 22, 32], velocities of [50, 40, 50], and initial acceleration of 0. The virtual lead car starts at position $x = 32$ and travels with speed $70m/s$ for the duration of the run. The merging car starts with at position $x = 2$ with a horizontal velocity of $40m/s$ merges at $t = 0.42s$. It can be seen again be seen that for when K is much smaller than the mass of the car, the controller converges very slowly. However, after 25 seconds, most cars have a speed roughly around the desired speed, which is that of the virtual lead car.

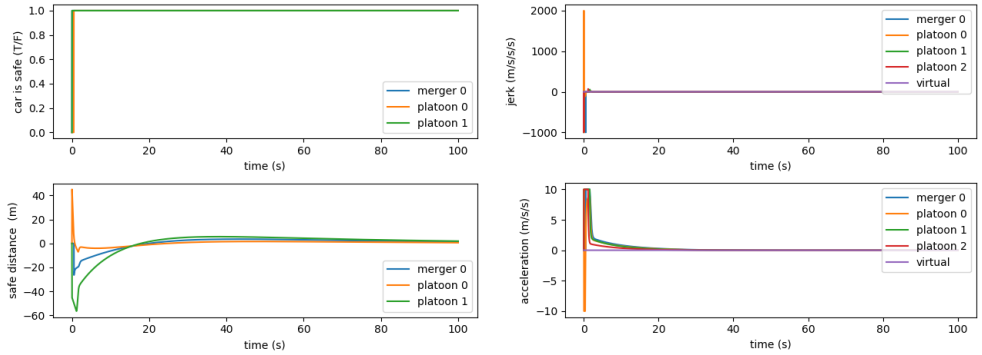


FIGURE 15: A simulation run over 100 seconds with $K = 200$, $D = 1000$, and starting positions for the platoon of [12, 22, 32], velocities of [50, 40, 50], and initial acceleration of 0. The virtual lead car starts at position $x = 32$ and travels with speed $70m/s$ for the duration of the run. The jerk has some very large peaks, due to the discontinuous nature of the error once a merger enters the platoon.

4.3 Dealing with friction

Though not implemented in the simulation, we would like to provide some solutions for dealing with air friction.

In the models before, we have ignored the effect of the force of friction F_f on the cars. However, the frictional force plays an important role when the car is driving at high speeds. This is because the force is proportional to the square of the car's velocity. The net force

analysis of the car results in

$$F_{net} = F_{engine} - F_f. \tag{28}$$

The friction force is not compensated for by the spring and the damper. Especially at higher speeds, this can result in a large steady state error, since the engine force will be proportional to the error and the error derivative, but the friction force might be large enough to counteract both of these contributions. Therefore, the controller will not move closer to the equilibrium, despite a non-zero error. This steady state error can be remedied through the use of feedforward control or PID control.

4.3.1 Feedforward Control

In feedforward control, the controller performs some control action based on the observed system state, rather than the error. In our case, the controller estimates the speed at which the car is moving, and uses this to compensate for what it believes the frictional force is. This can be seen in the form of a block scheme:

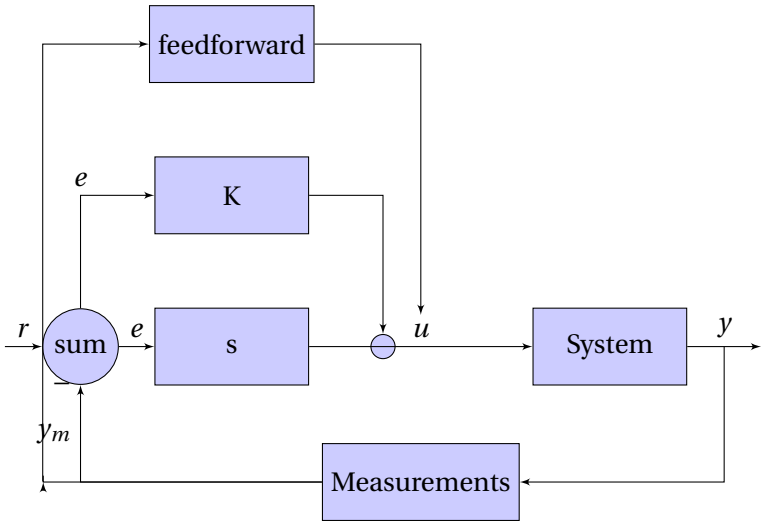


FIGURE 16: Feedforward control block diagram. The bottom arrow goes from Measurements to feedforward directly and does not interact with r

A disadvantage of feedforward control is that an accurate model of the system is necessary. This is because if the estimate of the velocity is inaccurate, the steady state error might not decrease much, as there can still be a nonzero net force making changes on the car: if the estimated velocity is much greater than the actual velocity, then the net force will be positive and the car will move even though the error is 0.

4.3.2 PID

Another way of eliminating the steady state error due to the friction force is by introducing an integrator. This integrator adds up all the errors over time. Compared to feed-forward control, adding an integrator provides several advantages. Firstly, it is model independent: whether the friction force or some other force is causing the error to stop reaching steady state does not matter for the integrator. It simply adds up the error contribution over time and applies this is a force. This means that we do not need to know the exact friction force,

as eventually, the integrator will simply compensate the friction force. Feed-forward requires some knowledge of the velocity, as well as the assumption that we modelled the air friction well enough.

4.4 Choosing K and D

As has become evident throughout this paper, the choice of K and D is anything but trivial. Rather, it is highly situational, and choices must be made as to what important performance indicators are for the controller. Therefore, we propose the following methods:

4.4.1 Model Predictive Control

In model predictive control, we look forward in time for some steps to determine what the control input should be to achieve the desired goal.
We have implemented this by running the simulation virtually for a few iterations, parallel to the "true" simulation, in order to obtain error convergence terms for the future steps. We use this to choose our current control input.
The cost function can be designed at will. We minimize the cost function using Gekko[4] in the following way[3]:

$$min \sum_{i=1}^{n} (x_i)^2 + \sum_{i=1}^{n} (\dot{e}_i)^2. \tag{29}$$

Here e_i is the error of the prediction step with a certain k and d. We minimize over k and d, trying to find the best ones.

