ADMM for Control of Mixed Traffic Flow with Human-driven and Autonomous Vehicles

AIMS Mini-Project 2

Matthew Newton

Supervised by
Antonis Papachristodoulou

1 Abstract

The issue of congestion on our roads is becoming of greater importance. As the number of vehicles worldwide increases, more advanced methods are being employed to improve traffic flow. Not only does traffic congestion increase travel times, but it also increases fuel consumption and has an impact on the environment. Traffic flow can be modelled through a non-linear dynamical system. The true nature of the dynamics is unknown and very complex, however the current models suggest that congestion can be caused by imperfections in human driving. Currently methods to deal with traffic flow involve enforcing restrictions on the roads themselves, which are limited to a static positioning. The rise of autonomous and connected vehicles presents a new opportunity to fix the traffic flow problem, as they have the ability to optimise the flow with a varying position. A traffic flow system can be modeled as a circular road, which aims to mimic the dynamics of the system along an infinite road - there are also real life experiments using this set-up to compare against. In this project, we assume there is one autonomous vehicle in the traffic system. An optimal control problem is set up to minimise a $\mathcal{H}_2$ norm that reflects the impact of disturbances on the system. This can be solved centrally and provides gains that are successful in stabilising the system. However, model privacy is not conserved, which is an issue in the real-world application, meaning this problem must be solved in a decentralized way to preserve this privacy. To implement model privacy, a controller structure constraint is enforced, which creates a non-linear optimal control problem. In this project an ADMM algorithm is proposed to solve a restricted version of the optimal control problem in a decentralised way.

2 Introduction

The number of vehicles on the roads within the UK has risen by approximately two million over the past decade [1]. And with worldwide car sales remaining at a high level [2], there is an every increasing demand to ensure that this mode of transport is both sustainable and efficient. Traffic congestion is a major issue on many of the world's most important roads and is unavoidable with the current technology available [3]. Not only does congestion increase passenger commuting time, but it has also been shown to increase fuel consumption [4], which in turn has huge negative impacts on the environment and the health of passengers. There has already been an extensive literature on the optimisation of traffic flow to reduce congestion [5] - the implementation of these methods over the years has seen a wide range of success [6]. Current methods include ramp meters (that control the influx of vehicles onto motorways), replacing intersections with roundabouts and imposing varying speed restrictions on motorways [7]. These techniques involve changing the roadway infrastructure to alter the driving patterns of humans and do so to varying degrees of success.

With recent developments in the production of autonomous vehicles [8], there is now an opportunity to create innovations to reduce congestion using techniques that were not possible previously. The previous traffic control measures are Eulerian, only varying in time and not in space. As autonomous vehicles move with the flow of traffic, they will be able to control the traffic flow in a Lagrangian fashion, varying both in space and time. As this has not been possible before, the potential impact this could have on reducing congestion is significant. Despite the fact that fully autonomous vehicles are not currently available, there has already been a large effort among the research community to create efficient and safe algorithms to optimise the flow of traffic using autonomous vehicles [8].

Along-side the large number of simulations that have attempted to model congestion, there have also been many interesting experiments that demonstrate the onset of traffic jams. The most famous lined up twenty-two human driven cars on a large circular road, with each vehicle starting with equal spacing [9]. Each participant was instructed to follow the other car at a constant velocity for a short period of time. What happened next was somewhat surprising, as the vehicle velocities quickly became unstable, forming a backwards traveling traffic wave that propagated around the circle. The video showing this experiment has almost surpassed three million views on YouTube [10]. It suggests that the resulting behaviour is surprising, as the task of following a car around in a circle appears so simple. However, this behaviour was predicted by researchers in this area, as traffic congestion has been commonly modelled through propagating waves [3].

The circular model is a simple way of modelling traffic flow on an infinitely long single-lane road and it is possible to create a state space model that imitates this behaviour. The model shown in [11] proposes a non-linear system, which can be linearised to generate a linear state space model. This model can recreate the dynamics as shown in the experiment and displays the backwards propagating wave and the instability among the vehicles [9].

In the 2017 SIAM Annual Meeting, a mini-symposium took place on Lagrangian Traffic Flow Control and Autonomous Vehicles. During this mini-symposium there was a talk where they present the original experiment [12]. The same traffic waves were recreated (sometimes referred to as stop and go waves due to the vehicles stopping and starting regularly). They then conducted a separate experiment with one of the vehicles was replaced with an autonomous vehicle. The amplitude of the waves was reduced dramatically and visibly the traffic congestion was reduced significantly. This is remarkable as it only took one autonomous vehicle among nineteen human driven vehicles to make a significant impact on the behaviour of traffic flow - a small penetration rate of 5%. The control strategy that was used is called a follower stopper algorithm [13].

This project is built upon research conducted by Y. Zheng, who previously worked on this problem and formulated an optimal control problem that used a centralised method to compute the feedback gain [14]. During his PhD he also worked on distributed design of decentralised controllers [15]. He created a decentralised control algorithm that took advantage of the properties of chordal decomposition of sparse block matrices, and applied the alternating direction method of multipliers (ADMM) to create the algorithm. The goal of this project is to incorporate the ADMM algorithm into the control strategy of the autonomous vehicles in the circular road model. It is also compared against the performance of the centralised optimal control strategy, which unlike the ADMM strategy does not preserve model privacy, which is important in a network of vehicles.

3 Background

3.1 Semi-Definite Programming

A semi-definite program is convex optimisation problem that involves optimising a linear objective function over a set of positive semi-definite matrices, often written with linear matrix inequalities (LMI) as constraints. This class of program can be written in both a primal and dual form.

Definition 1 *For a variable $x \in \mathbb{R}^m$ and symmetric matrices $A_0, A_1, \ldots, A_m \in \mathbb{S}^n$, an LMI constraint is a constraint in the form [16]*

$$A(x) := A_0 + \sum_{i=1}^{m} A_i x_i \succeq 0.$$

It can be shown through the convexity of the constraint that

$$A(x) \succeq 0 \Leftrightarrow v^T \left(A_0 + \sum_{i=1}^{m} A_i x_i \right) v \geq 0,\ \forall v \in \mathbb{R}^n.$$

This means that if $A(x) \succeq 0$ then all the eigenvalues of $A(x)$ are greater than or equal to zero.

Definition 2 *A semi-definite program is a convex optimisation problem which has a linear cost/objective function and an LMI constraint [16]. It can be written in its general primal form with variable $X \in \mathbb{S}^n$*

$$\begin{aligned} \text{minimize} \quad & \langle C, X \rangle,\ \text{w.r.t. } X \\ \text{subject to} \quad & \langle A_i, X \rangle = b_i,\ i = 1, \dots, m, \\ & X \succeq 0, \end{aligned}$$

where $C, A_i \in \mathbb{S}^n$, $i = 1, \dots, m$, $b \in \mathbb{R}^m$. The dual problem associated with this problem is

$$\begin{aligned} \text{maximize} \quad & b^T y,\ (\text{w.r.t. } y \text{ and } Z) \\ \text{subject to} \quad & Z + \sum_{i=1}^{m} y_i A_i = C \\ & Z \succeq 0, \end{aligned}$$

where $y \in \mathbb{R}^m$ and $Z \in \mathbb{S}^n$ are the dual variables associated with X.

The optimisation problems proposed in this project can be formed as a semi-definite program (SDP). There are many optimisation problems that can be written as SDP, meaning there are numerous SDP solvers available. SeDuMi [17], SDPT3 [18] and Mosek [19] are the most common, with SeDuMi being the main solver used throughout this project.

One of the largest issues with SDP is how the solvers scale with the number of states in the system. When the number of states gets large, the solvers can run into numerical issues or take too long to solve to be practical in real world applications [20]. As is the case in this project, methods must be developed to get around this issue and make the problem scalable to accommodate larger systems.

