NTNU
Kunnskap for en bedre verden

Department of Mathematical Sciences

Speed Limit and Traffic-Light Control for Traffic-Flow Networks

Author:
Torjei Helset

Supervisors:
Franz Georg Fuchs, Knut-Andreas Lie, Kjetil Olsen Lye

Date

Table of Contents

List of Figures **iii**

List of Tables **iv**

1 Introduction **1**

2 Models **2**

2.1 First-Order Models 2

2.1.1 The LWR model 2

2.1.2 Time-delayed LWR model 3

2.2 Higher-Order Models 4

2.2.1 Aw–Rascle coupled with LWR 4

2.3 Weak Solutions and Uniqueness 4

3 Finite-Volume Methods **7**

3.1 Godunov's Method 8

3.2 Lax–Friedrichs 9

3.3 Rusanov 9

3.4 Higher-Order and High Resolution Schemes 9

3.5 Boundary Conditions 14

3.6 Convergence Analysis 15

3.6.1 Example 1 16

3.6.2 Example 2 18

3.6.3 Example 3 19

4 Automatic Differentiation **19**

4.1 Basics of AD 20

4.2 Implementation of AD 20

4.3 Validation of AD 20

4.3.1 Constant density on road 20

4.3.2 Manually calculating derivatives 21

5 Model Considerations **21**

5.1 Junctions 21

5.1.1 Entropy condition 22

5.1.2 Demand and supply 22

5.1.3 Remarks 24
5.1.4 Optimal choice of priority parameters 25
5.2 Traffic Lights 25
5.3 Inflow and Outflow Boundary Conditions 27
5.4 Variable Speed Limit 28
5.5 Limitations of Model 28
5.5.1 Duty to give way 28
5.5.2 More complicated road networks 29

6 Optimizing Flow of Traffic **29**
6.1 Objective Function 29
6.1.1 Minimal total travel time 30
6.1.2 Minimal total travel time and maximal outflow 30
6.2 Optimization Approach 30
6.2.1 Gradient descent 31
6.2.2 Sequential quadratic programming 31
6.2.3 Variable continuous speed limits 32
6.2.4 Variable speed limits and traffic lights 33

7 Numerical Results **33**
7.1 Network A 33
7.1.1 Optimizing only control parameters of the traffic light 34
7.1.2 Optimizing constant speed limits on roads 35
7.1.3 Optimizing variable speed limits on the roads 36
7.1.4 Optimizing controls for both traffic lights and speed limits 39
7.2 Network B 40
7.2.1 Optimizing only control parameters of the coupled traffic lights 41
7.2.2 Optimizing controls for both traffic lights and speed limits 42
7.3 Network C 43
7.3.1 Optimizing flow 44

8 Conclusion **46**

Bibliography **47**

Appendix **49**

A Analytical Solution of the Transport Equation **49**

List of Figures

1 The figure shows the approximate solution of Burgers' equation with initial data $\rho_0(x) = e^{-5(x-\frac{1}{2})^2}$. 5

2 Example of grid that is used in a finite-volume method. 7

3 Lax–Wendroff appoximation of (35) at time $t = 2s$. 11

4 The minmod reconstruction for (28). 12

5 The figure shows the numerical approximations to the solution of (35) for different choices of stepsize Δx. Two different limiters, the minmod and superbee limiters, are compared as well as the first-order Rusanov scheme. The two plots show that the approximation improves as the step length decreases. It also shows that the second-order scheme with either the minmod or superbee limiters seems to outperform the first-order Rusanov scheme slightly. 17

6 Solution of the linear advection equation at time $t = 0.5$. 18

7 Numerical approximation of linear advection equation at time $t = 10$ for the minmod and superbee limiters. 19

8 Demand and supply functions for a single road 23

9 Logistic function and the derivative of the logistic function. 26

10 Activation functions for two coupled traffic lights. 27

11 Graphical representation of road network A. The network consists of two roads connected in a junction with a traffic light between them. 33

12 Objective value as a function of the number of iterations for different starting points. 34

13 The path taken to obtain optimal solution for different starting points. 35

14 The figure shows the density at the end of the simulation for two different choices of control parameters for the traffic light connecting the two roads. 35

15 Objective value as a function of the number of iterations for different starting points. 36

16 The path taken to obtain optimal solution for different starting points. 36

17 Optimal choice of speed limits for different number of control points. 37

18 Objective value at the optimal point as a function of the number of levels for the speed limit. 37

19 Objective value as a function of number of iterations for some different starting points and number of control points. The starting points are given in Table 3 . . . 39

20 Objective value as a function of number of iterations for some different starting points with three control points. The starting points are given in Table 4 39

21 Objective function as a function of number of iterations for some different starting points. The starting points are given in Table 5 40

22 The figure shows the density at the end of the simulation for the parameters given by starting point A and the optimal solution in Table 5. 40

23 Graphical representation of road network B. The network consists of four roads connected in a junction with two coupled traffic lights between them. 41

24 Reduction in objective value as a function for the number of iterations for some different starting points. 41

25 Path to optimal solution for different staring points. 42

26 The figure shows the density at the end of the simulation for two different choices of control parameters for the traffic light connecting the two roads. 42

27 Reduction in objective value as a function for the number of iterations for some different starting points. The starting points are given in Table 6 43

28 The figure shows the density at the end of the simulation for the control parameters of starting point A and the optimal solution in Table 6. 43

29 Graphical representation of road network C. 44

30 Reduction in objective value as a function of the number of iterations for different starting points. The path starting from A has one control point for the speed limits at $t = 500$s and the path starting from B only allows the speed limit to be constant in time. 45

31 The figure shows the density at the end of the simulation for the control parameters of starting point A and the optimal solution specified in Table 7. 45

List of Tables

1 Estimated order of convergence estimated using the minmod and the superbee limiter and the first-order Rusanov scheme. Results are from the transport equation (35) with initial data (71) at time $t = 2.0$. 17

2 Estimated order of convergence using the minmod and the superbee limiter and the first-order Rusanov scheme. Results are from the linear advection equation (36) with initial data (37) at time $t = 0.5$. 18

3 Starting points for the optimization method for different number of control points. The number in the left column denotes the number of control points, and the letter specifies which starting point it is. **1***, **3*** and **7*** are the optimal solutions for one control point, three control points and seven control points respectively. 38

4 Starting points for the optimization method with three control points. If only speed limit is specified for the second road, it means that this speed limit is chosen for all times. 38

5 Starting point for the speed limits and cycle times for the traffic light. The optimal solution and the objective values at each starting point is also included. 40

6 Starting point for the speed limits and cycle times for the traffic light. The optimal solution and the objective values at each starting point is also included. Note that the time under green light is the time the there is green light for the traffic light that road 1 leads into. 43

7 Starting point for the speed limits and cycle times for the two coupled traffic lights. The optimal solution and the objective values at each starting point is also included. A and B correspond to using one control point at $t = 500$s for the speed limits and using no control points respectively. The subcategories **A** and **B** under **Traffic Lights** denote how long each traffic lights stay in state A and B respectively. State A for the first coupled traffic lights is to have green light for road 1 and red light for road 2. State A for the last coupled traffic lights is to have green light for road 3 and red light for road 4. 45

1 Introduction

Road congestion is a major problem in modern cities, resulting in delays and increased fuel consumption. As cities continue to expand, these issues become even more severe, and the need for innovative solutions reducing the congestion becomes increasingly pressing. There are several measures that can be taken to address these issues and alleviate the problems. One natural approach of addressing the issue of road congestion, involves enhancing the existing infrastructure by developing more roads and more lanes on each road. Even though this approach might help reduce the congestion of traffic in the short term, further development of the infrastructure will require a lot of resources. Hence, if the goal is to reduce the amount of resources being used, this might not be the best solution. On the other hand, the existing infrastructure is most likely not being utilized to its full capacity. A different approach for reducing the congestion of traffic could therefore be to find a better way of putting the existing infrastructure to use. This is the approach that we will consider in this project, where we will try to find the optimal choice of control parameters for the speed limits on the roads in the road network as well as optimizing traffic lights connecting these roads to improve the flow of traffic.

To this end, we will first derive a model to approximate the behaviour of traffic in a road network. There are two main regimes of traffic flow modelling. The first one is a microscopic model that considers each car in a road network. The major drawback of microscopic models is that they will typically be very computationally expensive. This leads us to the other regime, namely the macroscopic models. Instead of looking at each car separately, these models approximate the density of cars as a continuous “fluid”. This is the model regime that will be made use of in this project.

The underlying goal of the project is to improve the flow of traffic in a road network. With this goal in mind, it is not enough simply to create a realistic model of the road network. In addition, some method of optimizing the control parameters of the network is needed. The major problem is to estimate the sensitivity of the parameters, that is how the result of a simulation depends on the parameters. The obvious solution is to us a gradient-based optimization technique, which in turn introduces the problem of evaluating the gradient of a numerical function. This problem will be addressed with the help of automatic differentiation.

It should be noted that although optimizing the control parameters to reduce road congestion may reduce the overall fuel consumption on a road system in the short term, the long-term effects are not so obvious. One possibility is that the reduction of road congestion will have the negative effect of making people switch from public transportation to driving their own car. This effect is called induced demand, and is a well-known effect. Therefore, it might be interesting to change the focus of the objective. The flexibility of the model we will describe makes it easy to apply it to different problems. One possibility is to instead of trying to optimize the total flow of traffic, prioritize only parts of the traffic flow. This, in turn, makes it possible to optimize the control parameters of the road network to optimize the flow of public transport, hopefully leading more people to make the switch to public transport and thereby reducing the amount of resources being used.

Section 2 describes the details of deriving a macroscopic model for traffic on a single lane. With a simple model in place, Section 3 looks into some methods that can be used to obtain unique solutions of the model equations. As mentioned, the optimization part of the project requires gradient information. To this end, Section 4 describes one approach of evaluating the gradient of a numerical function. Section 5 will extend the simple model to include junctions and traffic lights, as well as discussing some of the limitations of such a model. In Section 6, we will look at how an objective function can be constructed to describe the flow of traffic, as well as describing some methods for finding the optimal solution of such an objective function. With all of the necessary tools introduced, Section 7 will consider some specific road networks, and will find the optimal set of control parameters for each network.

2 Models

As described in the introduction, one of the regimes of traffic-flow modelling approximates the density of cars as a continuous "fluid". This section will start by deriving some first-order models, as well as one second-order model. Finally, the concept of weak solutions, which is needed to obtain solutions of these models, is introduced, and some conditions for the solution to be unique will be described.

2.1 First-Order Models

Before looking at more complicated traffic models, we will start with the simplest case, the first-order models. The most famous first-order model is the LWR model, which we will derive in the subsequent subsection.

2.1.1 The LWR model

The first macroscopic model for traffic flow was introduced by Lighthill and Whitham [16, 17] and Richards [21]. The underlying idea was to replace individual cars with a continuous density and using a simple conservation law, and in this way, a model equation can be derived. The derivation procedure will be explained in detail for a simple case.