Another interesting cost function could be the jerk. As mentioned before, we would like to minimize jerk as it is unpleasant for the passengers in the vehicle. Therefore we could design a cost function such as the one below:

$$\min \sum_{i=1}^{n} (\dddot{x}_i)^2. \tag{30}$$

4.5 Genetic Algorithm

The control law (23) can be tuned to make it more appealing for certain situations. For example, in the above example, it is desirable to reduce jerk. With the Genetic Algorithm, suitable parameters for the spring, damper, and integrator could be found that make the car adhere to certain preferences the passenger might have. Such a parameter could be found as follows:

Generate an initial population of genes (parameters K, D, I). Then determine their fitness according to some cost function, e.g. the jerk or the error convergence. Let the fittest genes reproduce by creating offspring that take parts of their genes from parent A and the other part from parent B. Introduce random mutations to keep the genetic diversity high Repeat

In the case of PID controller optimization, the PID parameters are the genes. We have given them a cost function based on what an entire simulation run with a certain gene would look like (a gene is a group of PID parameters, one for each car in the platoon). PID parameters

[3] Implemented with the help of an example on the APMonitor website: http://apmonitor.com/wiki/index.php/Main/Control

can be tuned using a genetic algorithm [14], however it remains to be investigated whetheher this is applicable for platoon control.

Using this algorithm, we might find an interesting match for the parameters K, D and I. However, the algorithm would need to be re-run once we have a different set of initial conditions for the platoon and the mergers, since we optimize the parameters given some initial conditions which define the system.

As it is currently implemented, the algorithm tunes the one-sided springs according to a cost function of the total squared error. Mergers are not taken into account explicitly, as a merging situation can be abstracted to a platoon configuration with different initial conditions. Therefore, if a merge occurs, the parameters for the system following the merge could be chosen based on a genetic algorithm of a platoon with these initial conditions.

The current algorithm[4] produces, after 600 iterations, the following parameters:

This could be because the platoon 0 should follow the lead car closely, but the lead car itself should not follow the virtual car too quickly as this results in large jerks for the entire platoon. As mentioned before, a large portion of the jerk seen in the figure is due to the simulation discretization, thus making results hader to analyze. However, this motivation for the genetic algorithm to choose those parameters over others does seem plausible.

The cost function with jerk:

$$\text{fitness} = \sum_i \dot{e}^2 + \sum_i e^2 + 10 \sum_i \dddot{x}. \tag{31}$$

We run the algorithm for 2 cars at positions 10 and 20, and no mergers, with both cars having initial speeds and accelerations of 70 and 2, respectively. The cars start The parameters when jerk is not part of the cost function, obtained after 600 iterations of the algorithm with a pop size of 600 and a mutation rate of 0.3:

Platoon car 0: $K = [95.65688048029267, D = 66.13115605728966]$
Platoon car 1: $K = [98.49593492413787, D = 48.71235392721249]$

and when it is included in the cost function:

Platoon car 0: $K = [96.89398872105822, D = 91.90203810816485]$
Platoon car 1: $K = [93.23198303023912, D = 69.86931016275393]$

This seems a moderately plausible result since the damping causes less rapid convergence which in turn causes less jerk. However, the parameters are still very high, so either the algorithm converged prematurely, or this method is not suitable for this problem. Another possibility is that the jerk cannot be minimized well due to the fact that fast error convergence and high jerk go hand in hand: fast error convergence requires a large K, which in turn produces high jerk at the start of the simulation. It is therefore possible that the cost function was poorly designed.

[4] Implemented with the help of an example on the inspyred website: https://pythonhosted.org/inspyred/tutorial.html

5 Scenarios and Results

In order to compare the different control strategies outlined in this paper, some merging scenarios are constructed. These will be simulated without friction as the emphasis is on the parameters K and D, and with a time-step of 0.01. The virtual lead car always has a speed of $50m/s$ and starts at the position of the lead car. Additionally, all cars calculate a safe distance away from the next car based on the metric introduced before, and then add $20m$ to this distance, which becomes their reference distance. This is done to reflect the fact that reaction time is ignored in the distance formula, and thus in a real scenario, likely some extra distance would be kept between the cars.

5.1 Scenario 1: Merging car Exceeds Platoon Speed

Consider a platoon P with $|P| = 3$, and mergers with $|M| = 1$. In this scenario it will be investigated how the controller performs when the platoon is initialized as follows: $x_p^{(1)}(0) = 10$, $x_p^{(2)}(0) = 40$, $x_p^{(3)}(0) = 70$, and $x_m^{(1)}(0) = 25$. For the merging vehicle, we also have its initial y position given by $y_m^{(1)}(0) = 5$, and $v_{m,x}^{(1)}(0) = 60$, $v_{m,y}^{(1)}(0) = -0.1$. Additionally, for all the cars in the platoon, $v_{y,p}^{(1)}(0) = 0$ as they are assumed to move only laterally.

	$c_p^{(1)}$	$c_p^{(2)}$	$c_p^{(3)}$	$c_m^{(1)}$
$x(0)$	10	40	70	25
$v(0)$	50	50	50	60
$a(0)$	0	0	0	0

TABLE 1: The values in this table (and the 2 below it) are still under construction

This scenario was chosen because it is important to analyze what the controller performance is when the speed of the merger is faster than that of the platoon, since this speed needs to be controlled by the controller to become the platoon speed. We see that this is indeed the case, but only after the car slows down a lot, since it merges at the head of the platoon.

