# Vehicle Platooning with Distributed EKF and NMPC

Marko Mizdrak
Department of Industrial Engineering, University of Trento

**Abstract**

*Cooperative driving systems, such as vehicle platooning, offer tremendous potential for improving road safety, traffic flow, and fuel efficiency. This paper demonstrates a simple and effective method for vehicle platooning, which improves state estimates and predictions through vehicle-to-vehicle communication of state estimates (obtained by combining several sensor measurements) and next controls. Vehicles in the platoon solve an optimal control problem over a set time horizon to maintain a time gap behind the vehicle ahead.*

*A comparison is given with reactive control systems, both with and without communication. It is shown that not only does the communication improve the estimation abilities of the vehicles but also how basing controls on the predictions of the future behaviour of the platoon can lead to more stable behaviour. The results demonstrate the significance of incorporating communication and optimal control to enhance the performance of vehicle platooning.*

## Introduction

Over the years, vehicle developers have been actively pursuing advancements in the efficiency, safety, and autonomy of vehicles. While improvements can be made at the individual car level, exploring the potential of a vehicle system opens up additional possibilities. Similarly to the natural phenomenon of geese flying in formation to reduce drag and increase flight endurance, a similar concept can be applied to vehicles through the formation of platoons.

### Energy Savings

A recent review summarises how the platooning of road vehicles can reduce total energy expenditure and therefore reduce emissions. [1]. Due to the complexities of modeling the exact aerodynamic effects and power characteristics of engines, it is difficult to calculate the exact optimum distances for any given vehicle platoon. However, promising experiments have shown good energy savings in real-world tests. [2]

### Traffic Flow

Another benefit platooning can bring is in improving traffic flow [3]. While some traffic jams are caused by a road network's limitations on throughput, many are in the form of ghost traffic jams. In ghost traffic jams, small braking actions of one vehicle can be magnified by the reactions of vehicles following which can often lead to dense slow traffic. Formally this is the property of string-stability. However, it has been shown string-stability is not enough to guarantee collision-free platooning.[4]

### From Adaptive Cruise Control Systems to Cooperation

The evolution toward vehicle platooning began with Cruise Control (CC). CC systems were first introduced to maintain speeds for drivers to be able to reduce the risk of speeding and drive efficiently without constantly looking at their speedometers. These systems simply attempt to maintain a user-given speed. Adaptive Cruise Control systems (ACC) were first introduced in the late 1990s and were the first steps in the direction of autonomous road vehicles.[5] These relied on some form of range sensor and rather than just maintain a certain speed, they can detect vehicles ahead and slow down from the user's given speed to avoid collisions. However, without communication with other vehicles, these systems exhibit the same string-instability problems as fully human-driven cars.

This is why recent research has focused on Cooperative Adaptive Cruise Control Systems (CACC) which utilise communication between vehicles (V2V)

to overcome the challenges of simply reactive systems.

## Problem Formulation

The fundamental problem we are addressing in this research pertains to Cooperative Adaptive Cruise Control (CACC) systems in vehicular networks. We aim to investigate the process by which each vehicle within the system utilizes sensor data and inter-vehicle communication to accurately estimate the position of other vehicles (most importantly the vehicle ahead). This estimation is then used for calculating appropriate control signals to maintain a predetermined time-based distance to the preceding vehicle.

Thus each vehicle is aiming to minimise the error of the estimates $\hat{\mathbf{x}}_i$ of the vehicle states $\mathbf{x}_i$ , for the vehicles $i = 1, ..., N$ in the platoon.

Each vehicle (aside from the lead vehicle) is also attempting to find control values $u$ to maintain a time difference $T_d$ to the vehicle ahead.

$$\min_{u_i(t)} \int (D_i(u_i(t), \mathbf{x}_i(t)) - T_d v_i)\, dt \tag{1}$$

Where $v_i$ is the velocity of vehicle $i$ and $D_i$ is the distance in the direction of travel between vehicles $i$ and $i-1$.

However, since we control the vehicles at discrete time intervals, more practically the problem becomes:

$$\min_{u_i(t)} \sum (D_i(u_i(t), x_i(t)) - T_d v_i) \tag{2}$$

With $t$ now representing a discrete-time point rather than a continuous variable.

# Adopted Models

## Communication Method

For this paper, the communication between the vehicles is modelled as taking place between adjacent vehicles (though this can easily be extended by editing one parameter in the code). That is, each vehicle communicates one vehicle ahead and one vehicle behind.