3.2 Chordal Graphs

A graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is defined as a set of vertices $\mathcal{V} = \{1, 2, \dots, n\}$ and a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. These vertices in the graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ can be split into a subset of vertices on a subgraph. A clique $\mathcal{C} \subseteq \mathcal{V}$ is subgraph such that all the vertices in the subgraph $\mathcal{C}$ form a complete graph - a complete graph is a graph such that any two nodes are connected by an edge. A maximal clique is a clique that is not a subset of any other clique. A graph can contain a cycle, which is defined by a set of pairwise distinct nodes $\{v_1, v_2, \dots, v_k\} \subset \mathcal{V}$ such that $(v_k, v_1) \in \mathcal{E}$ and $(v_i, v_{i+1}) \in \mathcal{E}$ for $i = 1, \dots, k-1$. A chord that lies on the graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is an edge that joins two non-adjacent nodes in a cycle.

Definition 3 *A connected undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is a chordal graph if every cycle of length four or greater has at least one chord.*

The properties of chordal graphs have been exploited in many problems, however there are also different classes of chordal graphs that have additional properties. For this reason it can be useful to extend a non-chordal graph to a chordal graph by adding additional edges to the non-chordal graph.

Definition 4 *The chordal extension of a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is denoted as $\hat{\mathcal{G}}(\mathcal{V}, \hat{\mathcal{E}})$, where $\mathcal{E} \subseteq \hat{\mathcal{E}}$.*

3.3 Sparse Matrix Decomposition

To be used in conjunction with semi-definite programming, chordal graphs can be represented as a symmetric matrix with a sparsity pattern, where the sparsity in this case corresponds to the positions of zeros in the matrix. For an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with a sparsity pattern represented by $\mathcal{E}$, a symmetric matrix $X \in \mathbb{S}^n$ has $X_{ij} = X_{ji} = 0$, $\forall i \neq j$ if $(i, j) \notin \mathcal{E}$. This means that the matrix X has a zeros in elements that correspond to the nodes that are not connected by edges on the graph.

It can be shown that the matrix X can be split up into smaller sub-matrices, with each sub-matrix corresponding to the maximal cliques of the graph. This is useful as it means that a problem with a large matrix that is computationally slow can be split into smaller matrices that are computationally faster to solve.

Theorem 1 *Consider the chordal graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ that is made up of maximal cliques $\{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k\}$. Then $Z \in \mathbb{S}^n_+(\mathcal{E}, 0)$ if and only if there exist $Z_k \in \mathbb{S}^{|\mathcal{C}_k|}_+$ for $k = 1, \ldots, n$ such that [21]*

$$Z = \sum_{k=1}^{n} E^T_{\mathcal{C}_k} Z_k E_{\mathcal{C}_k}, \tag{1}$$

where $\mathbb{S}^n_+(\mathcal{E}, 0) := \{X \in \mathbb{S}^n | X \succeq 0 | X_{ij} = X_{ji} = 0, \text{if } i \neq j \text{ and } (i, j) \notin \mathcal{E}\}$, $|\mathcal{C}_i|$ is the number of vertices in that clique and

$$(E_{\mathcal{C}_k})_{ij} = \begin{cases} 1, & \text{if } \mathcal{C}_k(i) = j \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

This idea can be extended to sparse block matrices, however the mathematics will not be displayed in this report, please refer to [15].

4 Modelling the Traffic Flow Problem

In this project, traffic flow will be modelled as a single-lane circular road of length L. Although this is a simplification of a real-world traffic-model, it provides a strong framework to analyse the system as it captures the traffic flow that would take place on an infinitely long road. The experiments conducted in [9] allow us to compare the theoretical model to the real world data, as well as using the real-world data to calibrate the parameters in the model.

It is assumed that the human vehicle's goal is to follow the car in front - although in practice the drivers tend to look across the ring and get distracted by other influences. Since the only input available to each vehicle is its own acceleration, the i-th vehicle is modelled with non-linear dynamics F at time t in the form

$$\dot{v}_i(t) = F\left(s_i(t), \dot{s}_i(t), v_i(t)\right), \tag{3}$$

where $s_i(t) = p_{i-1}(t) - p_i(t)$ is the distance between the i-th vehicle and the vehicle that is following it, $p_i(t)$ is the position of the i-th vehicle and $v_i(t)$ is the velocity of the i-th vehicle.

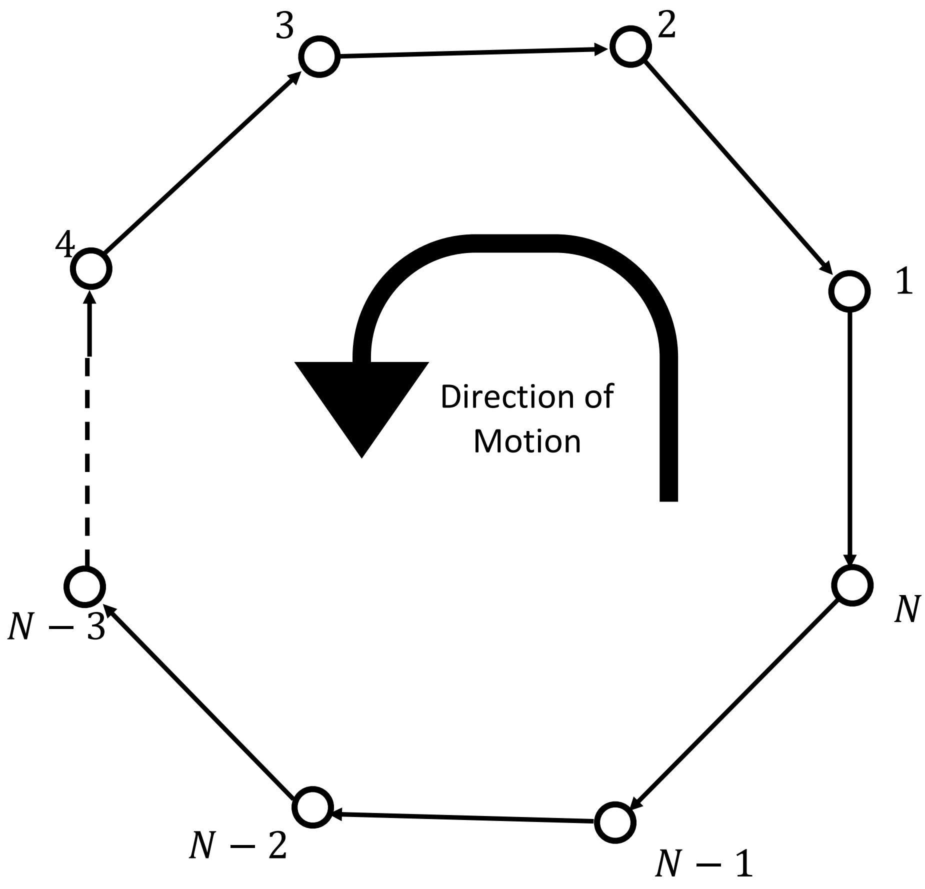


Figure 1: Plant graph of the ring model, showing that each vehicle is only influenced by the vehicle in front.