5.1.1 One-sided Controller

We simulate this with the choice of $K = 200$, $D = 1000$. In this simulation we observe that the platoon car can only merge after 35.01 seconds, once it has arrived on the highway. When it does, it is very far ahead of the platoon, and becomes the new lead car. This also explains why the distance with next suddenly shoots down into the negatives: the merging car is ahead of the virtual lead car and so attempts to slow down. Whether such behaviour is desirable must be analyzed, as it can lead to very volatile behaviour. It might be better to make it slow down only a little bit until the virtual car catches up to it, as slowing down on the highway is dangerous. Included in the legend is also who is in front of the current car, to verify that the merger does indeed merge in front of the virtual car.

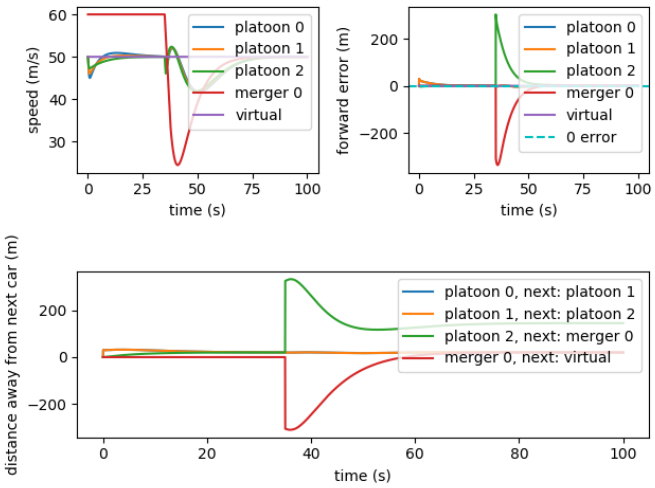
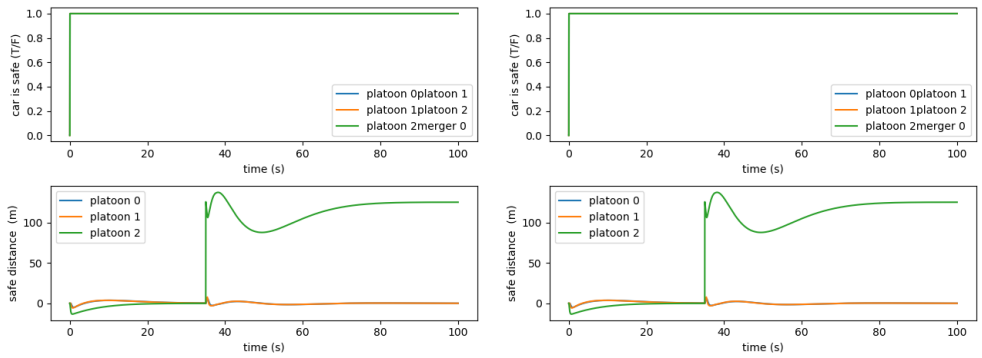


FIGURE 17: A simulation of scenario 1 with $K = 200$ and $D = 1000$. The merge occurs around $t = 35$, at which point the merging car is ahead of the virtual lead vehicle, producing large errors. As can be seen in the figure below, the cars are safe during the entirety of the simulation. The merger also has a negative distance between it and the next car, which normally would imply a collision, however, in this case it just means that the virtual car is behind it, so there are no safety issues. This is also illustrated in the legend, to verify that it is indeed the virtual lead car that is in front of merger 0.



5.1.2 Two-sided Controller

We simulate this with the choice of $K = 200$, $D = 1000$. Here, the car also merges after 35.01 seconds. The scenario is very similar to the previous one, but the two sided controller appears less safe. This can be seen in figure **??**. The distance between cars makes a small dip below 0, which is dangerous, and is reflected in the safety plot.

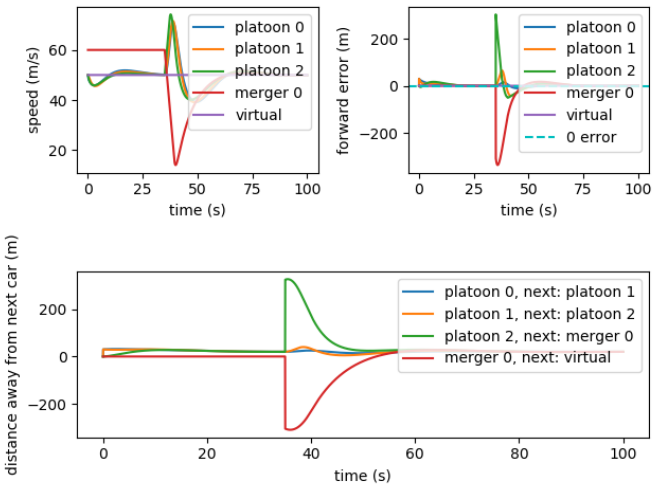
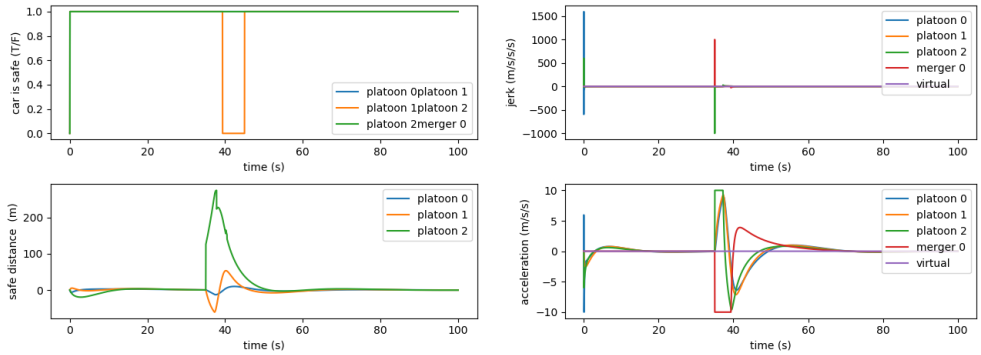


FIGURE 18: For the two-sided controller, a similar error convergence is seen. However, platoon 0 is pushed a bit closer to the car behind it, making the control method a bit less safe.