The vehicles send two types of messages. Firstly, an estimation message, which sends all of the vehicles estimates of other vehicles. Secondly, a control message sends the next controls that the vehicle is aware of other vehicles intending to make including its own intention.

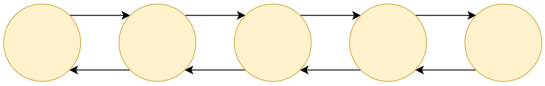

**Figure 1:** *The connectivity of the vehicle platoon.*

The specifics of the medium of communication are not in the scope of this paper - it is simply assumed that some given communication technology would be able to mediate the control signals. In the simulation, the messages are simply passed directly between the vehicles with an option for adding a loss ratio.

## System Model

Each vehicle is modelled as having 4 sensors. A GNSS-like sensor for measuring position, a speedometer-like sensor for measuring speed, a compass-like sensor for measuring heading, and a lidar/radar-like sensor for measuring the relative position to other vehicles.

These are kept as abstract since many different sensors could be used to give the same effect in a real vehicle system but since the test was not done on hardware the intricacies of any given sensor are not considered.

The vehicle state is modelled with: $\mathbf{x}_t = (x_t, y_t, \alpha_t, v_t)$ corresponding to a 2-dimensional position, a heading angle and a speed (in the direction of vehicle motion). The vehicle's motion is modelled with a noisy, discrete-time, single-track model.

$$v_{t+1} = v_t + (u_{at} + \epsilon_a)\Delta t \tag{3}$$

$$\alpha_{t+1} = \alpha_t + v_t \tan(u_{st} + \epsilon_s)\Delta t / L \tag{4}$$

$$x_{t+1} = x_t + v_t \cos(\alpha_t)\Delta t \tag{5}$$

$$y_{t+1} = y_t + v_t \sin(\alpha_t)\Delta t \tag{6}$$

Where $u_{at}$ is the acceleration control command at time $t$ and $u_{st}$ is the steering command at time $t$ and

$\epsilon_a, \epsilon_s$ represent the addition of Gaussian noise to the control commands. The wheelbase of the vehicle is represented by $L$.

# Solution

## Extended Kalman Filter

To minimise the estimation errors an Extended Kalman Filter (EKF) is used to fuse sensor data and make predictions. The EKF is a recursively estimates the system by approximating with a first-order Taylor series expansion to linearize the nonlinear functions around the current estimate.

The two stages are prediction and measurement. In the prediction step we propagate the state estimate $\hat{s}_t^-$ by simply of using the system dynamics as described in the single-track model above. In the non-communicative case, the vehicle assumes zero control values. The covariance matrix is updated with:

$$P_t^- = F_t P_{t-1} F_t^T + Q_t \tag{7}$$

Where $F_t$ is the Jacobian with respect to the state $s_t$ of the dynamics evaluated at $u_t, \hat{s}_{t-1}$.

Each measurement updates the state and covariance as follows:

$$\mathbf{z_t} = h(\mathbf{x_t}, \epsilon_t) \tag{8}$$

$$\gamma_t = \mathbf{z_t} - h(\hat{\mathbf{x}}_\mathbf{t}^-, 0) \tag{9}$$

$$S_t = H_t P_t^- H_t^T + R_t \tag{10}$$

$$K_t = P_t^- H_t S_t^{-1} \tag{11}$$

$$\hat{\mathbf{x}}_t = \hat{x}_t^- + K_t \, \gamma_t \tag{12}$$

$$P_t = (I - K_t H_t) P_t^- \tag{13}$$

Where $\mathbf{z_t}$ is the measurement which is result of a function of the state and noise. $H$ is the Jacobian of the function $h()$ evaluated at the current state estimate and $R_t$ is the covariance of the measurement noise.

In this case, the measurement step is performed for every sensor measurement to fuse the information into one estimate between every prediction step.

When the vehicles communicate, their estimates are stored in a buffer and once communication has finished for that timestep, the vehicles process the estimates in their buffers. The communicated estimates are treated similarly to measurements from any other sensor and the same Kalman Filter is used to incorporate these estimates.