We consider a single lane road of length L without intersections. As mentioned, we replace the discrete collection of vehicles with a continuous medium. To this end, we introduce the density of cars $\tilde{\rho}(\tilde{x},\tilde{t})$, which will be expressed as the number of cars per unit length at any point along the road. We assume that $\tilde{\rho}$ is a piecewise constant function of position and time, and that it is bounded as

$$0 \leq \tilde{\rho} \leq \rho_{\max},$$

where $\rho_{\max}$ denotes the maximum density of cars, assuming that all cars are of the same length. We further let $v_{\max}$ denote the speed limit on the single-lane road. We assume that the speed of the cars only depends on the density of cars and the speed limit of the road, i.e., we write the speed as

$$\tilde{v}(\tilde{\rho},\tilde{x},\tilde{t};v_{\max}) = \tilde{v}\big(\tilde{\rho}(\tilde{x},\tilde{t});v_{\max}\big).$$

We can now introduce the flux of cars, i.e., the number of cars passing a point per time unit, as

$$\tilde{f}(\tilde{x},\tilde{t}) = \tilde{\rho}(\tilde{x},\tilde{t})\tilde{v}\big(\tilde{\rho}(\tilde{x},\tilde{t};v_{\max})\big).$$

Next, we derive a model describing the density $\tilde{\rho}(\tilde{x},\tilde{t}) \in [0,\rho_{\max}]$ of cars at position $\tilde{x} \in [0,L]$ at time $\tilde{t} \in \mathbb{R}^+$. To this end, we consider the sub-interval of the road $(a,b) \subset [0,L]$. On any such sub-interval, the change in the total number of cars is equal to the number of cars entering the system minus the number of cars leaving the system. In integral form, the conservation law can be written as

$$\frac{d}{d\tilde{t}}\int_a^b \tilde{\rho}(\tilde{x},\tilde{t})\,d\tilde{x} = -\big[\tilde{f}(b,\tilde{t}) - \tilde{f}(a,\tilde{t})\big]. \tag{1}$$

Applying the fundamental theorem of calculus, we get the differential form

$$\tilde{\rho}(\tilde{x},\tilde{t})_{\tilde{t}} + \big(\tilde{\rho}(\tilde{x},\tilde{t})\tilde{v}(\tilde{\rho}(\tilde{x},\tilde{t});v_{\max})\big)_{\tilde{x}} = 0. \tag{2}$$

Omitting function parameters, we write the differential form as

$$\tilde{\rho}_{\tilde{t}} + \big(\tilde{\rho}\tilde{v}\big)_{\tilde{x}} = 0. \tag{3}$$

So far, we have said that the speed depends only on the density and the speed limit. We further assume that when the density of cars is low, the speed approaches the speed limit, and that as the density increases, the speed will decrease. A multitude of different models satisfying these assumptions have been used in prior works. Maybe the simplest one is a linear interpolation that leads to the simple relation

$$\tilde{v}(\tilde{\rho}; v_{\max}) = v_{\max}(1 - \tilde{\rho}). \tag{4}$$

So far we have used the $\sim$ notation for all variables without explaining its meaning. This notation is used to signify that these variables are physical variables, relating directly to the road system being modelled. However, instead of using these variables directly in the simulation, we will introduce new *scaled* variables $x \in [0, 1]$, $\rho \in [0, 1]$ and $v \in [0, 1]$ that satisfy

$$\begin{cases} \tilde{x} = Lx, \\ \tilde{\rho} = \rho_{\max}\rho, \\ \tilde{v} = v_{\max}v. \end{cases} \tag{5}$$

We insert the new variables using relation (5) and obtain

$$\frac{\partial \tilde{\rho}}{\partial \tilde{t}} + \frac{\partial(\tilde{\rho}\tilde{v})}{\partial \tilde{x}} = \rho_{\max}\frac{\partial \rho}{\partial \tilde{t}} + \frac{\rho_{\max}v_{\max}}{L}\frac{\partial(\rho v)}{\partial x} = 0. \tag{6}$$

Thus, we can write a new model equation as

$$\rho_{\tilde{t}} + \gamma(\rho v)_x = 0, \tag{7}$$

where $\gamma = v_{\max}/L$.

The relation between the new speed and density variables using the linear interpolation model will now be

$$v(\rho) = 1 - \rho. \tag{8}$$

Even though we keep the physical variable $\tilde{t}$, we will drop the $\sim$ notation in subsequent sections. Also dropping the $\sim$ notation for $\tilde{t}$, we rewrite the transport equation for the traffic flow as

$$\rho_t + \gamma(\rho v)_x = 0. \tag{9}$$

2.1.2 Time-delayed LWR model

The classical LWR model is a simplification of reality and fails to capture some phenomena observed in traffic. Goatin et al. [5] showed that the LWR model fails to qualitatively match experimental data, especially at higher densities. To overcome some of the issues with the LWR model, several second-order models have been proposed. Before we look at some of these, we will look at one way of improving the first-order model by adding a time delay term in the flux function as introduced by Newell [18]. This delay time takes into account that velocities can not change instantaneously, which is one of the main issues not considered in the LWR model.

One such time-delayed model was derived from a microscopic model in [2]. Letting $T \geq 0$ be the delay and keeping the delay time explicit, the delayed LWR model can be formulated as

$$\rho(x, t)_t + (v(\rho(x, t - T))\rho(x, t))_x = 0. \tag{10}$$

In [8], some of the theoretical and numerical properties of this model were investigated. It was shown that this model conserves mass and positivity of density. However, one drawback is that it no longer guarantees a maximum density, meaning that $\rho > \rho_{\max}$ is possible.

Comparing this model to the LWR model, it was shown that for the right choice of parameters that this model was able to reproduce *stop-and-go waves*, which cannot be reproduced by the LWR model. A stop-and-go wave occurs when a car in heavy traffic slows down slightly. This slowing down causes the car behind to slow down even more, and this slowing down propagates backwards while getting increasingly worse. This phenomena causes traffic jams to form without any external reason, like a traffic light or a car accident. This phenomena wass analysed by e.g. Laval and Leclercq [14] and Ros et al. [7].

2.2 Higher-Order Models

With the goal of addressing some of the known drawbacks of the first-order LWR model, we will now look at a second-order macroscopic model. Some models that attempted to correct the earliest prototypes of a second-order model were introduced by Aw and Rascle [1] and Colombo [3]. Following the work done in [5], we will now look at how the Aw–Rascle model can be combined with the LWR model.

2.2.1 Aw–Rascle coupled with LWR

Even though the Aw–Rascle model [1] is more suitable than LWR in some cases, it also has some drawbacks. Goatin et al. [5] attempted to fix some of these drawbacks by combining the Aw–Rascle with the LWR model. One major drawback of the Aw–Rascle model is that the maximal speed reached on an empty road depends on initial data, which is clearly wrong. Therefore, this model is not suitable for lower densities. Since the two models appear to have their separate domains where they are suitable, the idea is to use the LWR for lower densities and the Aw–Rascle model for higher densities. The two density phases can be described as free flow and congested flow. Let Ω_f denote the free flow-phase, and let Ω_c denote the congested-flow phase. The full model can then be given as

$$\begin{cases} \rho_t + (\rho v)_x = 0, & v = v(\rho) \qquad\qquad \text{if } (\rho, v) \in \Omega_f \\ \rho_t + (\rho v)_x = 0, & (y)_t + (yv)_x = 0 \quad \text{if } (\rho, v) \in \Omega_c \end{cases} \tag{11}$$

where $y = \rho(v + p(\rho))$ is a generalized moment of cars, and $p(\rho) = V_{\text{ref}} \ln(\rho/V_{\max})$ is a pressure function. (Can also be on a different form as long as some conditions are met.) The constant $V_{\max}$ is the maximal speed, and V_{ref} is some reference speed.

In Ω_c, we no longer assume the simple relation between ρ and v. The benefit of this model is that it fixes the issues with the Aw–Rascle model at lower densities, while at the same time giving results that fit experimental data better than the LWR model does. In future work, it would be interesting to use a more accurate model, such as this one, for modelling the flow of traffic. However, for the purposes of this project we will stick to the simple first-order LWR model.

2.3 Weak Solutions and Uniqueness

In this section, we introduce the concept of weak solutions and discuss some criteria that are needed to obtain a unique solution to the LWR model that we introduced in Section 2.1.1. The model equation of the LWR model (9) is a special case of a scalar conservation law, which takes the form

$$\rho_t + f(\rho)_x = 0, \tag{12}$$

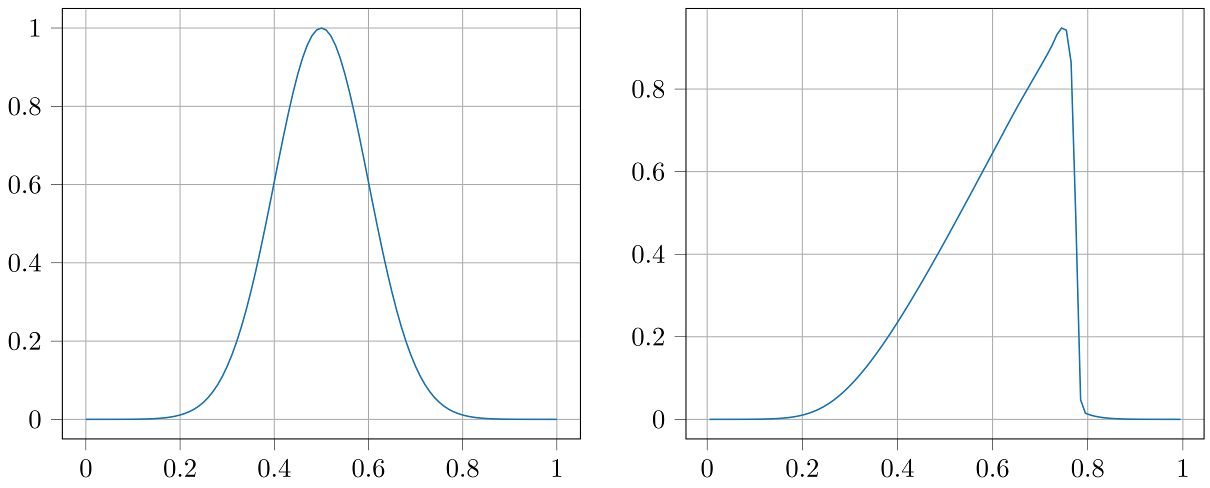
in general. If Equation (12) is nonlinear, smooth or classical solutions may not exist. However, since these equations model physical phenomena, some type of solution does exist, but if a classical solution does not exist, we need to introduce the notion of a *weak solution*.

The simplest nonlinear equation on the form of (12) is know as Burgers' equation, and is given as

$$\rho_t + \left(\frac{\rho^2}{2}\right)_x = 0. \tag{13}$$

This equation will in some cases develop discontinuities in finite time, known as shocks, even if the initial condition is smooth. In addition, rarefaction waves may form in finite time. The name rarefaction wave comes from the fact that the fluid becomes more rarefied when the density decreases. Figure 1 shows that both a rarefaction and a shock wave can form in finite time from smooth initial data.

We will now introduce the notion of a weak solution. Suppose for the moment that (12) has a classical solution. Let $\phi \in C_c^1(\mathbb{R} \times \mathbb{R}^+)$, where $C_c^1(\mathbb{R} \times \mathbb{R}^+)$ is the space of all functions that are continuously differentiable with compact support. If a classical solution ρ of (12) exists, the



(a) Initial data $\rho_0(x)$.

(b) Solution $\rho(x, t)$ at time $t = 0.3$.

Figure 1: The figure shows the approximate solution of Burgers' equation with initial data $\rho_0(x) = e^{-5(x-\frac{1}{2})^2}$.

equation must also hold if we multiply by the smooth test function ϕ and integrate over the domain. That is, the equation

$$\int_{\mathbb{R}\times\mathbb{R}^+} \rho_t\phi + f(\rho)_x\phi \, dxdt = 0, \tag{14}$$

must also hold true. We apply integration by parts to (14) and obtain

$$\int_{\mathbb{R}\times\mathbb{R}^+} \rho\phi_t + f(\rho)\phi_x \, dxdt + \int_{\mathbb{R}} \rho_0(x)\phi(x,0)\, dx = 0. \tag{15}$$

Since ϕ was chosen arbitrarily from the space $C_c^1(\mathbb{R}\times\mathbb{R}^+)$, the identity (15) holds true for all test functions ϕ and will be used to define what is meant by a weak solution. Following [15] we have the following definition of a weak solution.

Definition 2.1 (Weak solution)**.** *A function $\rho \in L^\infty(\mathbb{R}\times\mathbb{R}^+)$ is a weak solution of* (12) *with initial data $\rho_0 \in L^\infty$ if the identity* (15) *holds for all test functions $\phi \in C_c^1(\mathbb{R}\times\mathbb{R}^+)$.*

We can show that if a weak solution ρ of (12) is also differentiable, it will satisfy (12) point-wise. This means that the class of weak solutions will contain all classical solutions, or in other words, all classical solutions are also weak solutions.