5.2 Results

As can be observed throughout this paper, the one-sided spring damper controller matches the two-sided spring damper controller in almost all aspects, and outperforms it in safety. Though more simulations must be ran to say anything meaningful about the comparison of the two controllers, from the cases investigated in this paper it would appear that the one-sided controller is therefore better.

The increased safety of the one-sided spring damper controller could be explained by the fact that it always moves if the cars are too close, whereas for the double sided spring damper controller, it is possible that when both springs are pushed in equally hard on both sides, no net engine force is produced. This could explain why the two-sided spring damper controller is less safe.

Therefore, the one-sided spring damper controller seems to be the most feasible controller for platoon control. However, as discussed in the appendix, when two mergers enter the platoon at the same time, this controller is also unsafe. It is highly likely that the primary assumption on which this controller operates, namely, that the cars behind it will adjust for it in time, is not true when two mergers come into the platoon. This is because adjustments are made for

2 mergers so cars cannot simply accelerate and decelerate without regard for their predecessors.

6 Discussion and Future Work

The simulation is a discrete representation of a continuous time system, and this introduces errors and inaccuracies. Other methods of discretization, or perhaps even smaller step sizes, could potentially make the simulation more realistic. Investigating trajectory smoothing could also be of merit as this could smoothen the transition of the error signal once a merger merges, which now occurs instantly, creating discontinuous jumps in the error which blow up $\dot{e}$.

Additionally, the simulation could contain some bugs and can in general be improved upon heavily. For example, a 2D simulation could be constructed where cars can also move laterally. Better ways to record acceleration and jerk could also improve the simulation.

Unfortunately, critical damping was not evaluated as in-depth as it could have been for both controllers. Therefore, in future work, this could be an interesting topic.
The genetic algorithm could converge prematurely to a non-optimal solution. This could also be investigated through increasing the maximum iterations or introducing some sort of convergence metric for when the parameters are sufficient.

Although the implementation for the model predictive part of the algorithm seems to work, for some reason using the chosen K and D do not translate correctly from the model prediction to the actual simulation.

For future work, it could be interesting to investigate the stability of the system when the spring and damping constants are time-varying. This already happens in model predictive control but a more rigorous statement about stability could be of interest, as the energy-based Lyapunov function could potentially fail due to increasing system energy when the spring constant is increased. However, online trajectory optimization is rather time-consuming and cars probably do not have the hardware to do this, so potentially the optimal solutions can be pre-computed offline and then looked for online, so the car gets an input based on its current state, and this input is based on an offline optimization of this state.

Also, a coulomb force could be investigated that makes the cars act like opposing charges, repelling them heavily when they get too close to each other. Some sources already use a Newton force in their models, so inspiration may be drawn from them[6].

It could also be of interest to investigate the string stability of the both the two-sided platoon and one-sided platoon. Some work on this has been done by [7]
Different cost functions for the genetic algorithm could be investigated on their performance. For example, the cost function could increase by a lot if the cars (almost) collide, thereby potentially avoiding genes that cause collisions.

Additionally, the parameters of the genetic algorithm could be investigated. The parameters chosen for crossover and elitism have not been motivated, since it was not investigated how

the results varied with different parameters.

References

[1] Efficient road transport. https://www.acea.be/industry-topics/tag/category/efficient-road-transport. Accessed: 2010-09-30.

[2] Platooning roadmap. https://www.acea.be/uploads/publications/Platooning_roadmap.pdf. Accessed: 2010-09-30.

[3] Tucker Balch and Ronald C Arkin. Behavior-based formation control for multirobot teams. *IEEE transactions on robotics and automation*, 14(6):926–939, 1998.

[4] Logan Beal, Daniel Hill, R Martin, and John Hedengren. Gekko optimization suite. *Processes*, 6(8):106, 2018.

[5] F. Ben. The law of anomalous numbers. *Proc. Am. Philos. Soc.*, 78(4):551–572, 1938.

[6] Jean-Michel Contet, Franck Gechter, Pablo Gruer, and Abderrafiaa Koukam. Application of reactive multiagent system to linear vehicle platoon. In *19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007)*, volume 2, pages 67–70. IEEE, 2007.

[7] Jean-Michel Contet, Franck Gechter, Pablo Gruer, and Abderrafiaa Koukam. Bending virtual spring-damper: A solution to improve local platoon control. In Gabrielle Allen, Jarosław Nabrzyski, Edward Seidel, Geert Dick van Albada, Jack Dongarra, and Peter M. A. Sloot, editors, *Computational Science – ICCS 2009*, pages 601–610, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[8] Jean-Michel Contet, Franck Gechter, Pablo Gruer, and Abderrafiaa Koukam. Reactive multi-agent approach to local platoon control: stability analysis and experimentations. *IJISTA*, 10(3):231–249, 2011.

[9] Soumya Dasgupta, Varunkumar Raghuraman, Apratim Choudhury, T Nagacharan Teja, and Justin Dauwels. Merging and splitting maneuver of platoons by means of a novel pid controller. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8. IEEE, 2017.

[10] Sunan Huang and Wei Ren. Use of neural fuzzy networks with mixed genetic/gradient algorithm in automated vehicle control. *IEEE Transactions on Industrial Electronics*, 46(6):1090–1102, 1999.

[11] Vesna Antoska Knights, Zoran Gacovski, Stojce Deskovski, and Olivera Petrovska. Guidance and control system for platoon of autonomous mobile robots. *Journal of Electrical Engineering*, 6:281–288, 2018.

[12] C Kreuzen. Cooperative adaptive cruise control: using information from multiple predecessors in combination with mpc. 2012.

[13] EL-Zaher Madeleine, Baudouin Dafflon, Franck Gechter, and Jean-Michel Contet. Vehicle platoon control with multi-configuration ability. *Procedia Computer Science*, 9:1503–1512, 2012.

[14] Yasue Mitsukura, Toru Yamamoto, and Masahiro Kaneda. A design of self-tuning pid controllers using a genetic algorithm. In *Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251)*, volume 2, pages 1361–1365. IEEE, 1999.