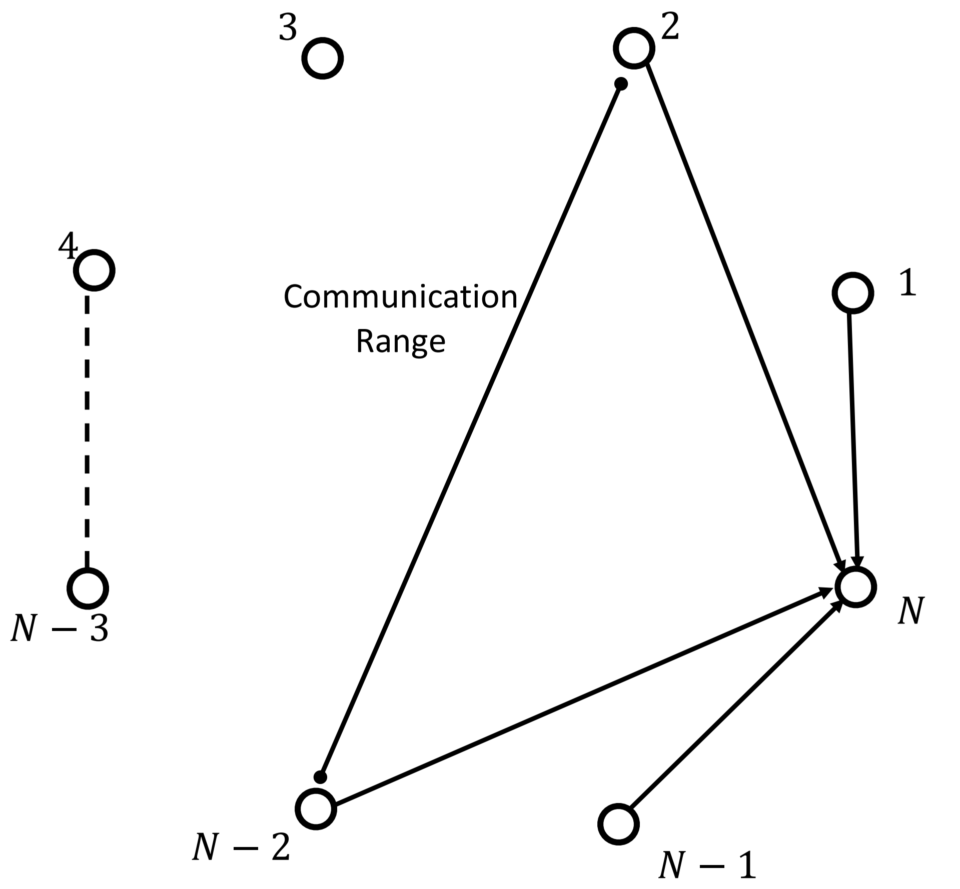


Figure 2: Communication graph of the ring model, showing that the autonomous vehicle can only receive information from vehicles within a certain communication range.

The system is linearised so it can be expressed in the standard linear dynamic model and therefore be used with the control optimisation algorithm

$$\dot{x}(t) = Ax(t) + Bu(t). \tag{4}$$

The system will be stable if $F(s^*, 0, v^*) = 0$ when $t \to \infty$, where s^* and v^* are the equilibrium spacing and velocity respectively. The error state of each vehicle is defined as

$$\begin{cases} \tilde{s}_i(t) &= s_i(t) - s^*, \\ \tilde{v}_i(t) &= v_i(t) - v^*. \end{cases} \tag{5}$$

It can be shown that the derivative is [14]

$$\begin{cases} \dot{\tilde{s}}_i(t) &= \tilde{v}_{i-1}(t) - \tilde{v}_i(t), \\ \dot{\tilde{v}}_i(t) &= \frac{\partial}{\partial t} F\left(s_i(t), \dot{s}_i(t), v_i(t)\right). \end{cases} \tag{6}$$

To simplify the time derivative of the velocity error, a first-order Taylor expansion about the equilibrium point is applied

$$
\begin{aligned}
\frac{\partial}{\partial t}F\left(s_i(t),\dot{s}_i(t),v_i(t)\right) &= \frac{\partial F}{\partial s}(s_i-s^*)+\frac{\partial F}{\partial \dot{s}}(\dot{s}_i-\dot{s}^*)+\frac{\partial F}{\partial v}(v_i-v^*)\\
&= \frac{\partial F}{\partial s}(s_i-s^*)+\frac{\partial F}{\partial \dot{s}}(\dot{s}_i-0)+\frac{\partial F}{\partial v}(v_i-v^*)\\
&= \frac{\partial F}{\partial s}\tilde{s}_i+\frac{\partial F}{\partial \dot{s}}(\tilde{v}_{i-1}(t)-\tilde{v}_i(t))+\frac{\partial F}{\partial v}\tilde{v}_i\\
&= \alpha_1\tilde{s}_i-\alpha_2\tilde{v}_i+\alpha_3\tilde{v}_{i-1},
\end{aligned}
$$

where $\alpha_1=\frac{\partial F}{\partial s}$, $\alpha_2=\frac{\partial F}{\partial \dot{s}}-\frac{\partial F}{\partial v}$ and $\alpha_3=\frac{\partial F}{\partial \dot{s}}$ - these represent the driver's sensitivity to the errors in the spacing and velocity. The α_i terms are tuned based on real world data and can be set to differ between drivers.

The state space equations can then be constructed by defining each state $x_i(t)=[\tilde{s}_i(t),\tilde{v}_i(t)]^T$. The derivative of the state is therefore

$$
\dot{x}_i(t)=\begin{bmatrix}\dot{\tilde{s}}_i(t)\\ \dot{\tilde{v}}_i(t)\end{bmatrix}^T=\begin{bmatrix}\tilde{v}_{i-1}(t)-\tilde{v}_i(t)\\ \alpha_1\tilde{s}_i(t)-\alpha_2\tilde{v}_i(t)+\alpha_3\tilde{v}_{i-1}(t)\end{bmatrix}^T. \tag{7}
$$

This can then be written as,

$$
\begin{bmatrix}\dot{\tilde{s}}_{i-1}(t)\\ \dot{\tilde{v}}_{i-1}(t)\\ \dot{\tilde{s}}_i(t)\\ \dot{\tilde{v}}_i(t)\\ \dot{\tilde{s}}_{i+1}(t)\\ \dot{\tilde{v}}_{i+1}(t)\end{bmatrix}=\begin{bmatrix}0&1&0&-1&0&0\\0&\alpha_3&\alpha_1&-\alpha_2&0&0\\0&0&0&1&0&-1\\0&0&0&\alpha_3&\alpha_1&-\alpha_2\\0&0&0&0&0&1\\0&0&0&0&0&\alpha_3\end{bmatrix}\begin{bmatrix}\tilde{s}_{i-2}(t)\\ \tilde{v}_{i-2}(t)\\ \tilde{s}_{i-1}(t)\\ \tilde{v}_{i-1}(t)\\ \tilde{s}_i(t)\\ \tilde{v}_i(t)\end{bmatrix}. \tag{8}
$$

This can be extended to all the vehicles/states in the system to form a block structured matrix, this is completed by defining two sub-matrices,

$$
A_1=\begin{bmatrix}0&-1\\ \alpha_1&-\alpha_2\end{bmatrix},\ A_2=\begin{bmatrix}0&1\\0&\alpha_3\end{bmatrix}. \tag{9}
$$

Assuming that all the vehicles are human driven, the matrix is block circulant and Toeplitz as it can be written as,

$$
\hat{A}=\begin{bmatrix}A_1&0&\dots&\dots&0&A_2\\A_2&A_1&0&\dots&\dots&0\\0&A_2&A_1&0&\dots&0\\ \vdots&\ddots&\ddots&\ddots&\ddots&\vdots\\0&\dots&0&A_2&A_1&0\\0&\dots&\dots&0&A_2&A_1\end{bmatrix}. \tag{10}
$$