The definition of a weak solution ρ does not require ρ to be differentiable or even continuous. This means that the solution can contain discontinuities, or shock waves. In general, the weak solution to a conservation law is not unique. Hence, to regain a unique solution, some extra conditions are needed. To get an idea of what conditions must be satisfied by shock waves, we consider the case where the weak solution ρ consists of two smooth regions with a shock wave separating the them. We will now investigate which conditions can be assumed on such a solution. As before we let $\phi \in C_c^1(\mathbb{R}\times\mathbb{R}^+)$ be a test function. We assume ϕ has support in Ω for some open set Ω. We define the two regions where the weak solution is smooth as Ω^+ and Ω^-. Since the weak solution is smooth on the two regions, we have $\rho \in C^1(\Omega^+)$ and $\rho \in C^1(\Omega^-)$. Since ρ is a weak solution, it will satisfy the identity (15). Using the compact support of ϕ, integration by parts and splitting

Ω into the two regions we rewrite (15) as

$$\begin{aligned}\int_{\Omega} \rho\phi_t + f(\rho)\rho_x \, d\Omega &= \int_{\Omega^+} \rho\phi_t + f(\rho)\rho_x \, d\Omega + \int_{\Omega^-} \rho\phi_t + f(\rho)\rho_x \, d\Omega \\ &= -\int_{\Omega^+} (\rho_t + f(U)_x)\phi \, d\Omega + \int_{\partial\Omega^+} \big(\rho^+(t)\nu^t + f(\rho^+(t))\nu^x\big)\rho \, d\Omega \\ &\quad - \int_{\Omega^-} (\rho_t + f(U)_x)\phi \, d\Omega + \int_{\partial\Omega^-} \big(\rho^-(t)\nu^t + f(\rho^-(t))\nu^x\big)\rho \, d\Omega \\ &= 0,\end{aligned}$$

where $\rho^+(t)$ and $\rho^-(t)$ are the trace values of ρ on the respective sides of the discontinuity, and ν is the unit outward normal of the shock wave.

For the unit normal of the shock wave $\sigma(t)$, we have

$$(\nu^t, \nu^x) = (-s(t), 1),$$

where $s(t) = \sigma'(t)$ is the speed of the shock curve.

We assumed the weak solution to be smooth in Ω^+ and Ω^-, and so Equation (12) is satisfied point-wise. This implies that

$$\int_{\Omega^+ \cup \Omega^-} (\rho_t + (f(\rho))_x)\phi \, d\Omega = 0.$$

Inserting this into the above identity gives

$$\int_{\partial\Omega} (s(t)(\rho^+(t) - \rho^-(t)) - (f(\rho^+(t)) - f(\rho^-(t))))\phi \, d\Omega = 0.$$

Since ϕ can be chosen arbitrarily, this implies that the integrand must be equal to zero. That is, the speed of the shock curve, $s(t)$, satisfies

$$s(t) = \frac{f(\rho^+(t)) - f(\rho^-(t))}{\rho^+(t) - \rho^-(t)}, \tag{16}$$

which is known as the *Rankine–Hugoniot* condition. This condition can be used to obtain a unique solution when a shock wave appears.

In this example, we only considered the formation of one shock wave, but as mentioned before, rarefaction waves may also start to form in finite time. The Rankine–Hugoniot condition is not enough to uniquely determine a weak solution in this case.

Motivated by the second law of thermodynamics, which states that the entropy of a system must be non-decreasing with time, [15] imposes some so called *entropy conditions* to regain uniqueness. The first entropy condition, is the Lax entropy condition:

Lax Entropy condition: *For a convex scalar conservation law, a discontinuity propagating with speed s given by* (16) *satisfies the Law entropy condition if*

$$f'(\rho^-) > s > f'(\rho^+). \tag{17}$$

A second entropy condition comes from the quantification of the rate of spreading of the characteristics.

Oleinik Entropy condition: *$\rho(x,t)$ is the entropy solution to a scalar conservation law on the form* (12) *with $f''(\rho) > 0$ (convex) if there is a constant $E > 0$ such that for all $a > 0$, $t > 0$, and $x \in \mathbb{R}$,*

$$\frac{\rho(x+a,t) - \rho(x,t)}{a} < \frac{E}{t}. \tag{18}$$

3 Finite-Volume Methods

The non-linear transport equation (9) typically does not have an explicit formula for the solution. Therefore, the solution will be approximated numerically using a finite-volume method, as described in [15] Since the transport equation (9) is non-linear, solutions may contain shock waves as well as rarefactions as described in Section 2.3. We therefore need the concept of weak solutions, and we need a way to obtain a unique solution. One of the main reasons for choosing the finite-volume method as opposed to other numerical methods, is that the finite-volume solution will converge to the unique entropy solution. This fact is guaranteed by the famous Lax–Wendroff theorem (Theorem 3.1 presented later) under some assumptions on the scheme.

We first consider a single road with endpoints at x_L and x_R. The first step of the finite-volume method is to discretize the spatio-temporal domain $[x_L, x_R] \times [0, T]$ in which we seek the solution. For simplicity the spatial domain of the road $[x_L, x_R]$ is discretized uniformly with mesh size Δx. We also need some temporal discretization for the time interval $[0, T]$ for some terminal time T. Although the discretization in time could also have been chosen to be uniform, we will take the time step to be as large as possible to speed up the simulation process. The maximum value of this time step follows from some conditions that we will come back to later.

Following a convention similar to that of [15], we divide the spatial domain $[x_L, x_R]$ into $N+1$ segments of uniform length, where the length of each segment is defined as $\Delta x = \frac{x_R - x_L}{N+1}$. We denote the points at the edges of these segments as $x_{j-\frac{1}{2}} = x_L + j\Delta x$ for $j = 0, \ldots, N+1$. The left and right edges of segment j will be $x_{j-\frac{1}{2}}$ and $x_{j+\frac{1}{2}}$, respectively. We will refer to each of the segments as computational cells, or *control volumes*, where each control volume is defined as

$$C_j = [x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}}), \text{ for } j = 0, ..., N.$$

The finite-volume method makes uses of the control volumes, in contrast to the finite-difference method, which makes use of the grid points x_j directly. A visualization of a portion of such a grid is shown in Figure 2.

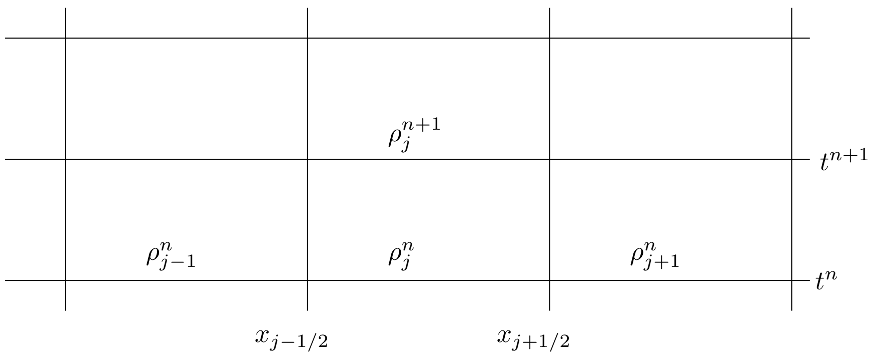


Figure 2: Example of grid that is used in a finite-volume method.

As we explained in the previous chapter, solutions of conservation laws will not be continuous in general, and point evaluations of the solution might thus not make sense. Therefore, a finite difference method that aims to approximate point values of the solution may not be very well suited. The finite-volume method considers instead the average of the solution on a control volume. At each time level t^n, this cell average is approximated as

$$\rho_j^n \approx \frac{1}{\Delta x} \int_{x_{j-1/2}}^{x_j+1/2} \rho(x, t^n)\, dx.$$

The benefit of this change in perspective is that the cell averages are defined for any integrable functions, and therefore will be defined for solutions to the conservation law.

The finite-volume method is a sequential procedure that aims to update the cell average at every step, using the approximation at the previous step. The initial cell averages are calculated as

$$\rho_j^0 = \frac{1}{\Delta x} \int_{x_{j-1/2}}^{x_j+1/2} \rho_0(x)\, dx,$$

where $\rho_0(x)$ defines the initial value of the solution.

Assuming that the cell averages ρ_j^n are known, we need some method to obtain the cell averages at the next time level t^{n+1}. To this end, we integrate the conservation law over the cell and in time. In this way, the cell average at the next time level can be computed. The integral equation reads

$$\int_{t^n}^{t^{n+1}} \int_{x_{j-1/2}}^{x_{j+1/2}} \rho_t\, dxdt + \gamma \int_{t^n}^{t^{n+1}} \int_{x_{j-1/2}}^{x_{j+1/2}} f(\rho)_x\, dxdt = 0.$$

We apply the fundamental theorem of calculus to simplify the equation, giving

$$\begin{aligned}\int_{x_{j-1/2}}^{x_{j+1/2}} \rho(x, t^{n+1})\, dx - \int_{x_{j-1/2}}^{x_{j+1/2}} \rho(x, t^n)\, dx &= \\ - \gamma \int_{t^n}^{t^{n+1}} f(\rho(x_{j+1/2}, t))dt &+ \gamma \int_{t^n}^{t^{n+1}} f(\rho(x_{j-1/2}, t)). \end{aligned} \tag{19}$$

For shorter notation, we define the time average of the flux over the cell as

$$\bar{F}_{j+1/2}^n = \frac{1}{\Delta t} \int_{t^n}^{t^{n+1}} f(\rho(x_{j+1/2}, t))dt,$$

and rewrite (19) as

$$\rho_j^{n+1} = \rho_j^n - \frac{\gamma \Delta t}{\Delta x} (\bar{F}_{j+1/2}^n - \bar{F}_{j-1/2}^n), \tag{20}$$

where again ρ_j^n denotes the cell average of the density.

From (20), it is apparent that as long as the fluxes $\bar{F}_{j+1/2}^n$ and $\bar{F}_{j-1/2}^n$ are known, the cell averages of the densities can be calculated. What remains is to establish some way of approximating these fluxes. The following sections will look into some different approximation methods.

3.1 Godunov's Method

With the goal of approximating the fluxes in (20), Godunov realized that since the cell averages ρ_j^n are constant, each interface between cells defines a Riemann problem. This Riemann problem can be written as

$$\rho_t + \gamma f(\rho)_x = 0, \qquad \rho(x, t^n) = \begin{cases} \rho_j^n & \text{if } x < x_{j+1/2}, \\ \rho_{j+1}^n & \text{if } x > x_{j+1/2}. \end{cases}$$

In its original form, the Godunov method relies on solving this Riemann problem exactly. The exact solution can be found in some cases using the method of the characteristics. The solution will consist of shock, rarefaction and compound wave with some speed. If the time step is chosen to be too large, waves from neighbouring interfaces will start to interact. To avoid this effect, the length of each time step is bounded. For any Riemann problem, the maximum speed of propagation is bounded from above by

$$\max_j |f'(\rho_j^n)|.$$

Thus, to ensure that waves from neighbouring cell interfaces do not intersect, it is sufficient to impose the Courant–Friedrichs–Lewy (CFL) condition

$$\max_j |f'(\rho_j^n)| \frac{\Delta t}{\Delta x} \leq \frac{1}{2}. \tag{21}$$

This condition will be used when choosing the timestep Δt.

3.2 Lax–Friedrichs

Instead of calculating the numerical fluxes exactly, which may be complicated, it is possible to approximate it. One such way is to replace the exact solution with two waves, one travelling to the left of the interface with speed $s^l_{j+1/2}$ and another with speed $s^r_{j+1/2}$.