### Optimal NMPC

The future states of the vehicles are predicted based on the other vehicle's shared future controls by modeling successive prediction steps with the EKF. These predictions form the set of future states to follow ${}^f\mathbf{X} = ({}^f\mathbf{x}_t, {}^f\mathbf{x}_{t+1}, ..., {}^f\mathbf{x}_{t+n})$. The desired states $\mathbf{X}^* = (\mathbf{x}_t^*, \mathbf{x}_{t+1}^*, ..., \mathbf{x}_{t+n}^*)$ are then calculated as follows:

$$d_t^* = v_t T_d + d_{safety} \tag{14}$$

$$\psi_t = \arctan 2({}^f y_t - {}^f y_{t-1}, {}^f x_t - {}^f x_{t-1}) \tag{15}$$

$$x_t^* = {}^f x_t - d_t^* \cos(\psi_t) \tag{16}$$

$$y_t^* = {}^f y_t - d_t^* \sin(\psi_t) \tag{17}$$

$$\alpha_t^* = {}^f \alpha_t \, , \; v_t^* = {}^f v_t \tag{18}$$

And the formulated optimal control problem is:

$$\min_{u_a, u_s} \sum_{t=1}^{n} f(\mathbf{x}_t, \mathbf{x}_t^*, u_a, u_s) \tag{19}$$

Where $f(.)$ is the cost function which in this case is a linear sum of position, heading, velocity square errors along with costs associated with large magnitude accelerations and steering angles to discourage jerky behaviour. The evolution of states follows a noise-free version of the single-track model described above.

This is minimised directly with the Powell method [6] to give values for steering and acceleration for each step in the time horizon. These intended controls are then communicated via the v2v network. Since vehicles send their messages from front to back,

each follower vehicle uses the latest intent from the car ahead to for the prediction used in control.

Once the vehicles have all communicated their intentions, the predictions for the following time step are updated with the most recent intents for all vehicles. This ensures we avoid 'Chicken and Egg' style problems with each car needing to know each other's intention to do its own while still having all predictions for the estimators based on the latest intents.

### Reactive Control

For the reactive controlled vehicle, an adapted Stanley Controller is used. The desired state of the vehicle is calculated with respect to the vehicle it is following as above but for only one step into the future. Then the control is calculated separately for the acceleration based on PID for the longitudinal error and a combination of path and heading error for the steering angle. These gains are tuned with a genetic algorithm to minimise follow-distance errors.

## Implementation Details

The simulation was written in Python and relies on the numpy library for vector/matrix operations and the scipy for the implementation of the Powell optimizer.

The code is structured in 7 main files:

- controller.py - the code for the vehicle controllers.
- vehicle.py - this keeps track of the ground truth car position using the noisy single-track model.
- vehicle_estimator.py - this is where the estimates are kept and the EKF is implemented.
- vehicle_system.py - this aggregates the controller, vehicle and estimators for each vehicle and allows information to pass between these parts of a vehicle's system.
- world.py - this handles communication between vehicles as well as some initial setup
- scenario.py - the main entry point, handles running the scenarios.
- tune_controller.py - a genetic algorithm for tuning the controller gains and cost weights

The editable parameters include: Vehicle numbers, covariances of sensors, network loss, communication distance (how many cars away the vehicles send data), network loss (how often messages are lost), number of steps in simulation and the road shape and speeds.

## Results

### Communication Improves Estimation

The first results ( Figures 2 and 3 ) show how allowing communication between vehicles can improve their estimates. Not only do the vehicles improve the estimate of their own position by around double, but by communicating they get a relatively good estimate of where all vehicles in the platoon are. Obviously without communication, vehicles can only gain an idea of where another is through direct measuremnt.

By not just communicating estimates, but also intended controls, vehicles can further improve their estimates as their predictions can be more accurate (Figure 4 ). This has the most effect on estimates of more distant vehicles where there are no direct measurements.