For a system that has N vehicles on a circular road, the single autonomous vehicle is assumed to be the N-th vehicle. The autonomous vehicle's acceleration is governed by the control input, therefore there are no elements in the A matrix that correspond to its acceleration. However, its change in position is still a function of the velocity, i.e. $\dot{\tilde{s}}_N(t) = \tilde{v}_{N-1}(t) - \tilde{v}_N(t)$. The matrix then becomes

$$A = \begin{bmatrix} A_1 & 0 & \dots & \dots & 0 & A_2 \\ A_2 & A_1 & 0 & \dots & \dots & 0 \\ 0 & A_2 & A_1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & A_2 & A_1 & 0 \\ 0 & \dots & \dots & 0 & C_2 & C_1 \end{bmatrix}, \text{ where } C_1 = \begin{bmatrix} 0 & -1 \\ 0 & 0 \end{bmatrix}, C_2 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}. \tag{11}$$

The control vector is then a single input on the autonomous vehicle's acceleration such that

$$B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}. \tag{12}$$

It is assumed that there is a scalar disturbance $\omega_i(t)$ for $i = 1, \dots, N$ on the acceleration of each vehicle. The state space is then written as

$$\dot{x}(t) = Ax(t) + Bu(t) + H\omega(t) \tag{13}$$

where

$$H = \begin{bmatrix} H_1 & 0 & \dots & 0 \\ 0 & H_1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & H_1 \end{bmatrix} \text{ and } H_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{14}$$

It is also assumed that each vehicle has a breaking system to avoid crashes when it gets too close to the vehicle in front. The criteria for this is

$$\dot{v}(t) = a_{\min}, \text{ if } \frac{v_i^2 - v_{i-1}^2}{2s_i} \geq |a_{\min}|. \tag{15}$$

4.1 Optimal Control Strategy

The optimal control problem is set up using a system performance method. The outputs of the system are the states in the $x(t)$ vector and the control input $u(t)$, the output vector $z(t)$ can then be written as

$$z(t) = [\gamma_s \tilde{s}_1(t), \gamma_v \tilde{v}_1(t), \dots, \gamma_s \tilde{s}_N(t), \gamma_v \tilde{v}_N(t), \gamma_u u(t)]^T \tag{16}$$

where $\gamma_s, \gamma_v, \gamma_u$ are the performance weights that penalise the spacing error, velocity error and control input respectively. This is written as the matrix equation

$$z(t) = \begin{bmatrix} Q^{\frac{1}{2}} \\ 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ R^{\frac{1}{2}} \end{bmatrix} u(t), \tag{17}$$

where

$$Q^{\frac{1}{2}} = \operatorname{diag}\left(\gamma_s, \gamma_v, \dots \gamma_s, \gamma_v\right) \text{ and } R^{\frac{1}{2}} = \gamma_u. \tag{18}$$

The aim of the controller is to minimise the magnitude of impact that the disturbance has on the outputs of the system, which can be quantified by the transfer function $G_{z\omega}$ from ω to z. By minimising the $\mathcal{H}_2$ norm of the transfer function, the optimal control problem can be written as a non-convex optimisation problem

$$\begin{aligned} \text{minimize} \quad & ||G_{z\omega}||^2, \text{ (w.r.t. } K) \\ \text{subject to} \quad & u = -Kx, \\ & K \in \mathcal{K} \end{aligned}$$

where $K \in \mathcal{K}$ represents the structure of the controller, which contains the information available to the autonomous vehicle. This optimal control problem is non-convex, however can be relaxed to a convex problem, that can be solved using an SDP solver.

4.2 Reformulation of the problem

It is shown in [14] that the optimal control problem can be reformulated into an SDP with linear inequality constraints instead of the $K \in \mathcal{K}$ sparsity constraint. The problem takes the form

$$\begin{aligned} \text{minimize} \quad & \text{Trace}(QX) + \text{Trace}(RY), \text{ (w.r.t. } X, Y, Z) \\ \text{subject to} \quad & AX + XA^T - BZ - Z^TB^T + HH^T \preceq 0, \\ & \begin{bmatrix} Y & Z \\ Z^T & X \end{bmatrix} \succeq 0, \; X \succ 0, \; ZX^{-1} \in \mathcal{K} \end{aligned}$$

where $X \in \mathbb{R}^{2N \times 2N}$, $Y \in \mathbb{R}$ and $Z \in \mathbb{R}^{1 \times 2N}$.

5 ADMM Algorithm

5.1 Splitting up the Problem

Before the ADMM algorithm can be used, the problem must be split up into smaller sub-problems. To do this, the ideas that were used in chordal decomposition section are applied. Firstly, the plant and communication graphs must be combined and chordally extended, the diagram below shows how this is done.

$$\begin{tikzcd}
G^p \arrow[d] & & G^c \arrow[d] \\
G^p_m \arrow[dr] & & G^c_m \arrow[dl] \\
& G_s \arrow[d] & \\
& G_{ext} &
\end{tikzcd}$$

Figure 3: Procedure of how the graph is chordally extended. G^p and G^c are the plant and communication graphs respectively, G^p_m and G^c_m are the mirror graphs, G_s is the super graph and G_{ext} is the chordally extended graph.

The chordal graph can then be divided into cliques, where each clique forms a sub-problem. The cliques can then be formed into a clique tree which shows how the problems are divided up, each connection between the cliques represents the overlap between the problems. Each problem can be solved sequentially by moving down the clique tree, however in this project ADMM is used to ensure that privacy is conserved between the cliques.

5.2 How ADMM works

For an in-depth description of the algorithm, please refer to [15]. However, this report will provide an overview, so that the basic ideas of how it solves the optimisation problem can be understood.

ADMM is a first-order operator-splitting method that works by solving a problem of the form

$$\begin{aligned} \text{minimize} \quad & f(x) + g(x) \\ \text{subject to} \quad & Ex + Fy = c, \end{aligned}$$

where $x \in \mathbb{R}^{n_x}$ and $y \in \mathbb{R}^{n_y}$ are decision variables for the convex functions f and g, $E \in \mathbb{R}^{n_c \times n_x}$, $F \in \mathbb{R}^{n_c \times n_y}$ and $c \in \mathbb{R}^{n_c}$ are the problem data. ADMM works in an iterative fashion, with each iteration h having the equation

$$\begin{aligned}
x^{h+1} &= \text{argmin}_x \left(f(x) + \frac{1}{2}\rho||Ex + Fy^{(h)} - c + \lambda^{(h)}||^2 \right), \\
y^{h+1} &= \text{argmin}_y \left(g(y) + \frac{1}{2}\rho||Ex^{h+1} + Fy - c + \lambda^{(h)}||^2 \right), \\
\lambda^{h+1} &= \lambda^{(h)} + Ex^{h+1} + Fy^{h+1} - c,
\end{aligned}$$

where $\rho > 0$ is the penalty parameter and $\lambda \in \mathbb{R}^{n_c}$ is the dual variable. Each clique can be thought of as two computing agents that negotiate with each other on the overlapping variables.

6 Results and Discussion

To test and compare each control strategy, a set of parameters must be fixed for the traffic model. To see how well the algorithm is working, the parameters are selected such that the open loop of the system is unstable. All the simulations were run on an Intel Core i7 Processor at 2.2GHz, 16GB DDR3 RAM and an Intel Iris Pro 1536MB graphics card. The SDP solver used was SeDuMi with a combination of CVX and YALMIP as the SDP parser, depending on which was appropriate for the problem.