We then get an approximation of the Riemann problem

$$\rho(x,t) = \begin{cases} \rho^n_j & \text{if } x < s^l_{j+1/2}t, \\ \rho^*_{j+1/2} & \text{if } s^l_{j+1/2}t < x < s^r_{j+1/2}t, \\ \rho^n_{j+1} & \text{if } x > s^r_{j+1/2}t. \end{cases}$$

The middle state can be determined by local conservation using the Rankine–Hugoniot conditions to arrive at the solution

$$f^*_{j+1/2} = \frac{s^r_{j+1/2}f(\rho^n_j) - s^l_{j+1/2}f(\rho^n_{j+1}) + s^r_{j+1/2}s^l_{j+1/2}(\rho^n_{j+1} - \rho^n_j)}{s^r_{j+1/2} - s^l_{j+1/2}}.$$

If we choose the speeds to be equal but of opposite sign, we get the simplification

$$f^*_{j+1/2} = \frac{f(\rho^n_j) + f(\rho^n_{j+1})}{2} - \frac{s_{j+1/2}}{2}(\rho^n_{j+1} - \rho_j),$$

where $s_{j+1/2} = s^r_{j+1/2} = -s^l_{j+1/2}$.

Then, the numerical flux is given by

$$F_{j+1/2} = F(\rho^n_j, \rho^n_{j+1}) = f^*_{j+1/2}.$$

The only missing ingredient is to the speeds $s_{j+1/2}$. To ensure waves from neighboring Riemann problems do not interact, some limit on wave speeds must be enforced. The maximum speeds allowed are

$$s^l_{j+1/2} = -\frac{\Delta x}{\Delta t}, \qquad s^r_{j+1/2} = \frac{\Delta x}{\Delta t}.$$

Substituting this into the equation above gives the Lax–Friedrichs flux

$$F^n_{j+1/2} = F^{LF}\left(\rho^n_j, \rho^n_{j+1}\right) = \frac{f(\rho^n_j) + f(\rho^n_{j+1})}{2} - \frac{\Delta x}{2\Delta t}(\rho^n_{j+1} - \rho^n_j).$$

3.3 Rusanov

One issue with the Lax–Friedrichs scheme is that it is quite diffusive around shocks (i.e., will smear the discontinuities over several cells). One explanation might be the choice of wave speeds, which were simply chosen to be the maximal allowed value. Instead, it is possible to locally select speeds such that

$$s^r_{j+1/2} = s_{j+1/2}, \qquad s^l_{j+1/2} = -s_{j+1/2},$$

where

$$s_{j+1/2} = \max(|f'(\rho^n_j)|, |f'(\rho^n_{j+1})|).$$

The above choice gives rise to the Rusanov flux, given by

$$F^n_{j+1/2} = F^{\text{Rus}}(\rho^n_j, \rho^n_{j+1}) = \frac{f(\rho^n_j) + f(\rho^n_{j+1})}{2} - \frac{\max(|f'(\rho^n_j)|, |f'(\rho^n_{j+1})|)}{2}(\rho^n_{j+1} - \rho^n_j).$$

3.4 Higher-Order and High Resolution Schemes

The finite-volume schemes considered so far have all been first-order schemes. Now we will look into a second-order scheme. The first idea of constructing a second-order scheme is to make use of

Taylor expansions. If we assume that ρ is a smooth solution of Equation (9), we have the following set of equations

$$\begin{aligned}\rho_t &= -\gamma f(\rho)_x,\\ \rho_{tt} &= -\gamma(f(\rho)_x)_t = -\gamma(f(\rho)_t)_x = -\gamma(f'(\rho)\rho_t)_x = \gamma^2(f'(\rho)f(\rho)_x)_x.\end{aligned} \tag{22}$$

Making use of the identities (22), the Taylor expansion of the exact solution can be written as

$$\begin{aligned}\rho(x_j,t^{n+1}) &= \rho(x_j,t^n) + \Delta t\rho_t(x_j,t^n) + \frac{\Delta t^2}{2}\rho_{tt}(x_j,t^n) + \mathcal{O}(\Delta t^3)\\ &= \rho(x_j,t^n) - \Delta t\gamma f(\rho(x_j,t^n)_x + \frac{\Delta t^2}{2}\gamma^2(f'(\rho(x_j,t^n))f(\rho(x_j,t^n))_x)_x + \mathcal{O}(\Delta t^3).\end{aligned} \tag{23}$$

Aiming to construct a second-order accurate scheme, we approximate the spatial derivatives with second-order accurate central differences as

$$\begin{aligned}f(\rho_j^n)_x &\approx \frac{f(\rho_{j+1}^n)-f(\rho_{j-1}^n)}{2\Delta x},\\ (f'(\rho_j^n)f(\rho_j^n)_x)_x &\approx \frac{1}{\Delta x}\left(f'\left(\frac{\rho_j^n+\rho_{j+1}^n}{2}\right)\left(\frac{f(\rho_{j+1}^n - f(\rho_j^n)}{\Delta x}\right) - f'\left(\frac{\rho_j^n+\rho_{j-1}^n}{2}\right)\left(\frac{f(\rho_j^n)-f(\rho_{j-1}^n)}{\Delta x}\right)\right).\end{aligned}$$

Combining the Taylor expansion with the second-order central difference approximations of the spatial derivatives and defining

$$a_{j+1/2}^n = f'\left(\frac{\rho_j^n+\rho_{j+1}^n}{2}\right),$$

gives the *Lax–Wendroff scheme*

$$\begin{aligned}\rho_j^{n+1} &= \rho_j^n - \frac{\Delta t\gamma}{2\Delta x}(f(\rho_{j+1}^n)-f(\rho_{j-1}^n))\\ &+ \frac{\Delta t^2\gamma^2}{2\Delta x^2}\left(a_{j+1/2}(f(\rho_{j+1}^n)-f(\rho_j^n)) - a_{j-1/2}(f(\rho_j^n)-f(\rho_{j-1}^n))\right).\end{aligned} \tag{24}$$

We can write the Lax–Wendroff scheme in the standard form with the numerical flux function

$$F_{j+1/2}^n = F(\rho_j^n,\rho_{j+1}^n) = \frac{f(\rho_j^n)+f(\rho_{j+1}^n)}{2} - \frac{a_{j+1/2}^n\Delta t\gamma}{2\Delta x}\left(f(\rho_{j+1}^n)-f(\rho_j^n)\right). \tag{25}$$

When constructing the Lax–Wendroff scheme, we assumed that the solution was smooth. Thus, we do not know how well the scheme is able to resolve discontinuities in the solution. Figure 3 shows that when a discontinuity forms, the Lax–Wendroff approximation develops oscillations. This motivates the search for a different construction of a more accurate scheme.

With the aim of eliminating the instabilities of the Lax–Wendroff scheme, we will investigate a different way to obtain a high-resolution method. To this end, we summarize the steps in the derivation of the Godunov scheme. The three steps are as follows:

- **R**econstruction: At time level t^n, we approximate the cell averages by ρ_j^n. In the first-order methods, these averages are realized by a piecewise constant function.
- **E**volution: The function from the previous step is updated in time, either exactly or numerically. This is done by solving the superposition of Riemann problems exactly or numerically.
- **A**veraging: The new cell averages are found by averaging the solution over each control volume.

These steps combine to form the so-called REA algorithm. The first-order schemes described above can be derived from the REA algorithm by applying a suitable evolution step. One major drawback of the REA algorithm we have seen so far is that it only makes use of piecewise constant functions in the reconstruction of the cell averages. One idea for obtaining a high-resolution scheme is to replace this first-order interpolation with a higher-order interpolation. The most natural extension is to

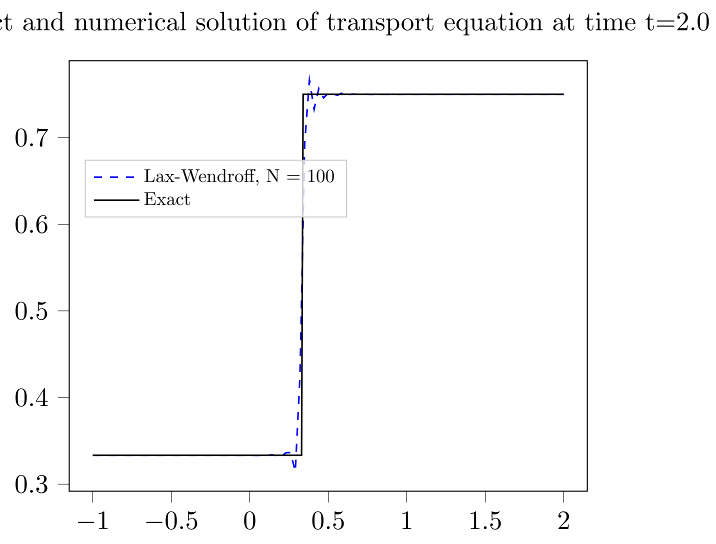


Figure 3: Lax–Wendroff appoximation of (35) at time $t = 2s$.

replace the piecewise constant reconstruction with a piecewise linear reconstruction. However, given the cell averages, there are several approaches for reconstructing such a piecewise linear function, and the slopes needs to be determined.

One key requirement of a stable and convergent approximation is for the scheme to be conservative. Therefore, we require that the reconstruction step is conservative as well. We impose

$$\frac{1}{\Delta x}\int_{x_{j-1/2}}^{x^{j+1/2}} p_j(x)\,dx = \rho_j^n,$$

where $p_j(x)$ denotes the piecewise linear function in the cell. The reconstructed function p_j must therefore be on the form

$$p_j(x) = \rho_j^n + \sigma_j^n(x - x_j),$$

for some slope parameter σ_j^n.

There is a multitude of different ways of picking the slope parameter. Some of the most obvious ones are

- Central:
$$\sigma_j^n = \frac{\rho_{j+1}^n - \rho_{j-1}^n}{2\Delta x}$$
- Backward:
$$\sigma_j^n = \frac{\rho_{j+1}^n - \rho_j^n}{\Delta x}$$
- Forward:
$$\sigma_j^n = \frac{\rho_j^n - \rho_{j-1}^n}{\Delta x}$$

Ideally we would like to be able to choose a slope leading to a second-order accuracy, but as we saw in the Lax–Wendroff scheme, this might lead to non-physical oscillations. Therefore we need to introduce some criteria that we may impose on the slopes to eliminate the oscillations. To this end, we introduce the *total variance* of a function. If Q is a grid function, the total variance can be defined as in [15]

$$\mathrm{TV}(Q) = \sum_{i=-\infty}^{\infty} |Q_i - Q_{i-1}|, \tag{26}$$

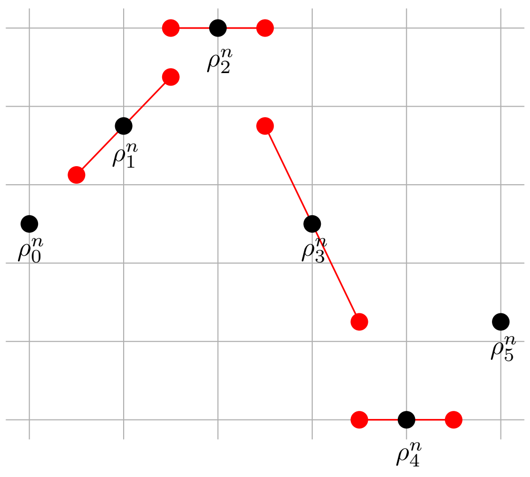


Figure 4: The minmod reconstruction for (28).

or for an arbitrary function $q(x)$,

$$\mathrm{TV}(q) = \sup \sum_{j=1}^{N} |q(\xi_j) - q(\xi_{j-1})|, \tag{27}$$

where the supremum is taken over all possible subdivisions of $\mathbb{R}$, $-\infty = \xi_0 < \xi_1 < \cdots < \xi_N = \infty$. If a numerical method introduces oscillations, we expect the total variation to increase with time. Thus, one idea for eliminating oscillations is to require that the method does not increase the total variation. This gives rise to the so-called limiter, here exemplified by the minmod limiter given by

$$\sigma_j^n = \mathrm{minmod}\Big(\frac{\rho_{j+1}^n - \rho_j^n}{\Delta x}, \frac{\rho_j^n - \rho_{j-1}^n}{\Delta x}\Big),$$

where the minmod function is defined as

$$\mathrm{minmod}(x_1, ..., x_2) := \begin{cases} \mathrm{sign}(x_1)\min_{1\le i\le n}(|x_i|), & \text{if } \mathrm{sign}(x_1) = \cdots = \mathrm{sign}(x_n) \\ 0, & \text{otherwise .} \end{cases}$$

Given a list of elements, the minmod function picks the smallest one in absolute value if the elements are of the same sign. If some of them differ in sign, the minmod function returns zero. Applied to the problem of picking a slope parameter, the minmod limiter compares the upwind and downwind slope. If they are of the same sign, the smallest in absolute value is picked, and if the sign differs, zero is picked.