[15] Brian Porter and AH Jones. Genetic tuning of digital pid controllers. *Electronics letters*, 28(9):843–844, 1992.

[16] Albert Rizaldi, Fabian Immler, and Matthias Althoff. A formally verified checker of the safe distance traffic rules for autonomous vehicles. In *NASA Formal Methods Symposium*, pages 175–190. Springer, 2016.

[17] Aakash Soni and Huosheng Hu. Formation control for a fleet of autonomous ground vehicles: A survey. *Robotics*, 7(4):67, 2018.

[18] Lars Svensson and Jenny Eriksson. Tuning for ride quality in autonomous vehicle: Application to linear quadratic path planning algorithm, 2015.

[19] Yuan-Ying Wang and Hung-Yu Wei. Safe driving capacity of autonomous vehicles. *arXiv preprint arXiv:1806.01777*, 2018.

A Appendix: Some additional platooning scenarios

A.1 Scenario 0: Normal Platoon Stabilization

Consider a platoon P with $|P| = 3$, and mergers with $|M| = 0$. In this scenario it will be investigated how the controller performs when the platoon is initialized as shown in the table. This scenario allows us to evaluate the behaviour of the platoon when it is stabilizing from small errors. This should go smoothly, and we should be able to see a clear pattern between the cars' velocities. We investigate this scenario with both controllers, for $K = 200$ and $D = 1000$.

	$c_p^{(1)}$	$c_p^{(2)}$	$c_p^{(3)}$
$x(0)$	10	40	70
$v(0)$	50	50	50
$a(0)$	0	0	0

A.1.1 One-sided

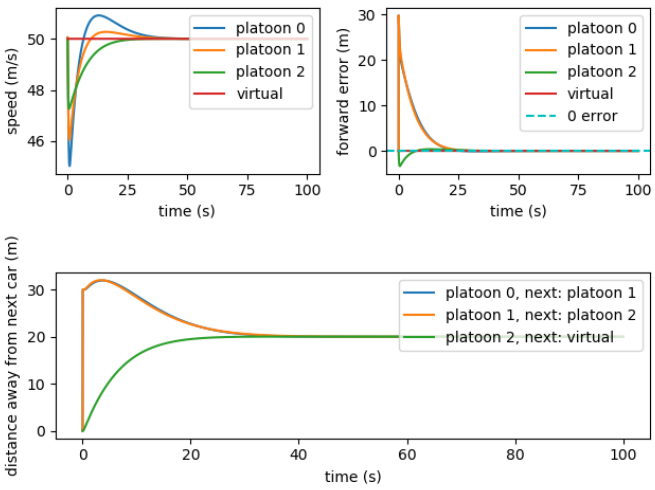


FIGURE 19: One-sided controller error convergence for scenario 0. It can be seen that the controller converges at a reasonable rate, comparable to the two-sided controller

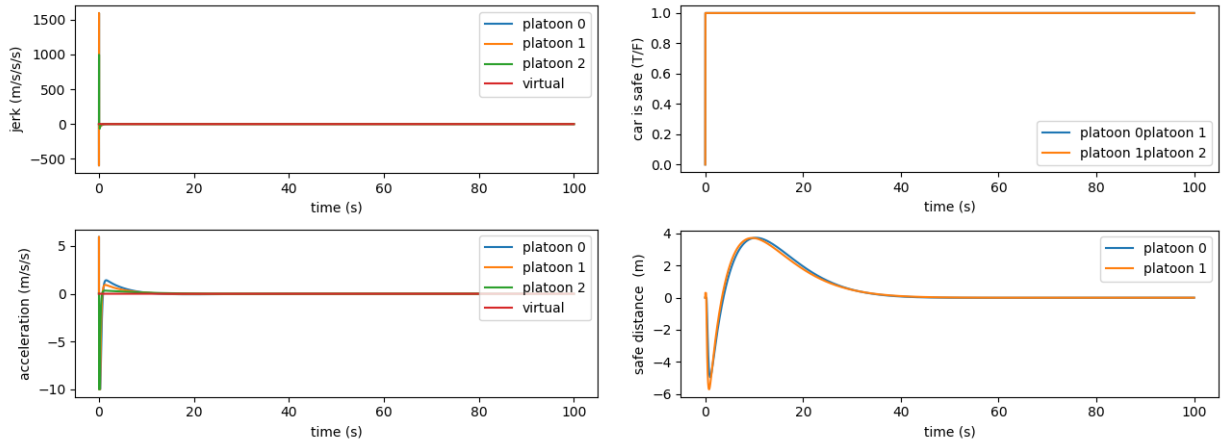


FIGURE 20: One-sided jerks and safety for scenario 0. The cars are immediately safe, as can be seen from the safety plot.