6.1 Model Parameters

Throughout all the analysis in this project, it has been assumed that there is only one autonomous vehicle, therefore all the tests will be performed with one autonomous vehicle, with the rest being human driven vehicles. The number of vehicles in the ring model is $N = 20$ as this is enough to show the traffic flow congestion, as well as keep the state space equations small. The length of the ring is $L = 400$m, which gives 20m of spacing between each vehicle, this is also the length that was used in the experimental tests [9], so direct comparisons can be made to the performance. The desired velocity is 15ms^{-1}. The driver sensitivity coefficients are $\alpha = 0.6 + U(-0.1, 0.1)$ and $\beta = 0.9 + U(-0.1, 0.1)$, where $U(\cdot)$ represents a uniform distribution. However, the sensitivity coefficients were altered during testing to examine the effects on the system. The vehicle parameters were set to: maximum acceleration $a_{\max} = 2\text{ms}^{-2}$, minimum acceleration $a_{\min} = -5\text{ms}^{-2}$, maximum velocity $v_{\max} = 30\text{ms}^{-1}$ and spacing parameters $s_{st} = 5\text{m}, s_{go} = 35 + U(-5, 5)\text{m}$. For the initial conditions, each vehicle was given equal spacing and an initial velocity of $v_0 = 15 + U(-4, 4)\text{ms}^{-1}$. The output performance parameters were set to $\gamma_s = 0.03, \gamma_v = 0.15, \gamma_u = 1$.

6.2 Open Loop Performance

To measure how strongly the controllers perform, it is useful to see how the system behaves with no controller input. For this, $K = 0$ and $A = \hat{A}$. It should also be noted that the continuous time state space model is converted to a discrete time model. To update each state a forward difference method is used such that

$$x(t + 1) = \dot{x}(t)\Delta T + x(t)$$

where $\Delta T = 0.01$s is small to ensure sufficient accuracy. Using this update equation, each state in x is updated with each time step - the trajectory of each vehicle can be calculated using this information. The velocity of each vehicle is plotted over time at different time scales.

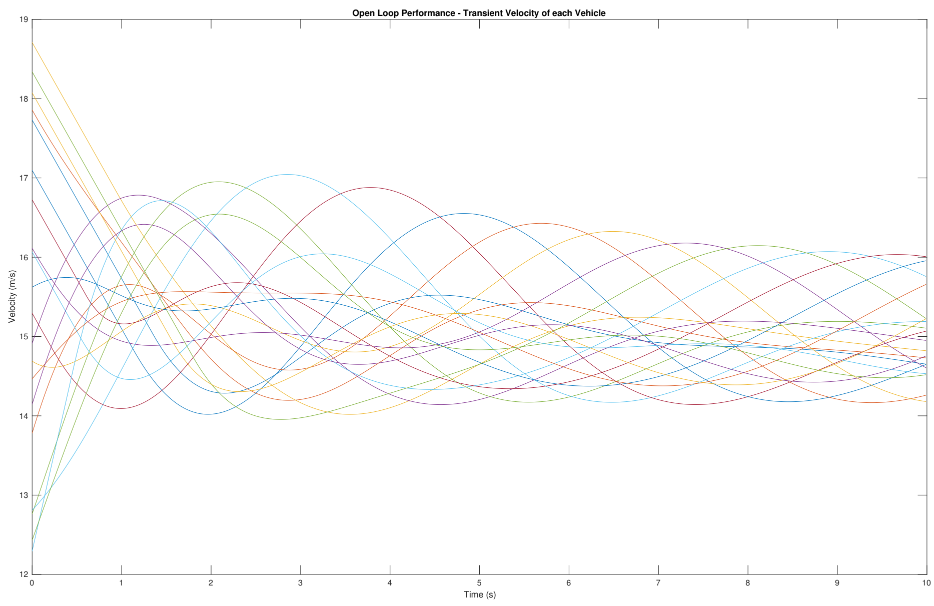


Figure 4: Velocity of each vehicle for time $0 - 10$s

It can been seen that each vehicle follows a sinusoidal pattern initially with varying frequency and amplitude, it is unclear from the transient behaviour if the system is stable or unstable.

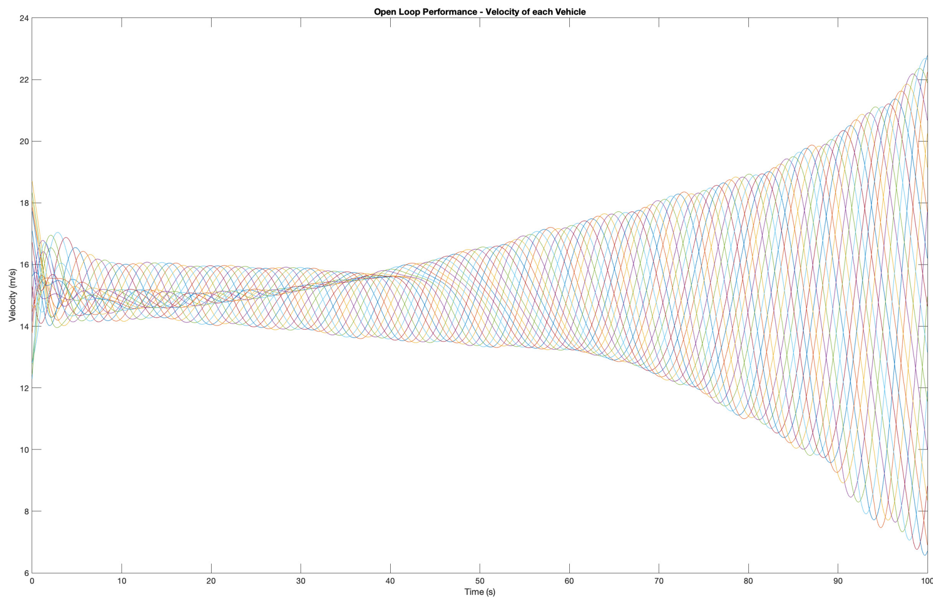
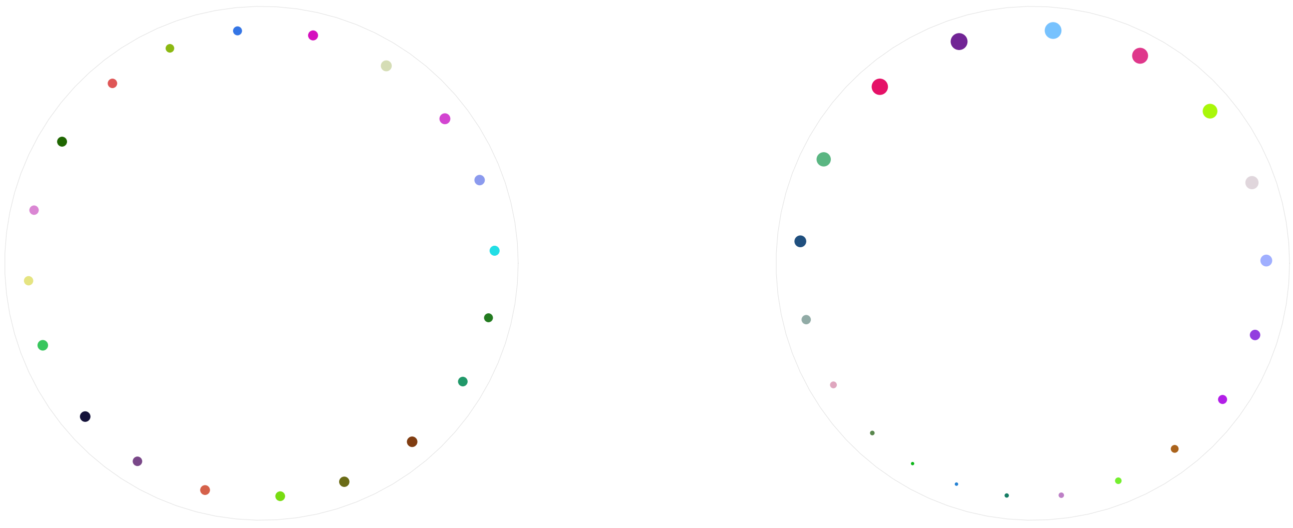


Figure 5: Velocity of each vehicle for time $0 - 100$s

The amplitude of each vehicles sinusoidal profile stays constant until approximately 40s, the behaviour then transitions and the amplitude of each wave starts to get larger. It is clear that the system is behaving in an unstable manner, as the fluctuations of each vehicle's velocity get larger. Eventually the velocity gets too large and the vehicles intersect/crash, the model then breaks down and the simulation is stopped.