To visualize how the minmod reconstruction may look like, we consider the cell averages

$$\rho^n = \Big(\rho_0^n, \rho_1^n, \rho_2^n, \rho_3^n, \rho_4^n, \rho_5^n\Big) = \Big(\frac{1}{2}, \frac{3}{4}, 1, \frac{1}{2}, 0, \frac{1}{4}\Big). \tag{28}$$

Figure 4 shows the piecewise linear reconstruction for this case.

Some other limiters that may be used are:

- Superbee:

$$\sigma_j^n = \mathrm{maxmod}(\sigma_j^l, \sigma_j^r),$$

  where maxmod is defined as

$$\mathrm{maxmod}(x_1, ..., x_2) := \begin{cases} \mathrm{sign}(x_1)\max_{1\le i\le n}(|x_i|) & \text{if } \mathrm{sign}(x_1) = ... = \mathrm{sign}(x_n) \\ 0 & \text{otherwise .} \end{cases}$$

$$\sigma_j^l = \text{minmod}\Big(2\frac{\rho_j^n - \rho_{j-1}^n}{\Delta x}, \frac{\rho_{j+1}^n - \rho_j^n}{\Delta x}\Big)$$
$$\sigma_j^r = \text{minmod}\Big(\frac{\rho_j^n - \rho_{j-1}^n}{\Delta x}, 2\frac{\rho_{j+1}^n - \rho_j^n}{\Delta x}\Big).$$

- MC (monotonized central):

$$\sigma_j^n = \text{minmod}\Big(2\frac{\rho_{j+1}^n - \rho_j^n}{\Delta x}, \frac{\rho_{j+1}^n - \rho_{j-1}^n}{2\Delta x}, 2\frac{\rho_j^n - \rho_{j-1}^n}{\Delta x}\Big).$$

To obtain a high-resolution method, we first return to the starting point of the finite-volume method with the cell average denoted as

$$\rho_j(t) = \frac{1}{\Delta x}\int_{x_{j-1/2}}^{x_j+1/2} \rho(x,t)\,dx.$$

Using this notation, we integrate (9) over space; that is,

$$\frac{1}{\Delta x}\int_{x_{j-1/2}}^{x_j+1/2} \big(\rho_t + \gamma f(\rho)_x\big)\,dx = 0.$$

From the integral we obtain

$$\frac{d}{dt}\rho_j(t) + \frac{\gamma}{\Delta x}(f(\rho(x_{j+1/2}^-,t)) - f(\rho(x_{j-1/2}^+,t))) = 0,$$

where

$$\rho\big(x_{j+1/2}^{\pm},t\big) = \lim_{x\to x_{j+1/2}^{\pm}} \rho(x,t).$$

The limits are used in this case, since the reconstruction of the cell averages is a piecewise linear function, leading to possible discontinuities at the cell interfaces.

We now let F approximate the flux terms to arrive at the semi-discrete form

$$\frac{d}{dt}\rho_j(t) + \frac{\gamma}{\Delta x}\big(F_{j+1/2}^-(t) - F_{j-1/2}^+(t)\big) = 0. \tag{29}$$

If a piecewise constant reconstruction had been used, this would reduce to (20).

We will use the Rusanov flux function to get the flux approximation F. The semi-discrete form leads to a system of ODEs that must be integrated in time. This can be done to the first order by using a forward Euler method, or to the second order by utilizing a second-order *strong stability-preserving* (SSP) Runge–Kutta method as in [15].

As discussed before, we want to replace the piecewise constant reconstruction by a piecewise linear reconstruction. We let the linear function in cell C_j be given as

$$p_j(x) = \rho_j + \sigma_j(x - x_j),$$

where σ_j is a suitable slope parameter (minmod, superbee, mc). The piecewise linear function is the combination of the linear functions and has the form

$$p(x,t) = p_j(x) \text{ for } x_{j-1/2} \le x < x_{j+1/2}. \tag{30}$$

Since we no longer have constant functions in each cell in general, we also define the reconstructed values at cell interfaces as

$$\begin{aligned}\rho_j^+ &= \rho_j(x_{j+1/2}),\\ \rho_j^- &= p_j(x_{j-1/2}).\end{aligned} \tag{31}$$

We need to replace the cell averages with relevant values at cell interfaces in the approximation of the fluxes. We then get for the approximate flux, F,

$$F_{j+1/2} = F(\rho_j^+, \rho_{j+1}^-),$$

where F is a *consistent flux function*, like the Rusanov flux function. For a flux function to be called consistent, it needs to satisfy some basic requirements. First, if the function $\rho(x,t) = \bar{\rho}$ is constant, it will not change in time. If the numerical approximation is also constant, $\rho^n_{i-1} = \rho^n_i = \bar{\rho}$, we expect the numerical flux function F to reduce to $f(\bar{\rho})$, that is we require

$$F(\bar{\rho}, \bar{\rho}) = f(\bar{\rho})$$

for any $\bar{\rho}$. In addition to this requirement, F is expected to be *Lipschitz continuous*, e.g. there exists a constant L such that

$$|F(\rho_{i-1}, \rho_i) - f(\bar{\rho})| \leq L \max(|\rho_i - \bar{\rho}|, |\rho_{i-1} - \bar{\rho}|).$$

To simplify notation for the time stepping, we rewrite the semi discrete form as

$$\frac{d}{dt}\boldsymbol{\rho} = \mathcal{L}(\boldsymbol{\rho}),$$

where

$$\mathcal{L}(\boldsymbol{\rho})_j := -\frac{\gamma}{\Delta x}(F^-_{j+1/2}(t) - F^+_{j-1/2}(t))$$

and $\boldsymbol{\rho}$ is a vector containing the cell averages.

A two-stage Strong Stability Preserving (SSP) Runge-Kutta, see e.g. [9], scheme can be written as

$$\begin{aligned} \boldsymbol{\rho}^* &= \boldsymbol{\rho}^n + \Delta t \mathcal{L}(\boldsymbol{\rho}^n), \\ \boldsymbol{\rho}^{**} &= \boldsymbol{\rho}^* + \Delta t \mathcal{L}(\boldsymbol{\rho}^*), \\ \boldsymbol{\rho}^{n+1} &= \frac{1}{2}(\boldsymbol{\rho}^n + \boldsymbol{\rho}^{**}), \end{aligned} \tag{32}$$

where the time step needs to satisfy some CFL condition like (21).

3.5 Boundary Conditions

The derivation of the finite-volume schemes has so far neglected to mention boundary conditions. In this section we will describe how physical boundary conditions, such as Dirichlet, Neumann or periodic boundary conditions, can be implemented for the first- and second-order schemes. As in the start of Section 3, we let $[x_L, x_r]$ be the domain of interest. For a first-order scheme, we need to introduce two new cells. Following the terminology in [15], we introduce the two *ghost cells*:

$$C_{-1} = [x_L - \Delta x, x_L), \text{ and } C_{N+1} = [x_R, x_R + \Delta x).$$

In the case of Dirichlet boundary conditions; that is,

$$\rho(x_L, t) = g_L(t), \text{ and } \rho(x_R, t) = g_R(t),$$

we implement the boundary conditions as

$$\rho^n_{-1} = g_L(t^n), \text{ and } \rho^n_{N+1} = g_R(t^n).$$

Periodic boundary conditions may be implemented as

$$\rho^n_{-1} = \rho^n_N, \text{ and } \rho^n_{N+1} = \rho^n_0.$$

Finally, we may implement non-reflecting Neumann type boundary conditions as

$$\rho^n_{-1} = \rho^n_0, \text{ and } \rho^n_{N+1} = \rho^n_N.$$

For the second-order schemes, we need to introduce one more ghost cell on each side of the domain. We introduce the two new ghost cells

$$C_{-2} = [x_L - 2\Delta x, x_L - \Delta x), \text{ and } C_{N+2} = [x_R + \Delta x, x_R + 2\Delta x).$$

Similar to the first-order schemes, periodic boundary conditions are implemented as

$$\rho_{-2}^n = \rho_{N-1}^n, \quad \rho_{-1}^n = \rho_N^n,$$
$$\rho_{N+1}^n = \rho_0^n, \quad \rho_{N+2}^n = \rho_1^n.$$

For the Dirichlet boundary conditions, one possibility is to set

$$\rho_{-2}^n = g_L(t^n), \quad \rho_{-1}^n = \rho_{-2}^n,$$
$$\rho_{N+2}^n = g_R(t^n), \quad \rho_{N+1}^n = \rho_{N+1}^n.$$

Finally, we can implementing non-reflecting Neumann type boundary conditions as

$$\rho_{-2}^n = \rho_{-1}^n = \rho_0^n, \text{ and } \rho_{N+2}^n = \rho_{N+1}^n = \rho_N^n.$$

If we want to make use of an even higher-order scheme, more ghost cells need to be defined in a similar fashion.

3.6 Convergence Analysis

One of the main criteria for judging the quality of a numerical approximation, is that it should converge to the true solution as the grid is refined, i.e., as $\Delta x, \Delta t \to 0$. This generally requires that the method is *consistent* and *stable*, see, e.g., [15]. A consistent method will have good local approximations, and a stable method ensures that small errors from each time step will not grow too large.

To be able to properly quantify the error, we need to decide a norm in which to measure the error. Given an L^1 function ρ_{exact} and an L^1 approximation ρ, we define the continuous error simply as

$$e(x) = \rho_{\text{exact}}(x) - \rho(x).$$

Similarly, given a vector with n elements $\boldsymbol{\rho}_{\text{exact}}$ and the approximation $\boldsymbol{\rho}$, we define the discrete error as

$$E = \boldsymbol{\rho}_{\text{exact}} - \boldsymbol{\rho},$$

meaning that we can write the ith component of E as

$$E_i = (\boldsymbol{\rho}_{\text{exact}})_i - \boldsymbol{\rho}_i.$$

To quantify the continuous error, we can use the L^1-norm

$$\|e\|_{L^1} = \int_{-\infty}^{\infty} |e(x)|\, dx. \tag{33}$$

Assuming the vector E corresponds to a cell-wise constant approximation of the error, with grid spacing Δx, we can quantify the discrete error as

$$\|E\|_{\ell^1} = \Delta x \sum_{i=1}^{n} |E_i|. \tag{34}$$

Following [15], we say that a method is convergent at time T if

$$\lim_{\Delta t, \Delta x \to 0} \|E^N\|_{\ell^1} = 0,$$

where $N\Delta t = T$. We also say that a method is *accurate of order* γ at time T if

$$\|E^N\|_{\ell^1} = O(\Delta t^\gamma).$$

The famous Lax–Wendroff theorem stated in [15] guarantees that whenever there is convergence, the method will converge to the solution of the problem.

Theorem 3.1 (Lax–Wendroff)**.** Consider a sequence of grids indexes by $j = 1, 2, \dots$ with mesh parameters $\Delta t^j, \Delta x^j \to 0$ as $j \to \infty$. Let $\rho^j(x,t)$ denote the numerical approximation computed with a consistent and conservative method on the jth grid. Suppose that ρ^j converges to a function ρ as $j \to \infty$, in the sense discussed above. Then $\rho(x,t)$ is a weak solution of the conservation law.