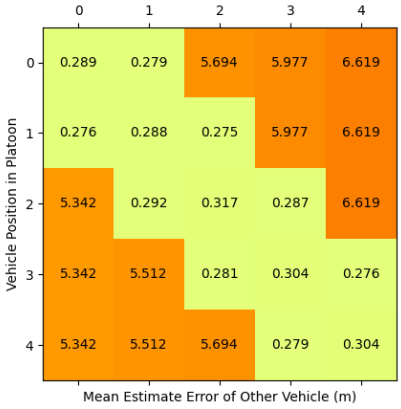


**Figure 2:** *Vehicle estimates of other vehicles in platoon for non-communicative scenario*

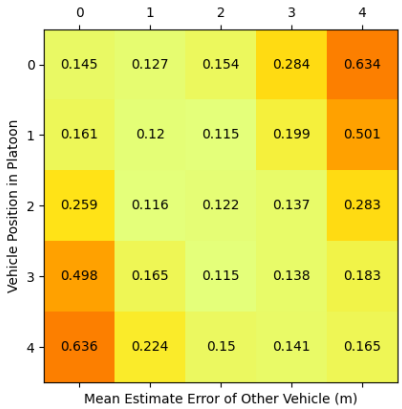


**Figure 3:** *Vehicle estimates of other vehicles in platoon for communicative, reactive scenario*

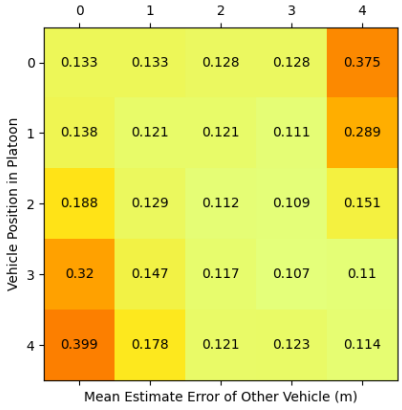


**Figure 4:** *Vehicle estimates of other vehicles in platoon for the NMPC scenario*

## NMPC Control Improves Follow Error

As shown in Figure 5, communicative systems are better at maintaining a set follow time. With the reactive controllers, this improvement comes from the improved estimates. Indeed, the improvement in follow error is nearly the same in magnitude as the improvement in estimation.

Greater improvements to following are made by using the optimal NMPC-based control. This is likely because the controls can be optimised not just for immediate cost minimisation at any given time, but can act in a way that considers how instantaneous actions will impact the future ability to maintain the desired follow time. Without this, the controller can easily get itself into longer-term trouble by chasing short-term rewards.

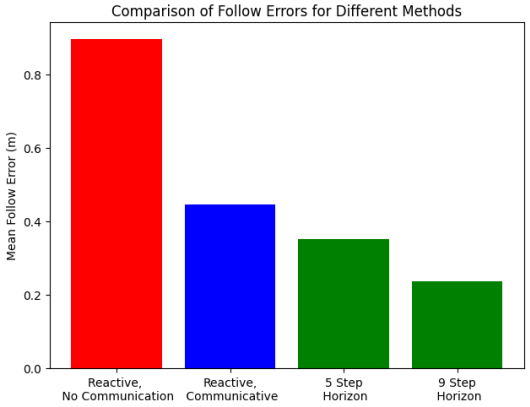


**Figure 5:** *Comparisson of average follow errors for various control and communication scenarios*

Longer MPC windows lead to better behaviour, however, this increases the size of the optimisation problem which increases the computation time. The best horizon would therefore depend on the efficiency of the optimizer used and the computational power of the vehicle. In fact, in the case where computation is limited and/or a faster control frequency is required, the MPC horizon can be so short and the optimizer can have too little time to reach a better solution than the reactive controller.

By looking at the shapes of the paths taken by the vehicles we can also see there is more to the story than just follow distance errors. We can see with reactive control, small perturbations can get magnified leading to the paths to warp over time. (Figure 6).

In the NMPC case (Figure 7), while we do not see this same magnification of errors, we do see a flattening of the curves. This is caused by the term in the cost function penalizing harder steering which encourages the vehicle to 'cut corners'.

In the extreme, this led to crashes when emergency stops were modeled in the reactive systems. Since they only react to a problem when the distance between the vehicles shrinks while the MPC can react as soon as the intent of the vehicles ahead changes.

In a more realistic design, the vehicles would laterally follow the road and not just each other. This would limit lateral drift and amplification of lateral errors.

In this case, the time delay was not modeled between the calculation of the controls and sending them on to the following vehicles. For this assumption to hold a very rapid optimization would need to be done.