The trajectories of each vehicle can be visualised on the circular model to show how the dynamics unfold over time. It is difficult to present these visualisations through static images on paper, the MATLAB code to create the dynamic visualations are available at the GitHub link provided [22]. However, the two snapshots below give an idea of the behaviour of the dynamics.


(a) Time after 10s

(b) Time after 100s

Figure 6: A visualisation of each vehicle on the road of length L. The size of the dot is proportional to the magnitude of the vehicle's velocity.

Initially the variations in the spacing and velocity appear to be dispersed around the road. However, over time the spacings and velocities start to form a pattern as they get larger and smaller following a sinusoidal wave. In the dynamic visualisation, the wave is shown to be travelling in the opposite direction to the velocity of the vehicles, as it propagates around the circle, getting larger and larger over time. This visualisation shows the system behaving similarly to how it would in a real-world experiment, which shows that the linearised model is performing adequately.

The eigenvalues of the open loop show the instability, with two positive eigenvalues of value 0.0271 and one zero eigenvalue. The zero eigenvalue represents the circular movement of all the vehicles, as the continual trajectory of all the vehicles is uncontrollable and is neither stable nor unstable.

6.3 Centralised Optimal Control

This optimal control strategy is an idealised method, as it assumes that all the vehicles have perfect communication with one another. It is essentially solving the optimisation problem in (19) but without the constraint on the structure of the controller. The optimisation problem then becomes a standard convex $\mathcal{H}_2$ optimal control problem

$$\begin{aligned}
\text{minimize} \quad & \text{Trace}(QX) + \text{Trace}(RY), \ (\text{w.r.t. } X, Y, Z) \\
\text{subject to} \quad & AX + XA^T - BZ - Z^TB^T + HH^T \preceq 0 \\
& \begin{bmatrix} Y & Z \\ Z^T & X \end{bmatrix} \succeq 0, \ X \succ 0
\end{aligned}$$

where $X \in \mathbb{R}^{2N\times 2N}$, $Y \in \mathbb{R}$, $Z \in \mathbb{R}^{1\times 2N}$. After testing the algorithm 10 times, the average time taken for it to solve was 11.2s. The value of the controller gain for the autonomous vehicle input is $K_c = 0.453$. This value can be used as a reference value for the other controllers, as the centralised case gives the most accurate results due to the perfect information assumption.

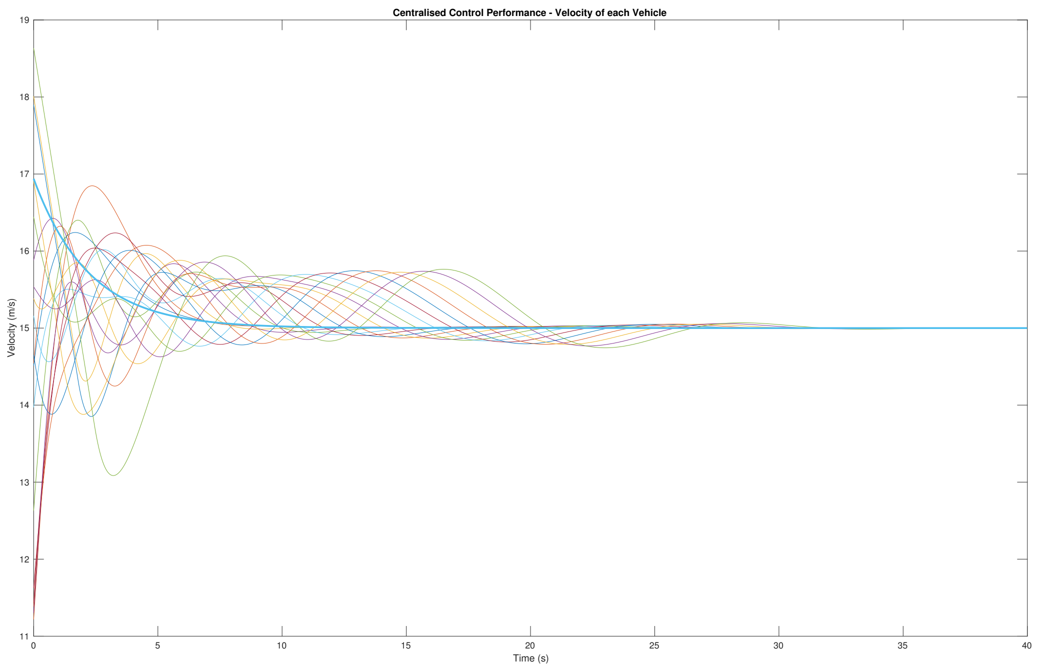


Figure 7: Velocity of each vehicle for time $0 - 40$s with one autonomous vehicle with controller value $K_c = 0.453$. The wider solid light blue line represents the velocity of the autonomous vehicle.

The plot shows that the centralised controller is successful in stabilising the system, with all of the states reaching their equilibrium at approximately 35s. The autonomous vehicle's trajectory does not follow the same pattern as the human driven vehicle. As the human driven vehicles oscillate around each other, the autonomous vehicle maintains a steadier velocity that acts as a dampening agent for all the other vehicles.

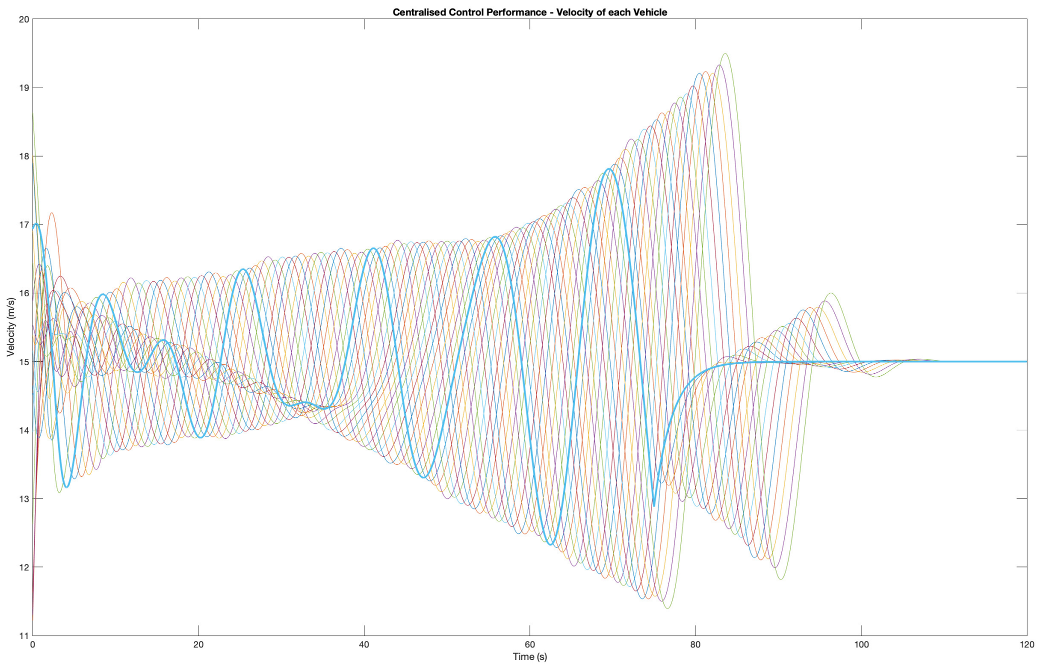


Figure 8: Velocity of each vehicle with control being activated 75s into the simulation. The wider solid light blue line represents the velocity of the autonomous vehicle.