In some cases it is also possible to show convergence of a finite-volume scheme; see, e.g., Theorem 4, and Theorem 6 presented by Fjordholm and Lye [4], or Example 3.17 in the book by Holden and Risebro [13]. These theorems and the example show that in a large class of problems, notably those with so-called bounded initial total variation it is possible to guarantee a convergence rate of 1/2.

We will now investigate the experimental convergence order of some different schemes for a few simplified problems. To, this end we compare numerical results with analytical results for various choices of step lengths Δx and Δt.

3.6.1 Example 1

Consider the equation

$$\rho_t + \big(\rho(1-\rho)\big)_x = \rho_t + (1-2\rho)\rho_x = 0. \tag{35}$$

Let ρ be defined on the domain $[-1,2] \times \mathbb{R}^+$ and the initial data be given as

$$\rho(x,0) = \rho_0(x) = \begin{cases} \rho^L & \text{if } x \le 0, \\ \rho^L + (\rho^R - \rho^L)x & \text{if } 0 < x \le 1 \\ \rho^R & \text{if } 1 < x. \end{cases}$$

Section A in the appendix considers the analytical solution for this problem both in the general case, and for the specific case $\rho^L = \frac{1}{3}$ and $\rho^R = \frac{3}{4}$.

With an explicit form of the solution in place, we are ready to compare the numerical results with the analytical results. In Figure 5, the numerical approximations are compared to the exact solution at two times. The second-order schemes make use of the minmod and superbee limiters for the reconstruction step. The plot also shows the results for the first-order Rusanov scheme. As the step length decresaes, the approximations improve.

To find out how quickly the approximates improve, we are interested in the order of convergence. To this end, we first define the relative error

$$\epsilon_{\Delta x} = 100 \cdot \frac{\|\rho^{\Delta x} - \rho^{\text{ref}}\|_{\ell^1}}{\|\rho^{\text{ref}}\|_{\ell^1}},$$

where the ℓ^1-norm needs to be estimated. The reference solution ρ^{ref} could either be an analytical density if available, or a solution estimated on a finer grid. If ρ^{ref} is estimated using a finer grid, one might find a too high order of convergence when the grid size approaches the grid size used when calculating ρ^{ref}.

With the relative error defined, we can estimate the order of convergence as

$$\text{EOC}_{\Delta x, \Delta y} = \frac{\log(\epsilon_{\Delta x}) - \log(\epsilon_{\Delta y})}{\log(\Delta x) - \log(\Delta y)},$$

where Δx and Δy are two different mesh sizes.

Table 1 shows the estimated order of convergence of the solution for the transport equation with initial data (71) at time $t = 2.0$ for some different schemes. Even though the schemes using the minmod and superbee limiters were constructed to be more accurate than the first-order Rusanov scheme, the results show that all three schemes have an experimental order of convergence close to one. However, even though all three schemes appear to have a similar order of convergence, the high-resolution methods with the minmod and superbee limiters seem to outperform the Rusanov

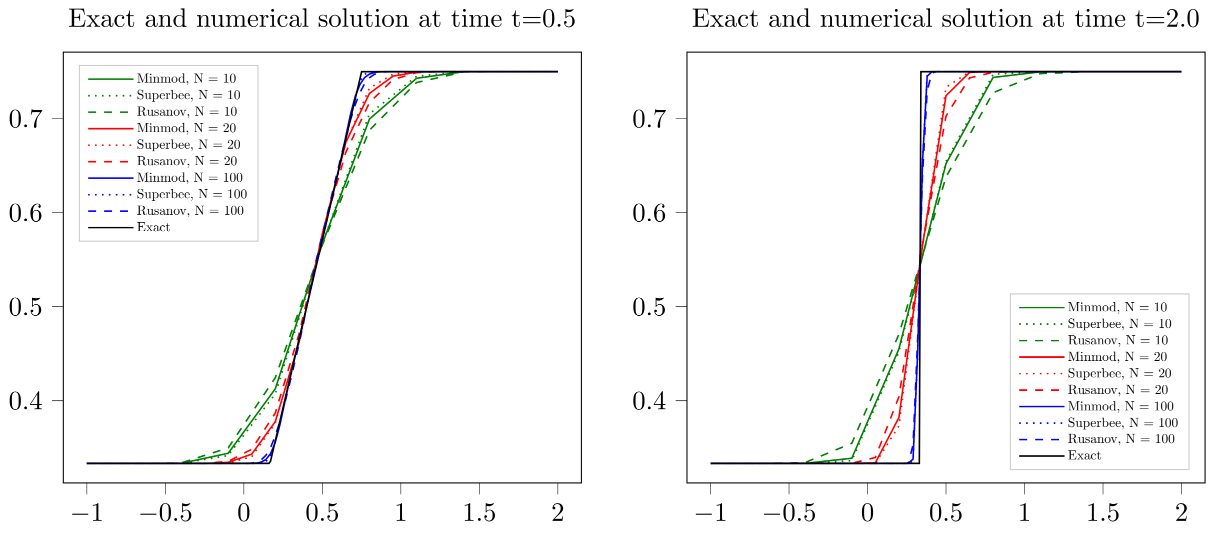


Figure 5: The figure shows the numerical approximations to the solution of (35) for different choices of stepsize Δx. Two different limiters, the minmod and superbee limiters, are compared as well as the first-order Rusanov scheme. The two plots show that the approximation improves as the step length decreases. It also shows that the second-order scheme with either the minmod or superbee limiters seems to outperform the first-order Rusanov scheme slightly.

Table 1: Estimated order of convergence estimated using the minmod and the superbee limiter and the first-order Rusanov scheme. Results are from the transport equation (35) with initial data (71) at time $t = 2.0$.

No. of cells	**Minmod**		**Superbee**		**Rusanov**	
	$\epsilon_{\Delta x}$	**EOC**	$\epsilon_{\Delta x}$	**EOC**	$\epsilon_{\Delta x}$	**EOC**
10	3.7528	—	3.5611	—	4.7978	—
20	2.1742	0.9837	2.0138	0.9809	2.6384	0.9845
40	1.0636	0.7875	0.9863	0.8224	1.3009	0.8627
80	0.5027	1.0316	0.4757	1.0298	0.6329	1.0202
160	0.2777	1.0811	0.2583	1.0519	0.3387	1.0395
320	0.1322	0.8562	0.1232	0.8811	0.1627	0.9019
640	0.0628	1.0709	0.0602	1.0683	0.0800	1.0583

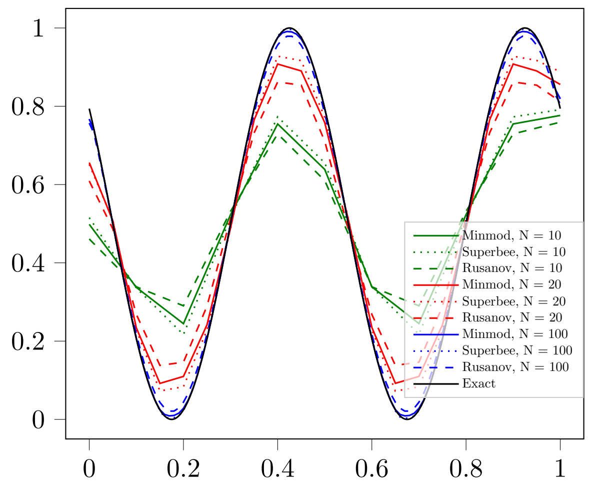


Figure 6: Solution of the linear advection equation at time $t = 0.5$.

Table 2: Estimated order of convergence using the minmod and the superbee limiter and the first-order Rusanov scheme. Results are from the linear advection equation (36) with initial data (37) at time $t = 0.5$.

No. of cells	**Minmod**		**Superbee**		**Rusanov**	
	$\epsilon_{\Delta x}$	**EOC**	$\epsilon_{\Delta x}$	**EOC**	$\epsilon_{\Delta x}$	**EOC**
10	28.6505	—	25.9513	—	32.4147	—
20	9.9489	1.8358	8.0593	1.7965	15.3087	1.0823
40	2.7811	1.5259	2.1108	1.6871	6.9573	1.0823
80	0.6883	1.8389	0.6647	1.9329	3.1906	1.1247
160	0.1775	2.0145	0.1990	1.6669	1.5091	1.1377
320	0.0487	1.9551	0.0551	1.7397	0.7313	1.0801
640	0.0138	1.8660	0.0148	1.8530	0.3596	1.0452

slightly for all choices of grid spacing. The difference between the two limiters does not appear to be very big. This is also support by Figure 5, which shows that the Rusanov scheme performs slightly worse for all grid sizes compared to the two second-order methods.

3.6.2 Example 2

To check if the high-resolution schemes obtain a higher order of convergence when the initial data and the solution are smooth, we consider the linear advection equation

$$\rho_t + a\rho_x = 0, \tag{36}$$

where $a \in \mathbb{R}$. We let the initial data be given as

$$\rho(x,0) = \rho_0(x) = \frac{1}{2}(1 - \sin(4\pi x)). \tag{37}$$

Using the method of characteristics as before, the solution at position (x,t) is calculated as

$$\rho(x,t) = \rho_0(x - at).$$

Table 2 shows the estimated order of convergence for the second-order scheme using the minmod and the superbee limiters for the linear advection equation (36) with smooth initial data (37). The

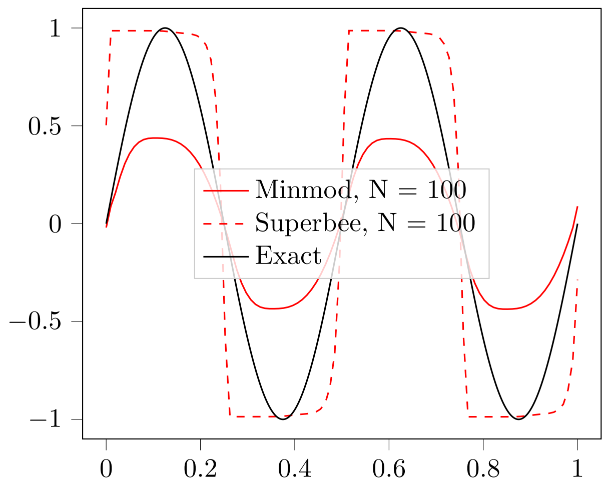


Figure 7: Numerical approximation of linear advection equation at time $t = 10$ for the minmod and superbee limiters.

results show that the second-order scheme perform similarly for the two limiters. It also shows that the experimental order of convergence is close to two for the two second-order schemes, and close to one for the Rusanov scheme. The results are supported by Figure 6, which shows the numerical approximations of the linear advection equation for some different grid sizes at time $t = 0.5$. Results from the transport equation and the linear advection equation indicate that the superbee limiter performs slightly better than the minmod limiter for this example. As the grid gets finer, the differences become smaller, with the minmod outperforming the superbee limiter in some cases. Combining these results with the results from Example 3.6.1, we see that the high-resolution schemes are able to resolve discontinuities without oscilliations, while also being able to approximate smooth solutions to second-order.

3.6.3 Example 3

In the two previous it was difficult to see any noticeable difference between the minmod and the superbee limiters. In this example we will look at a case showing some of the difference between them. Once more, we consider the linear advection (36), but this time we let the simulation run for a longer time period than we did in the previous example. Figure 7 show the difference between the minmod and the superbee limiters when we simulate untill time $t = 10s$. We see that the amplitude is significantly dampened when using the minmod limiter, and that the superbee limiter tightens the smooth solution to a more square profile.