A.1.2 Two-sided

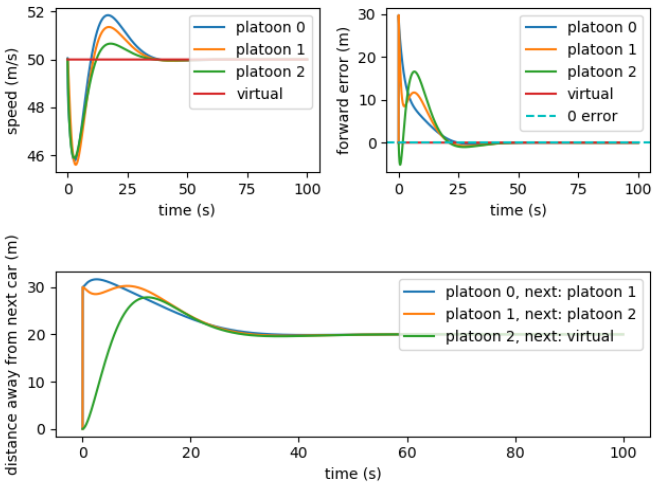


FIGURE 21: Two-sided error convergence for scenario 0

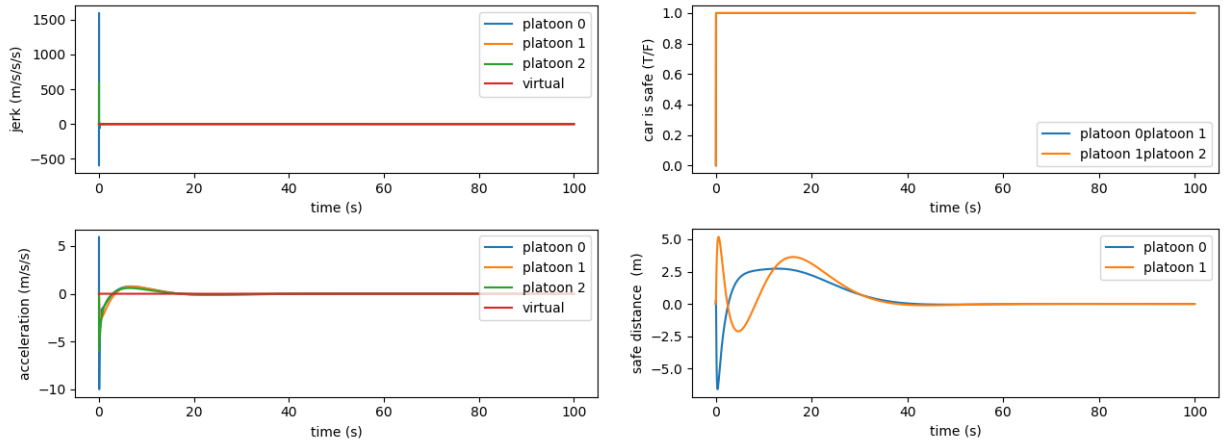


FIGURE 22: The two-sided control converges a bit more smoothly than the one-sided controller for scenario 0

A.1.3 Comparison

In scenario 0, we investigate controller performance for stabilizing a system. As can be seen from the figures in the previous subsections, the controller is able to do this effectively both for double and single sided methods. Additionally, the controller is very safe.

A.2 Scenario 2: 2 Merging cars that both exceed platoon speed

Consider a platoon P with $|P| = 3$, and mergers with $|M| = 2$. In this scenario it will be investigated how the controller performs when the platoon is initialized as follows: $x_p^{(1)}(0) = 0$, $x_p^{(2)}(0) = 2$, $x_p^{(3)}(0) = 4$, and $x_m^{(1)}(0) = 0$, $x_m^{(1)}(0) = 0$. Additionally, both mergers start at $y = 0$ and have y velocities of -0.1, and accelerations of 0 in both x and y. We investigate this scenario with both controllers, for $K = 200$ and $D = 1000$.

	$c_p^{(1)}$	$c_p^{(2)}$	$c_p^{(3)}$	$c_m^{(1)}$	$c_m^{(2)}$
$x(0)$	10	40	70	25	45
$v(0)$	50	50	50	60	60
$a(0)$	0	0	0	0	0

This scenario allows us to evaluate a similar situation to scenario 2, only when the merging car is slower. It is expected that this scenario will be more difficult for the controllers to handle, as space needs to be created in 2 different locations in the platoon in order to let the mergers merge.

A.2.1 One-sided

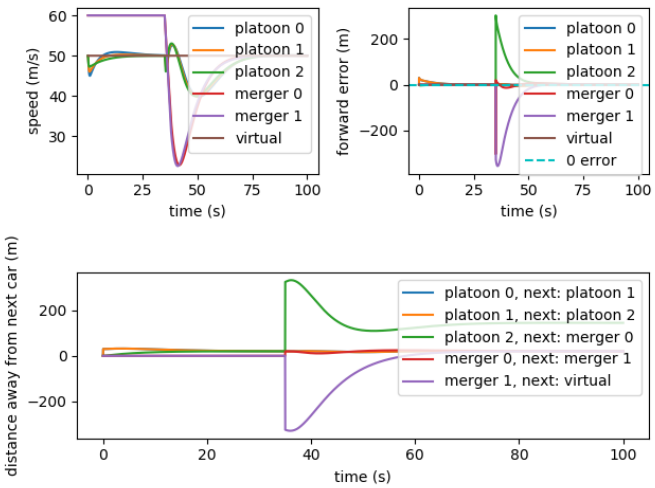


FIGURE 23: One-sided controller error convergence. It can be seen that the controller converges at a reasonable rate, comparable to the two-sided controller