Even after the system has started to behave in an unstable manner, if the controller is engaged part-way through the simulation it is possible for it to stabilise the system. The figure above shows the drastic change in the velocity of the autonomous vehicle once the controller is turned on and how the amplitude of the oscillations of the other vehicles quickly reduces until the whole system is stabilised. It is much easier to see how the system behaves in the dynamical visualisation, however it is not possible to represent the behaviour with static screenshots.

All of the eigenvalues of the closed loop system $(A - BK)$ are negative apart from the one zero eigenvalue. As mentioned previously, this does not mean the system is unstable, it reflects the constant movement of the states.

6.4 Decentralised Optimal Control in Centralised Way

The centralised controller performs well, however it is not a realistic control method. As the size of the state space gets larger, it will take longer to compute and the algorithm must be scalable. But most importantly, it does not preserve privacy in the system and assumes perfect communication between all of the states.

The decentralised controller aims to conserve model privacy, however the controller can be computed in such a way that it does not conserve this privacy, but still satisfies the constraints in the corresponding optimal control problem. To compute the decentralised controller, the controller structure constraint $K \in \mathcal{K}$ must be enforced, so the optimisation problem becomes the same as in (19).

Since $K = ZX^{-1}$ it follows that $ZX^{-1} \in \mathcal{K}$ - each matrix can then have its own sparsity pattern such that $Z \in \mathcal{K}$ and $X \in \mathcal{S}$. It can be shown that the sparsity pattern of $\mathcal{S}$ is [14]

$$S_{ij} = \begin{cases} 0, & \text{if } R_{ij} = R_{ji} = 1, \\ 1, & \text{otherwise} \end{cases}, \quad \forall i, j \in \{1, \dots, n\}, \tag{19}$$

where

$$R_{ij} = \begin{cases} 0, & \text{if } \exists l \in \{1, \dots, m\} \text{ s.t. } K_{lj} = 0, K_{li} = 1 \\ 1, & \text{otherwise} \end{cases}, \quad \forall i, j \in \{1, \dots, n\} \tag{20}$$

and $S \in \mathbb{R}^{m \times n}$. The notation used defines the sparsity pattern such that if $S_{ij} = 0$ then the corresponding X variable is $X_{ij} = 0$ and if $S_{ij} = 1$ then X_{ij} can take any value.

By enforcing these sparsity patterns, the non-convex problem is converted into a convex problem, which provides a sub-optimal solution to the original optimisation problem. This is because although the sparsity patterns enforced on Z and X ensure that $ZX^{-1} \in \mathcal{K}$, it may be possible that Z and X do not follow their individual sparsity pattern and still satisfy the sparsity pattern of $\mathcal{K}$.

When adding the constraints to the SDP, the optimal control problem is solved in an average time of 4.5s. This is faster than the centralised control and is likely because the sparsity pattern reduces the space available for the algorithm to search. The value of the controller is $K_{dc} = 0.508$, which is similar to the purely centralised case and performs comparably well.

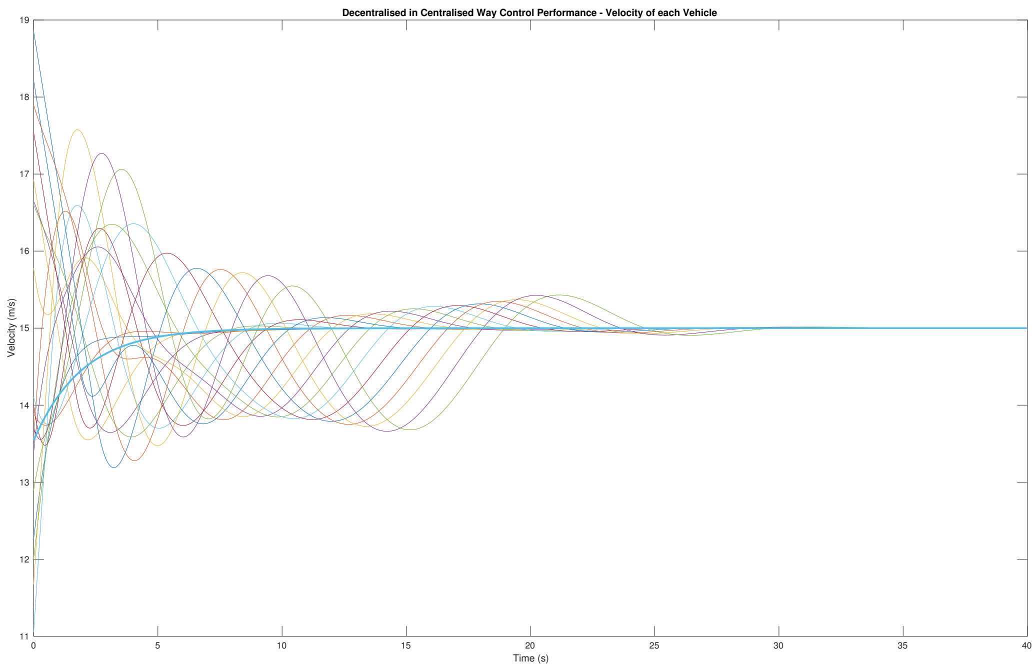


Figure 9: Velocity of each vehicle. The wider solid light blue line represents the velocity of the autonomous vehicle.

6.5 Decentralised Control through ADMM

Although the above decentralised controller solves the optimisation problem, it does so in a centralised way that does not preserve privacy between the different vehicles. As explained in section 5, ADMM can be used to solve optimisation problems through an exploitation of the properties of chordal graphs.

To set up this problem for ADMM, first of all the graph structure must be formalised. From figure 3, the plant graph can be expressed as the matrix

$$\mathcal{G}_p = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & \ddots & 0 & 1 \\ 1 & 0 & \dots & 0 & 0 \end{bmatrix}, \tag{21}$$

and the communication graph can also be written as

$$\mathcal{G}_c = \begin{bmatrix} 0 & 0 & 0 & \dots & \dots & \dots & 1 \\ 0 & 0 & 0 & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \ddots & \ddots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 & 1 \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 & \vdots \\ 0 & \dots & 0 & 0 & 0 & 0 & 1 \\ 0 & \dots & 0 & 0 & \dots & 0 & 1 \end{bmatrix}, \tag{22}$$

where the number of 1s in the last column is equal $2 \times CR + 1$, (CR is the size of the communication range).

As shown in figure 3, to find the chordal extension for ADMM, the super graph must first be created. The super graph therefore takes the form

$$\mathcal{G}_s = \begin{bmatrix} 0 & 1 & 0 & \dots & \dots & \dots & 1 \\ 1 & 0 & 1 & \ddots & \ddots & \ddots & \vdots \\ 0 & 1 & 0 & \ddots & \ddots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 & 1 \\ \vdots & \ddots & \ddots & \ddots & \ddots & 1 & \vdots \\ 0 & \dots & 0 & 0 & 1 & 0 & 1 \\ 1 & \dots & 0 & 1 & \dots & 1 & 1 \end{bmatrix}. \tag{23}$$