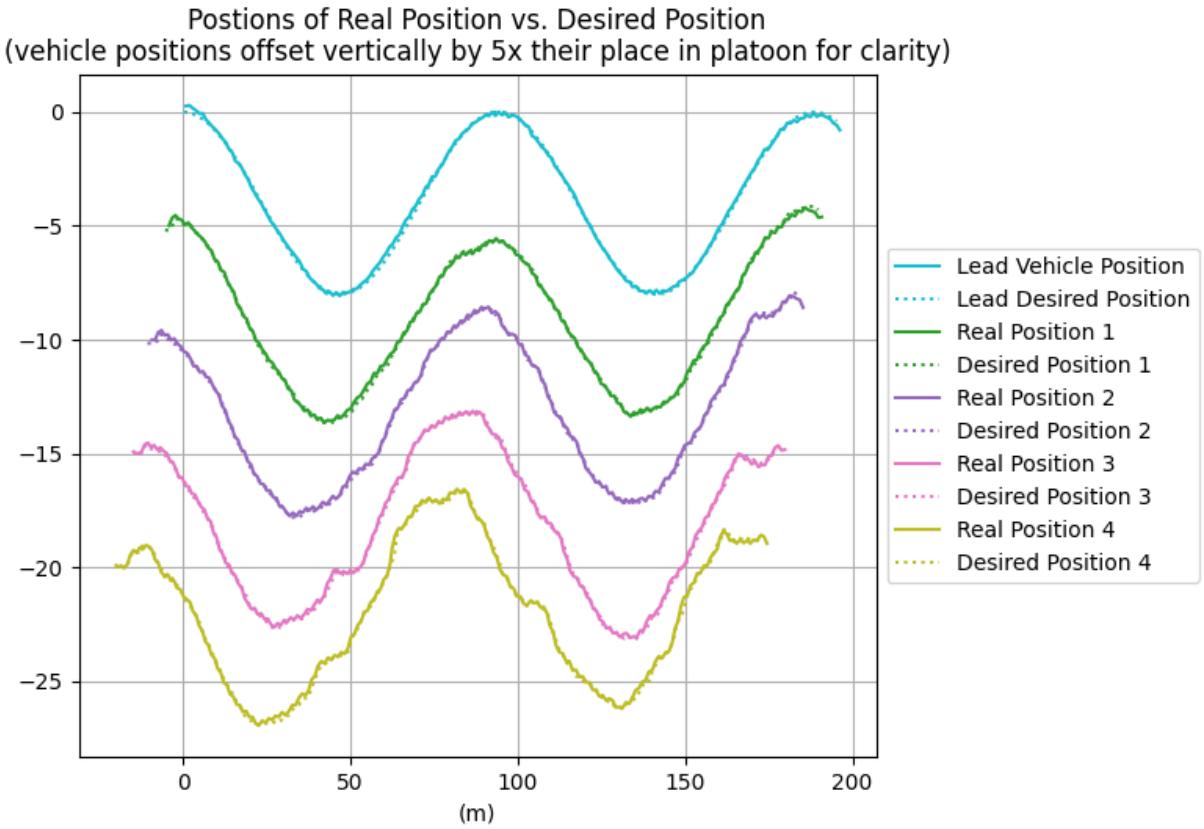


**Figure 6:** *Paths of vehicles in the communicative, reactive platoon*

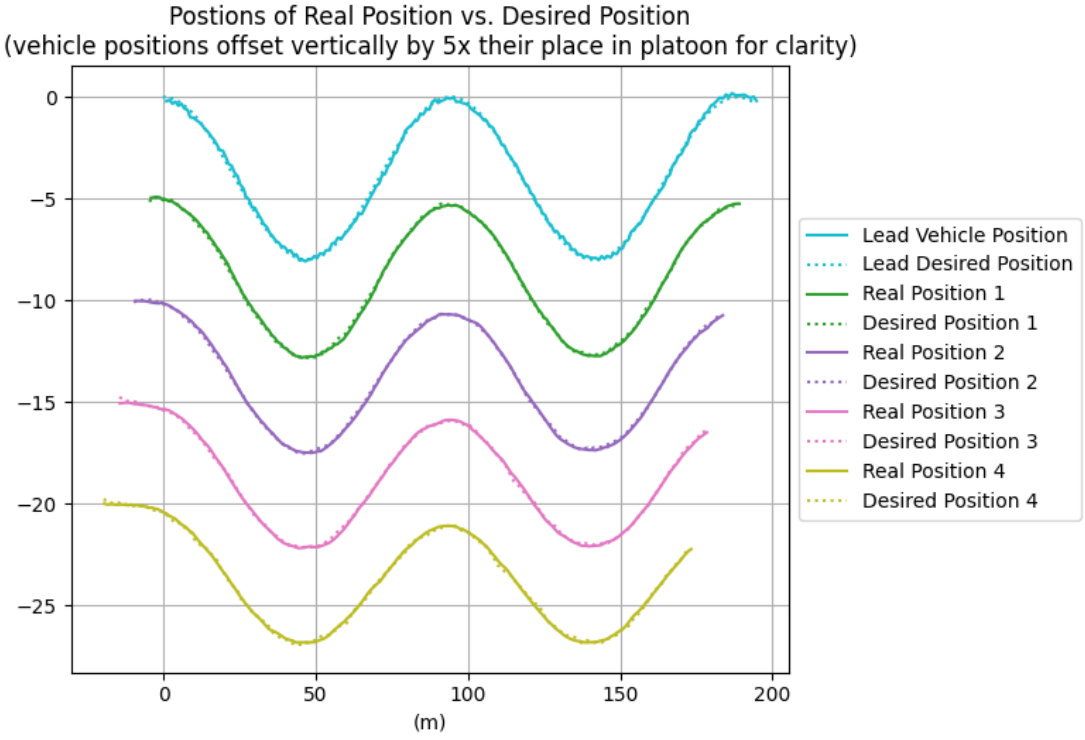


**Figure 7:** *Paths of vehicles in the communicative, npmc platoon with 9 time step horizon*

## Conclusion

Automated vehicle platooning provides an opportunity to increase efficiency, safety, and traffic flow. In this paper, a simple but effective method was shown to perform very well in a simulated situation compared to a reactive vehicle platoon.

Vehicle-to-Vehicle Communication, not only of estimates, but also of control intentions allows better coordination of movement through better estimation and prediction of other vehicle states compared to direct measurement alone.

The inclusion of Model Predictive Control allows control to be based on a longer time horizon of effect, leading to smoother, safer, more efficient driving.

Systems like the one presented here could be of great benefit to global transportation by reducing costs and pollution. Similar ideas could be taken beyond road transportation and extended to suit marine and air-based transport too.

However, the simulation lacks several aspects which limit its applicability to the real world. Firstly, time delays were not considered. The system was modeled as if all calculations and messaging were instantaneous.

While modern processors and communication infrastructure such as 5G allow for very low latencies, this delay will never be zero and will increase for larger fleets dealing with more information. Vehicles could compensate for the delays, however, the information they are using would not be as recent as modeled and the estimates and predictions would be worse.

Secondly, the model of the vehicle was very simple. Real vehicle physics is very complex, especially during accelerating or steering motions and on slippery surfaces. This again means estimates and predictions would be worse.