4 Automatic Differentiation

As we have mentioned in the introduction, the goal is not only to create a realistic model for the flow of traffic, but also to be able to choose the best possible parameters optimizing the flow of traffic in the road network. To reduce the number of iterations needed for the optimization algorithm, a gradient-based optimization technique will be used. However, this introduces the problem of being able to accurately and efficiently evaluate or approximate the derivatives of some objective function describing the flow with respect to a set of parameters we want to optimize. The solution to this problem comes in the form of automatic differentiation. Automatic differentiation is a techinque for calculating the derivatives of any degree of a numeric function which is defined programatically. Verma [22] and Harrison [11] describe some of the basics of automatic differentiation.

4.1 Basics of AD

As mentioned, automatic differentiation (AD) is a technique for evaluating derivatives that is based on the chain rule. AD exploits the fact all computer operations can be broken down to a finite set of elementary operations like $\sin(\cdot)$, $\sqrt{\cdot}$, +, -, $\cdot$, etc. Since the set of elementary operations is finite, the partial derivatives of these can be stored. By using the chain rule to break any program down to a composition of the partial derivatives, the overall derivative can be calculated. It should be noted that AD does not produce derivative expressions, but instead produces numerical values of the derivative. AD is accurate to machine precision, meaning that if calculations could be performed in real arithmetic, the results would be exact.

AD has two modes: the forward and the backward mode. In the forward mode, the derivatives are propagated forward using the chain rule. Every time a new operation is performed, its derivative is calculated, and at the end, the final derivative will be calculated. At each step, the new derivative is calculated using the derivative from the previous step.

In the backward mode, the derivatives are evaluated in reverse order, more closely related to the intuition of the chain rule. meaning that intermediate derivatives will have to be stored in memory.

4.2 Implementation of AD

In recent years, the use of deep learning modelling techniques has become increasingly popular. This modelling technique involves the construction of a large neural network, containing sometimes millions of parameters. To fit these parameters to a data-set, typically a gradient based optimization is applied. To this end, several frameworks that employ AD have been developed, such as PyTorch [20]. The PyTorch framework overloads primitive operations so that they perform an evaluation trace and save intermediate values that are needed to obtain the derivatives. Most of the framework is implemented in C++, leading to a more efficient run-time than pure Python library. The parameters that the gradient should be calculated with respect need to be specified before running the simulation. In our case, these parameters will be the speed limit of each road, as well as some control parameters of traffic lights, that we will discuss in detail in Section 5.2.

4.3 Validation of AD

Instead of blindly relying on results from AD, some sanity check on the results is useful. It would be infeasible to manually calculate the derivative of the full scheme, so some simpler cases need to be considered. Two such cases will be discussed below. A final check of the AD derivatives is related to the optimization procedure. If gradient-based optimization routines are not able to find local extrema employing the gradient evaluated using AD, the logical conclusion is that the AD results must be wrong. This final check of AD will be discussed in more detail later.

4.3.1 Constant density on road

One method of validating that the AD derivative is correct is to consider a simple case where the analytical derivative can be calculated. One such case is a single road where the initial density of cars is constant in space; that is,

$$\rho_t + \big(v\rho(1-\rho)\big)_x = 0, \quad (x,t) \in [0,1] \times [0,T), \qquad \rho(x,0) = 0.5. \tag{38}$$

An objective function $G : (0,\infty) \to \mathbb{R}$ can be defined by

$$G(v) = v\big(1 - \rho(0.5, 0.5)\big),$$

with analytical derivative given as

$$\frac{\partial}{\partial v} G = 1 - \rho(0.5, 0.5).$$

We perform the test using the high-resolution scheme with the Rusanov flux function. The time stepping was done by a SSP Runge-Kutta method, and the limiter used for the reconstruction of the cell averages was the minmod-limiter- In this test, the AD derivative and the analytical one were found to be equal. It should be noted that in this simple case $\rho(0.5, 0.5)$ is independent of the speed, which is not true in general.

4.3.2 Manually calculating derivatives

Another way to validate the AD derivative is by performing only one time step of the scheme both manually and using AD. If the objective function, e.g., is the density at a single node at a given time, the derivative of this value with respect to the speed limit can be calculated manually without too much work. After the derivative is calculated manually, this value can be compared to the one obtained from automatic differentiation.

In this case, we consider a single road of length $L = 1$ with speed limit $v_{\max} = 10$. The scheme considered will be a second order in space Euler scheme with Rusanov flux function and the minmomod limiter. The initial density will be a linear function between 0 and 0.5. The density after one time step will then be

$$\rho^1 = \rho^0 + \Delta t \mathcal{L}(\rho^0).$$

We consider the case with three interior nodes and $\Delta x = \frac{1}{4}$. In this example we choose the time-step as $\Delta t = \frac{1}{40}$, independently of the speed limit. However, in later simulations the time-step will be chosen using the CFL condition (21), meaning that Δt will also depend on the speed limit. As we said, the slope used in the reconstruction step will be chosen using the minmod delimiter. The initial density, including the boundaries, is given as $\rho^0 = (0, \frac{1}{12}, \frac{1}{6}, \frac{1}{4}, \frac{5}{12}, \frac{1}{2})^T$. Manually calculating the first timestep gives for the interior points

$$\rho^1_{interior} = \left(\frac{1}{6} - \frac{\Delta t}{2\Delta x} v_{\max} \frac{5}{36}, \frac{1}{4} - \frac{\Delta t}{2\Delta x} v_{\max} \frac{1}{9}, \frac{1}{3} - \frac{\Delta t}{2\Delta x} v_{\max} \frac{1}{12}\right)^T = \left(\frac{7}{72}, \frac{7}{36}, \frac{7}{24}\right)^T.$$

We now consider the objective function $f(v_{\max}) = \rho^1_4 v_{\max}$. The derivatives were manually calculated as

$$\frac{df(v_{\max})}{dv_{\max}} = \frac{1}{4} - \frac{\Delta t}{9\Delta x} v_{\max} = 0.1389$$

and

$$\frac{d^2 f(v_{\max})}{dv^2_{\max}} = -\frac{\Delta t}{9\Delta x} = -0.011.$$

Both of the derivatives were found to match with the evaluation of the derivatives from the PyTorch AD, indicating that the AD is implemented correctly.

5 Model Considerations

The models described in Section 2 were derived with a single unidirectional road in mind. To be able to model more complex road networks, these models will need to be extended. This section will describe how the models in Section 2 can be extended to include junctions and traffic lights. The speed limits on each road will also be allowed to vary in time. Finally, some of the limitations of the modelling approached being used will be considered.

5.1 Junctions

One of the interesting extension of the first-order LWR model from Section 2.1.1 is the inclusion of a junction with n incoming roads and m outgoing roads. Assume that roads $1, \dots, n$ enter the junction and that roads $n+1,\dots,n+m$ leave the junction. Let the junction be denoted by J. Let road i have end points at a_i and b_i. Then for each junction J, we have

$$\sum_{i=1}^{n} f\big(\rho(b_i,\cdot)\big) = \sum_{i=n+1}^{n+m} f\big(\rho_i(a_i,\cdot)\big), \tag{39}$$

where ρ_i is the density of cars on road i. The above equation simply states that all cars that enter a junction, will also leave the junction. That is, there is no accumulation of cars in the junction. This condition can be called the Rankine–Hugoniot condition for junctions. However, using only the Rankine–Hugoniot condition on its own does not lead to a unique solution.

5.1.1 Entropy condition

With the aim of finding a unique solution not only for interior points but also across the junction, we can impose a so-called entropy condition for the flux across the junction as done by Holden and Risebro [12]. To this end, we first write the solution at the junction as

$$\bar{\rho} = (\bar{\rho}_1, \bar{\rho}_2) \in \mathbb{R}^{n+m}, \tag{40}$$

where

$$\begin{aligned}\bar{\rho}_1 &= \big(\rho_1(b_1,\cdot),\dots,\rho_n(b_n,\cdot)\big),\\ \bar{\rho}_2 &= \big(\rho_{n+1}(a_{n+1},\cdot),\dots,\rho_{n+m}(a_{n+m},\cdot)\big).\end{aligned}$$

Further, we define the set of possible densities using the Rankine–Hugoniot condition as

$$H_J = \Big\{\bar{\rho} \in \mathbb{R}^{n+m} : \sum_{i=1}^{n} f(\rho_i(c_i,\cdot)) = \sum_{i=n+1}^{n+m} f(\rho_i(c_i,\cdot))\Big\}. \tag{41}$$

Finally, we need the entropy of the junction. Let g be a strictly concave and differentiable function on $\mathbb{R}$. Then, the entropy of the junction reads

$$E_J = \sum_{i=1}^{n+m} g\Big(\frac{f_i}{f_{max}}\Big), \quad f_i = f\big(\rho_i(c_i)\big). \tag{42}$$

In the context of traffic modelling, it is reasonable to assume that the function f has a unique maximum,

$$f(0) = f(1) = 0, \quad \exists \sigma \in (0,1) : f'(\sigma) = 0, \quad (\rho - \sigma) f'(\rho) < \rho, \quad \rho \neq \sigma. \tag{43}$$

Assuming that f satisfies Equation (43), the entropy condition can be stated as

> Find $\bar{\rho} = (\rho_1,\dots,\rho_{n+m}) \in H_J$ that maximizes E_j such that $\rho_i \in [\sigma, 1]$ for $i = 1,\dots,n$, and $\rho_i \in [0,\sigma]$ for $i = n+1,\dots,n+m$. (44)

Holden and Risbro [12] looked at existence and uniqueness of a solution to the entropy problem, and showed that a unique solution of the Cauchy problem can be found.

5.1.2 Demand and supply

A different approach for finding the fluxes at junctions is inspired by [6]. We again consider a junction with n incoming roads and m outgoing roads, and assume that roads $i \in \mathcal{I}$, are entering the junction and that roads $i \in \mathcal{O}$ are leaving the junction, where $\mathcal{I} \cap \mathcal{O} = \emptyset$ and $\mathcal{I} \neq \emptyset \neq \mathcal{O}$. Suppose that each road i has domain $[a_i, b_i]$, where ends may extend to infinity. As in [12], we impose the Rankine–Hugoniot condition (39) for the junction.

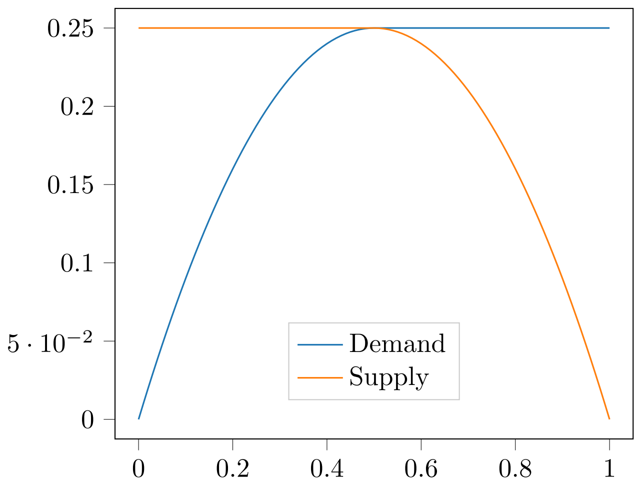


Figure 8: Demand and supply functions for a single road

To distribute the flux of cars across the junction, we first define the some distribution parameters $\alpha_{i,j}$ for $i \in \mathcal{I}$, $j \in \mathcal{O}$, and some priority parameters $\beta_{i,j}$ for $i \in \mathcal{I}$, $j \in \mathcal{O}$. The distribution parameters describe how cars from each incoming road divide into the outgoing roads. The priority parameters describe where cars are coming from for each outgoing road.