The chordal extension can be calculated by first finding a minimum degree ordering of the nodes and then performing a Cholesky factorisation. The chordal extension of the graph can be computed to be

$$\mathcal{G}_{\text{ext}} = \begin{bmatrix} 1 & 1 & 0 & 0 & \dots & 0 & 1 \\ 1 & 1 & 1 & 0 & \ddots & \ddots & 1 \\ 0 & 1 & 1 & \ddots & \ddots & 0 & \vdots \\ 0 & 0 & \ddots & \ddots & \ddots & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 1 & 1 \\ 0 & \dots & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & \dots & \dots & 1 & 1 & 1 \end{bmatrix} \tag{24}$$

with the graph being

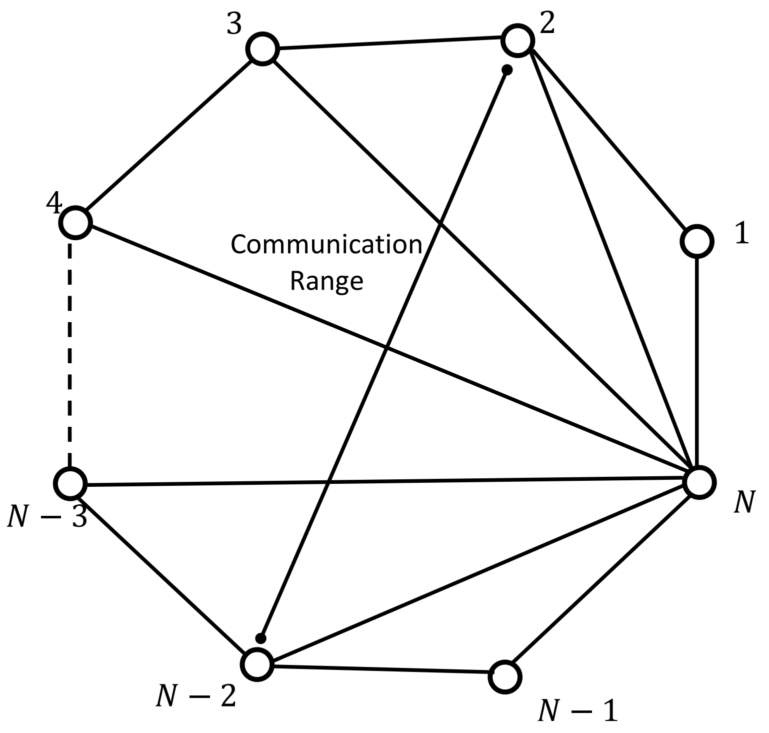


Figure 10: Chordally extended graph of the traffic system.

The ADMM algorithm in [15] was altered to operate with the structure of this problem. The ADMM algorithm was successfully implemented into the traffic model. Unfortunately, the algorithm was unable to converge to a solution. Multiple different techniques were used to try and overcome this issue, however none were successful in allowing the algorithm to converge. The zero eigenvalue of the A matrix was removed, which appeared to improve the operation of the algorithm, but it still would not converge. The size of the penalty parameter was adjusted, but each time the dual residual would reach the stopping criteria and the primal residual would continue to oscillate, unable to reach the stopping criteria.

This result is surprising as the decentralised control problem was solved in a centralised way, which means that the conditions for the ADMM algorithm to converge are satisfied. More research needs to be done on this problem to determine the cause of the algorithm not to converge and also if the

ADMM algorithm can be altered to generate a purely decentralised solution to the problem. The cause of this issue is likely due to a bug in the ADMM algorithm.

7 Conclusion

The main efforts in this project were to implement an ADMM algorithm to solve the optimal control problem for the traffic flow system. A traffic model was established that contained one autonomous vehicle among a set of human driven vehicles. The model was linearised into the standard control state space form. The optimal control problem was set up using the linearised dynamical model of the traffic system. The control problem was solved in a centralised way (by removing the controller structure constraint) to reproduce results that have been found previously. The optimisation was completed using a combination of the SDP parsers CVX [23] and YALMIP [24] with SeDuMi [17] and SDPT3 [18] as the solvers. A convex relaxation of the decentralised problem allowed it to be solved in a centralised way. This produced similar controller gains to the purely centralised case and resulted in comparable controller performance. The ADMM algorithm was unable to find a solution to the problem, as it could not converge. The algorithm should converge as all of the criteria are satisfied, further research must be conducted to find the cause of this issue, which is likely a bug in the ADMM algorithm.

References

[1] Statista. Number of cars in the uk 2000-2016, 2019.

[2] Statista. Global car sales 1990-2019, 2019.

[3] Benjamin Siebold. Lagrangian traffic flow control and autonomous vehicles. SIAM AN, 2017.

[4] S.K. Zegeye, Bart De Schutter, J Hellendoorn, Ewald Breunesse, and A Hegyi. Integrated macroscopic traffic flow, emission, and fuel consumption model for control purposes. *Transportation Research Part C: Emerging Technologies*, 31:158–171, 06 2013.

[5] Gabor Orosz, R Eddie Wilson, and Gábor Stépán. Traffic jams: Dynamics and control. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 368:4455–79, 10 2010.

[6] Stef Smulders. Control of freeway traffic flow by variable speed signs. *Transportation Research Part B: Methodological*, 24(2):111–132, April 1990.

[7] Markos Papageorgiou, Elias Kosmatopoulos, and Ioannis Papamichail. Effects of variable speed limits on motorway traffic flow. *Transportation Research Record: Journal of the Transportation Research Board*, 2047, 12 2008.

[8] Daniel Fagnant and Kara Kockelman. Preparing a nation for autonomous vehicles: Opportunities, barriers and policy recommendations. *Transportation Research Part A: Policy and Practice*, 77, 07 2015.

[9] Yuki Sugiyama et al. Traffic jams without bottlenecks—experimental evidence for the physical mechanism of the formation of a jam. *New J. Phys.*, 77, 07 2008.

[10] New Scientist. Shockwave traffic jams recreated for first time, 2008.

[11] Dirk Helbing. Traffic and related self-driven many-particle systems. *Reviews of Modern Physics*, 73, 12 2000.

[12] Raphael Stern. Controlling stop and go traffic with a single autonomous vehicle: Experimental results. SIAM AN, 2017.

[13] Raphael E. Stern, Shumo Cui, Maria Laura Delle Monache, Rahul Bhadani, Matt Bunting, Miles Churchill, Nathaniel Hamilton, R'mani Haulcy, Hannah Pohlmann, Fangyu Wu, Benedetto Piccoli, Benjamin Seibold, Jonathan Sprinkle, and Daniel Work. Dissipation of stop-and-go waves via control of autonomous vehicles: Field experiments. *Transportation Research Part C: Emerging Technologies*, 89, 05 2017.

[14] Yang Zheng, Jiawei Wang, and Keqiang Li. Smoothing traffic flow via control of autonomous vehicles, 12 2018.

[15] Yang Zheng. Chordal sparsity in control and optimization of large-scale systems (phd thesis). university of oxford., 2019.

[16] Eric Feron Stephen Boyd, Laurent El Ghaoui and Venkataramanan Balakrishnan. *Linear Matrix Inequalities in System and Control Theory.* 1994.

[17] J.F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11–12:625–653, 1999. Version 1.05 available from `http://fewcal.kub.nl/sturm`.

[18] M.J. Todd K.C. Toh and R.H. Tutuncu. Sdpt3 — a matlab software package for semidefinite programming, optimization methods and software. 1999.

[19] MOSEK ApS. *The MOSEK optimization toolbox for MATLAB manual. Version 9.0.*, 2019.

[20] Anirudha Majumdar, Georgina Hall, and Amir Ali Ahmadi. A survey of recent scalability improvements for semidefinite programming with applications in machine learning, control, and robotics, 08 2019.

[21] Naonori Kakimura. A direct proof for the matrix decomposition of chordal-structured positive semidefinite matrices. *Linear Algebra and Its Applications - LINEAR ALGEBRA APPL*, 433:819–823, 10 2010.

[22] mnewtonOX GitHub. Visualisation of traffic flow, 2019.

[23] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. http://cvxr.com/cvx, March 2014.

[24] J. Löfberg. Yalmip : A toolbox for modeling and optimization in matlab. In *In Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.