Therefore, given how safety-critical such systems need to be, simulation taking into account latencies and real-world physics would need to be conducted followed by extensive testing on real vehicles would need to be conducted prior to any release of such a system into production vehicles.

## References

[1] Dawei Pi et al. "Automotive platoon energy-saving: A review". en. In: *Renewable & Sustainable Energy Reviews* 179 (June 2023), pp. 113268–113268. DOI: https://doi.org/10.1016/j.rser.2023.113268. URL: https://doi.org/10.1016/j.rser.2023.113268 (visited on 07/19/2023).

[2] S Tsugawa, S Kato, and S Aioki. ""An automated truck platoon for energy saving," in: *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2011), pp. 4109–4114. DOI: 10.1109/IROS.2011.6094549..

[3] Thijs H. A. van den Broek, Jeroen Ploeg, and Bart D. Netten. "Advisory and autonomous cooperative driving systems". In: *2011 IEEE International Conference on Consumer Electronics (ICCE)*. 2011, pp. 279–280. DOI: 10.1109/ICCE.2011.5722582.

[4] C. Canudas De Wit and B. Brogliato. "Stability issues for vehicle platooning in automated highway systems". In: *Proceedings of the 1999 IEEE International Conference on Control Applications (Cat. No.99CH36328)*. Vol. 2. Kohala Coast, HI, USA: IEEE, 1999, pp. 1377–1382. ISBN: 9780780354463. DOI: 10.1109/CCA.1999.801173. URL: http://ieeexplore.ieee.org/document/801173/ (visited on 08/08/2023).

[5] Lingyun Xiao and Feng Gao. "A comprehensive review of the development of adaptive cruise control systems". en. In: *Vehicle System Dynamics* 48.10 (Oct. 2010), pp. 1167–1192. ISSN: 0042-3114, 1744-5159. DOI: 10.1080/00423110903365910. URL: http://www.tandfonline.com/doi/abs/10.1080/00423110903365910 (visited on 08/08/2023).

[6] R. Fletcher and M. J. D. Powell. "A Rapidly Convergent Descent Method for Minimization". en. In: *The Computer Journal* 6.2 (Aug. 1963), pp. 163–168. ISSN: 0010-4620, 1460-2067. DOI: 10.1093/comjnl/6.2.163. URL: https://academic.oup.com/comjnl/article-lookup/doi/10.1093/comjnl/6.2.163 (visited on 08/09/2023).