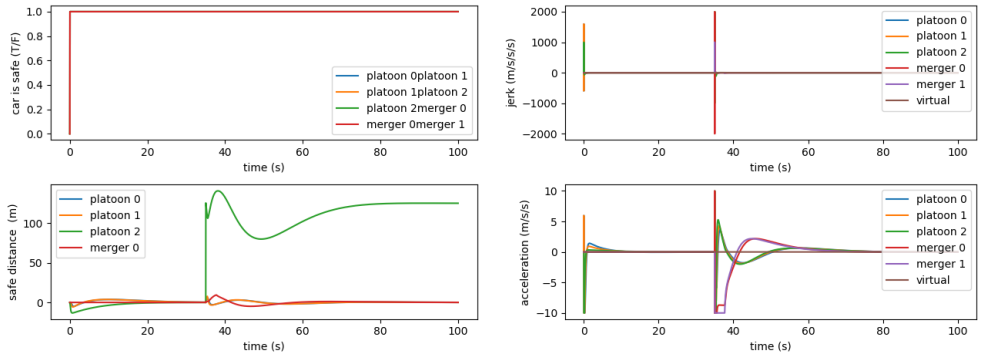


FIGURE 24: one-sided 2

A.2.2 Two-sided

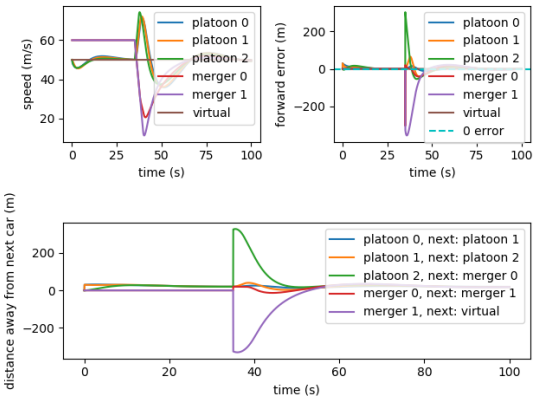


FIGURE 25: two-sided

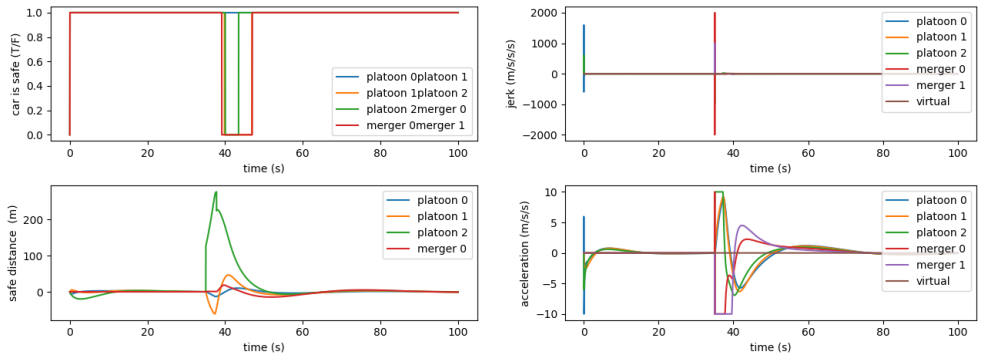


FIGURE 26: For the two-sided controller, again, the issue appears to be safety. The scenario is otherwise much like the previous one, only with 1 extra car.

A.2.3 Comparison

As we have seen before, the two-sided controller generally performs worse in safety than the one-sided controller. However, other than this, there are not many differences. The platoon seems to handle both mergers well, and the system is never unsafe. A large increase in acceleration and jerk that is characteristic of the merge maneuver can be seen around $t = 35$

A.3 Scenario 3: Two mergers in different places

Consider a platoon P with $|P| = 3$, and mergers with $|M| = 1$. In this scenario it will be investigated how the controller performs when the platoon is initialized as follows: $x_p^{(1)}(0) = 0$, $x_p^{(2)}(0) = 2$, $x_p^{(3)}(0) = 4$, and $x_m^{(1)}(0) = 0$. Both mergers start at $y = 5$ with a y velocity of -1 and a y acceleration of 0. Both mergers also have velocities of $66m/s$ and we also investigate this scenario with both controllers, for $K = 200$ and $D = 1000$.

	$c_p^{(1)}$	$c_p^{(2)}$	$c_p^{(3)}$	$c_m^{(1)}$	$c_m^{(2)}$
$x(0)$	10	40	70	0	45
$v(0)$	50	50	50	50	50
$a(0)$	0	0	0	0	0

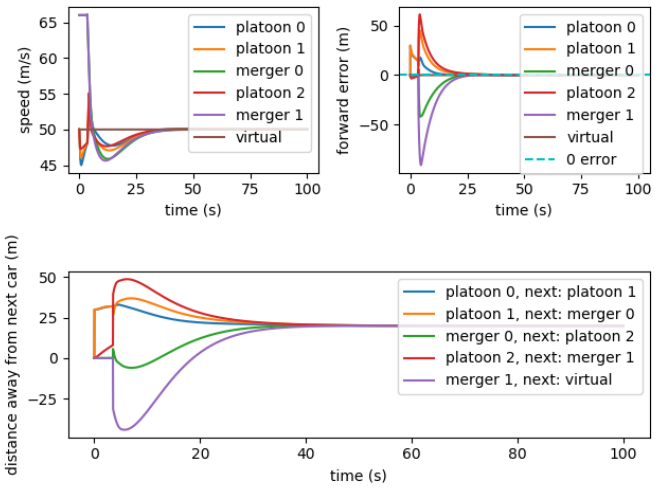
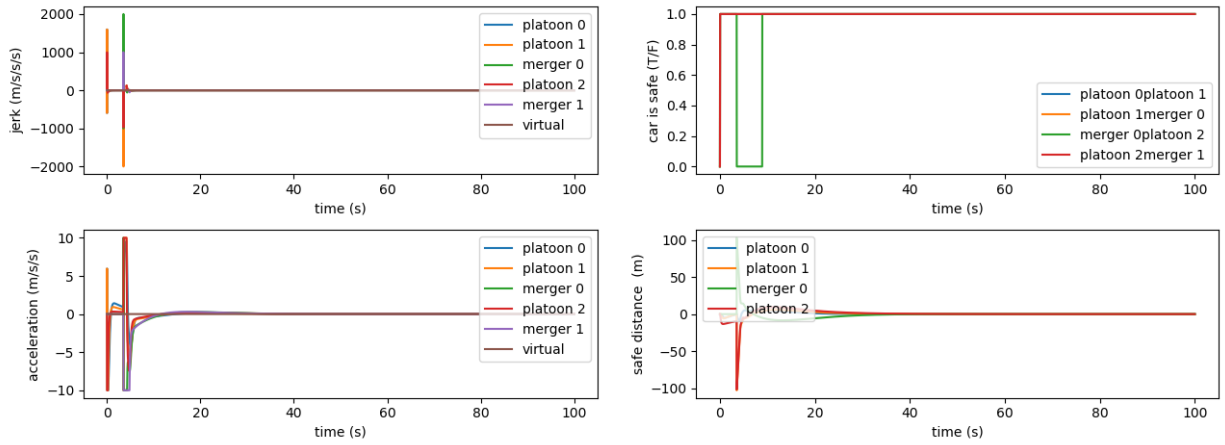


FIGURE 27: A simulation of scenario 3 with $K = 200$ and $D = 1000$. As can be seen in the figure below, the cars are not safe during the entirety of the simulation: a small window of time occurs just after the merge where one of the cars some of the cars are collide. In the ledger for the distance between cars is again shown who the next car is for each car, and from this it can also be observed that the mergers do indeed merge in different locations within the platoon.



A.3.1 Two-sided

We simulate this with the choice of $K = 200$, $D = 1000$. The scenario is very similar to the previous one, but now both controllers are unsafe.

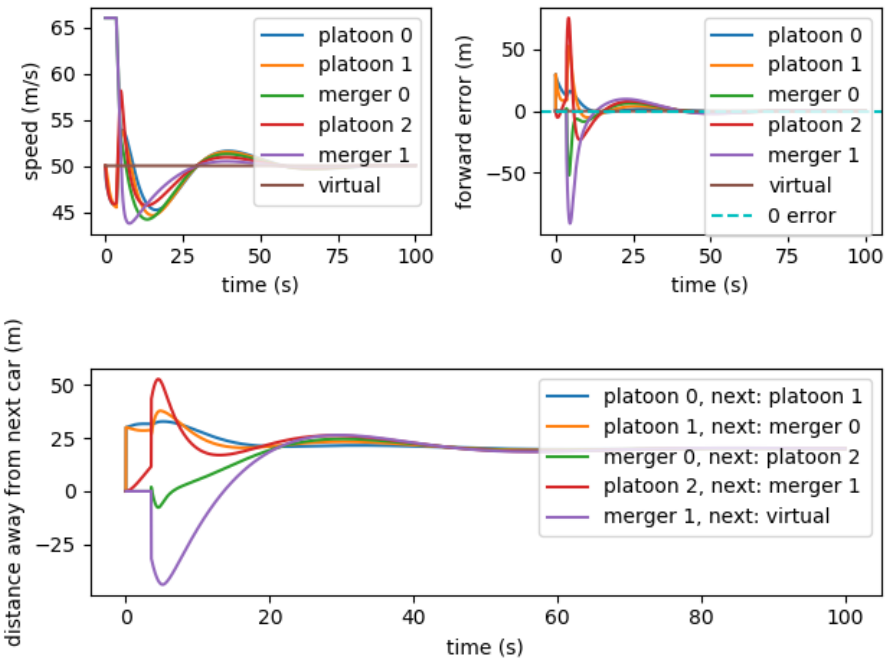


FIGURE 28: Two-sided error convergence for scenario 3

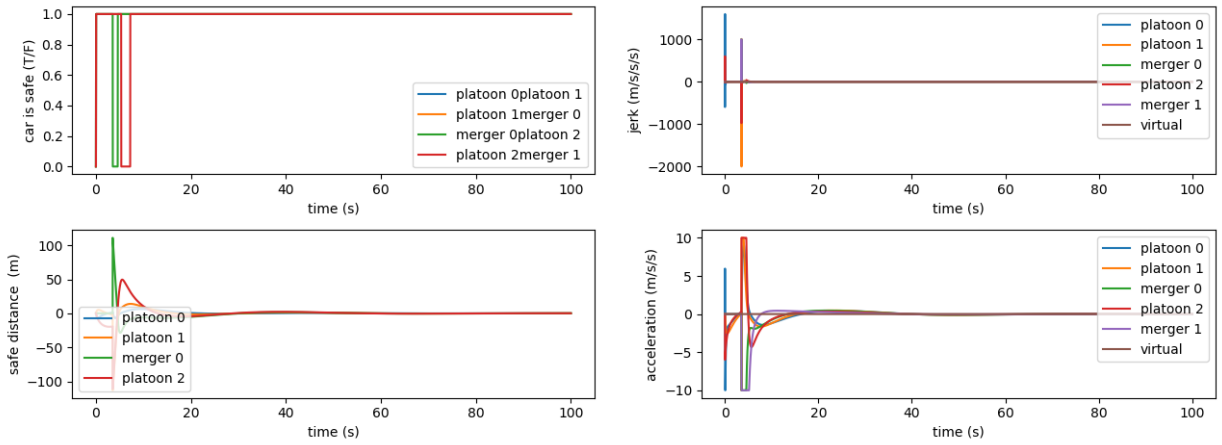


FIGURE 29: Two-sided error convergence for scenario 3

A.3.2 Comparison

In this scenario, both controllers were unsafe. Error convergences are rather similar, but the fact that both are unsafe warrants further investigation into parameter choices.
A possible downside of the one-sided controller is that it only accounts for one car, so if a car is merging in front of and behind the same car, in the one-sided controller this could potentially lead to collisions.

B Appendix: safety check

```
s_stop_e = car.x - (car.xspeed**2)/(2*car.max_decel)
s_stop_o = car.next.x - (car.next.xspeed**2)/(2*car.max_decel)

t_stop_e = -(car.xspeed)/car.max_decel
t_stop_o = -(car.next.xspeed)/car.next.max_decel

a_o = car.next.max_decel
a_e = car.max_decel
v_o = car.next.xspeed
v_e = car.xspeed


precondition = (car.x <= s_stop_e) and (car.max_decel > car.next.max_decel)
and (car.xspeed < car.next.xspeed) and (t_stop_e < t_stop_o)

safe_distance_1 = -(v_e**2)/(2*a_e)
safe_distance_2 = (v_o**2)/(2*a_o) - (v_e**2)/(2*a_e)

dist_cars = car.next.x - car.x
if dist_cars < 0:
    if car.next.name != "virtual":
        print("collision!")
    safe = False
if dist_cars > safe_distance_1:
    safe = True
    safe_distance = safe_distance_1
elif precondition:
    # safe_distance_3 = ((v_o - v_e) ** 2) / (2 * (a_o - a_e))
    # this distance is not used, because it is infinite if a_0 = a_e
    safe = dist_cars > safe_distance_1
    safe_distance = safe_distance_1

else:
    safe = dist_cars > safe_distance_2
    safe_distance = safe_distance_2
```