Since there are no sources or sinks at the junction, the distribution and priority parameters need to satisfy

$$\sum_{j \in \mathcal{O}} \alpha_{i,j} = 1 \;\; \forall i \in \mathcal{I}, \tag{45}$$

and

$$\sum_{i \in \mathcal{I}} \beta_{i,j} = 1 \;\; \forall j \in \mathcal{O}. \tag{46}$$

Like in [6], we also define the demand and supply functions as

$$\begin{aligned} D_i(\rho) &= \begin{cases} f_i(\rho) & \text{if } \rho \leq \sigma_i, \\ f_i^{\max} & \text{otherwise,} \end{cases} \\ S_i(\rho) &= \begin{cases} f_i^{\max} & \text{if } \rho \leq \sigma_i, \\ f_i(\rho) & \text{otherwise,} \end{cases} \end{aligned} \tag{47}$$

where σ_i satisfies $\operatorname{argmax}_\rho f_i(\rho) = \sigma_i$. A visualization of the demand and supply functions can be seen in Figure 8. These functions are used to describe the maximal flux out of a road, and the maximal flux leading in to a road.

It should be noted that this model allows for different flux functions f_i for each road, but if the maximum capacity on each road is different, the flux functions should be scaled to take this into account. The demand describes the maximal flux out of a road, and the supply function describes the maximal flux into a road. These functions will be used to divide the total flux of cars across the junction.

The flux from road i to road j is computed as the minimum of the maximal flux out of road i and the maximal flux into road j. With the distribution and priority parameters, we set the maximal flux out to be $\alpha_{i,j} D_i(\rho_i(b_i, \cdot))$ and the maximal flux in to be $\beta_{i,j} S_j(\rho_j(a_j, \cdot))$.

Thus, the flux from road i to road j can be computed as

$$\hat{f}_{i,j} = \min\{\alpha_{i,j} D_i(\rho_i(b_i, \cdot)), \beta_{i,j} S_j(\rho_j(a_j, \cdot))\}. \tag{48}$$

Summing over all the outgoing roads, the total flux out from road i is computed as

$$\hat{f}_i = \sum_{j \in \mathcal{O}} \hat{f}_{i,j}. \tag{49}$$

Similarly, summing over all the incoming roads, the total flux into road j is computed as

$$\hat{f}_j = \sum_{i \in \mathcal{I}} \hat{f}_{i,j}. \tag{50}$$

With this formulation, the Rankine–Hugoniot condition will be trivially satisfied, since

$$\sum_{i \in \mathcal{I}} \hat{f}_i = \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{O}} \hat{f}_{i,j} = \sum_{j \in \mathcal{O}} \sum_{i \in \mathcal{I}} \hat{f}_{i,j} = \sum_{j \in \mathcal{O}} \hat{f}_j.$$

What remains is to determine the distribution and priority parameters. One method is to choose $\alpha_{i,j}$ and $\beta_{i,j}$ to maximise the total flux, given as

$$f_{\text{tot}} = \sum_{i \in \mathcal{I}} \hat{f}_i. \tag{51}$$

However, simply maximizing the flux across the junction might not give an accurate representation of the real word. What is more likely is that more drivers will pick one road over another, e.g., if the junction is between a freeway and a road leading out to a small residential area, most drivers will pick the freeway. Thus, specifying the distribution parameters when creating the model will give a more realistic simulation.

Suppose that the coefficients $\alpha_{i,j}$ and $\beta_{i,j}$ are determined. What remains is then to impose some boundary conditions. From the coefficients, the flux out and in of each road can be determined. This flux can then be used as one the numerical fluxes in (20). The other numerical flux at the cell interface could come, e.g., from the Rusanov flux function.

5.1.3 Remarks

We note that whenever $\alpha_{i,j} D_i(\rho_i(b_i, \cdot)) \neq \beta_{i,j} S_j(\rho_j(a_j, \cdot))$, we could potentially set different fluxes across the junction. If $\alpha_{i,j} D_i(\rho_i(b_i, \cdot)) > \beta_{i,j} S_j(\rho_j(a_j, \cdot))$, road i will be able to send out more flux than it actually does. This flux could, e.g., be sent to a different road instead. If instead $\alpha_{i,j} D_i(\rho_i(b_i, \cdot)) < \beta_{i,j} S_j(\rho_j(a_j, \cdot))$, road j will have capacity to take in more flux than it gets. In this case, some of the flux from other roads could be sent into this road instead.

We will now visualize this effect with an example. Consider the case of a junction with two incoming roads and two outgoing roads. Let the roads be ordered such that roads 1 and 2 are entering and roads 3 and 4 are leaving the junction. Suppose that $\rho_1(b_1, \cdot) = 0.9$ and that $\rho_2(b_2, \cdot) = 0.1$, such that $D_1(\rho_1(b_1, \cdot)) = 0.25$ and $D_2(\rho_2(b_2, \cdot)) = 0.09$, with flux function $f(\rho) = \rho(1 - \rho)$. We further assume that $\rho_3(a_3, \cdot) = 0.9$ and that $\rho_4(a_4, \cdot) = 0.1$ such that $S_3(\rho_3(a_3, \cdot)) = 0.09$ and $S_4(\rho_4(b_4, \cdot)) = 0.25$

We now let the distribution and priority parameters be defined as $\alpha_{1,3} = 0.9$, $\alpha_{1,4} = 0.1$, $\alpha_{2,3} = 0.1$, $\alpha_{2,4} = 0.9$ and $\beta_{1,3} = 0.1$, $\beta_{1,4} = 0.9$, $\beta_{2,3} = 0.9$, $\beta_{2,4} = 0.1$.

Calculating the fluxes using these assumptions, we get $f_{1,3} = f_{2,3} = 0.009$, and $f_{1,4} = f_{2,4} = 0.025$, giving a total flux across the junction of $f_{\text{tot}} = 0.068$. The maximum flux into to the junction is equal to $f^{\text{in}}_{\text{tot}} = 0.34$ and the maximum flux out of the junction is equal to $f^{\text{in}}_{\text{tot}} = 0.34$. In other words, only a small portion of the potential flux across the junction actually goes through.

In some cases this might be realistic. A driver would not take off from a congested freeway to drive onto a road leading to the countryside simply because this road is empty. On the other hand, some roads might lead to the same destination, with one road potentially being slightly longer. Even though drivers in general would prefer the shorter route, if there is heavy traffic along the shorter route, more drivers would prefer the longer and less congested route. In general, however, it is not a good idea to adjust the flux out of a road in this way.

On the other side, it might be a good idea to adjust the flux into a road. Even though some roads might have priority of others, if there is not a lot of traffic coming from the prioritized road, the road with less priority should be allowed to send more traffic through the junction. The model should be extended to allow for this to get a more realistic representation of traffic.

In other words, it might be a good idea to assume that the distribution parameters are known beforehand and to solve a small optimization problem to obtain the correct priority parameters.

5.1.4 Optimal choice of priority parameters

As mentioned in the previous subsection, it might be a good idea to solve a small optimization problem in each timestep to obtain the priority parameters. In this section, we describe this optimization problem and the solution approach we have used.

Although drivers have a preference for which road they choose, each road does not in general have any preference for where cars are coming from. Therefore, to best utilize the capacities of the roads leading away from the junction, we want to find the priority parameters maximizing the flux across the junction. We write the optimization problem as

$$\max_{\{\beta_{i,j}\}_{i,j}} f_{\text{tot}} \text{ s.t. } \sum_{i=1}^{n} \beta_{i,j} = 1 \qquad \forall j \in \{n+1, \ldots, n+m\} \text{ and } \beta_{i,j} \geq 0. \tag{52}$$

Assuming that the distribution parameters are given and that the densities at each road is known, this optimization takes the form of a simple linear optimization problem. This linear optimization problem will be solved using the Gurobi Optimizer [10]. It can be noted that solving the optimization problem in this way, the partial derivatives of the priority parameters with respect to the model parameters will be assumed to be zero.

5.2 Traffic Lights

Another feature of junctions is traffic lights that control the flow of traffic. The times at which the traffic lights are red or green will be controlled by some cycles describing how long to stay in any state; the cycle $[10, 20]$ can, for instance, describe that the light should first be green for 10 seconds before switching to red for 20 seconds. After these 20 seconds, the cycle is repeated. The number of elements in one cycle has a lower bound of two, but does not have an upper limit. However, it should contain an even number of elements since there are two states a light can be in, and the two states should be visited equally many times for each cycle.

Different choices of times in the cycle give rise to different flow of traffic. We want to find the choices for these times that give the best possible flow. The optimization method in use is gradient based, and we therefore want to be able to evaluate the gradient of the objective function with respect to to the cycle times.

One way of modelling traffic lights is to multiply the flux coming into a junction by some activation function. When the light is red, there should be no flux coming into the junction, and so the activation function should be equal to zero. When the light is green, all flux should be allowed to reach the junction, and the activation function should be equal one. A natural choice of activation function satisfying these properties is a step function, changing from zero to one or from one to zero at the times defined in the cycle. However, a step function is not differentiable, meaning that we will have problems when trying to take the partial derivative of the objective function with respect to the cycle times. Instead, we will make use of a smoothed step function, which will be differentiable. One option for smoothing the step function, is to approximate each jump as a logistic function, so that the combined function will be a linear combination of logistic functions.

The logistic function is defined as

$$\sigma(x) = \frac{1}{1 + e^{-x}} = \frac{e^{x}}{1 + e^{x}}, \tag{53}$$

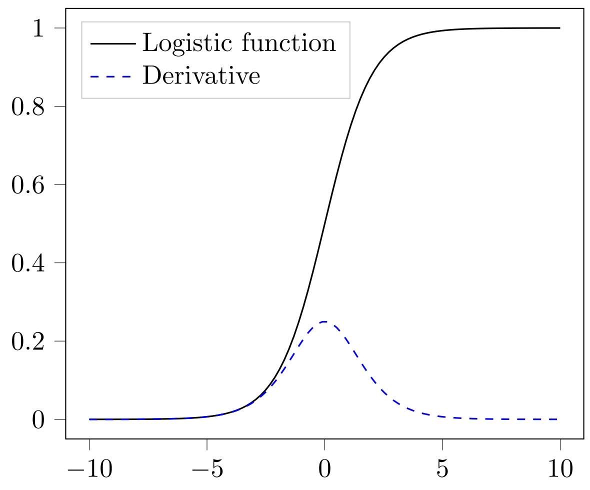


Figure 9: Logistic function and the derivative of the logistic function.

and has a characteristic S-shape, which can be seen in Figure 9. Each change in light will then be approximated by a single logistic function. Making use of such a smooth function means that there are some points in time where only a portion of the flux is allowed to enter the junction. Although this might seem nonphysical, it could model the reaction times of the drivers. When the light changes from red to green, drivers will need some time to react, and hence it seem reasonable to have a period where only a portion of the flux enters the junction. Likewise, when the light changes from red to green, some drivers will stop, whereas other drivers will keep driving. Again it seems reasonable to have a period where only a portion of the flux enters the junction.

We don't want any flux to enter the junction before the light changes, and thus, instead of centering the logistic function at the location of each jump, it will be shifted slightly to the right, so that approximately zero flux enter the junction before the light changes. Figure 9 shows that transitioning from one state to the other takes around 10 seconds, hence we will shift the function by 5 seconds. In addition to shifting the logistic function, the slope can also be controlled. This can be done using a scaling parameter, α. Assuming the light changes at time t, we define the shifted and scaled logistic function as

$$\tilde{\sigma}(x; t, \alpha) = \sigma(\alpha(x - t) - 5). \tag{54}$$

We note that the shift -5 is be kept constant even if the scaling parameter α changes.

The logistic function changes very rapidly only locally around the jump it is centered around. Therefore, the contribution from a logistic function away from its center, will be approximately constant. This can be seen from Figure 53, which shows that the derivative of the logistic function is almost zero away from the jump. Thus, a sequence of transitions can be modelled as a sum of logistic functions. Assuming the starting state is given as $s_{\text{start}} \in \{0, 1\}$, and the times of positive and negative transition times are T_p and T_m respectively, we define the total activation function as

$$\gamma(x) = s_{start} + \sum_{t \in T_p} \tilde{\sigma}(x; t, \alpha_t) - \sum_{t \in T_n} \tilde{\sigma}(x; t, \alpha_t), \tag{55}$$

. Such an activation function will be differentiable, as well as being a reasonable approximation of the fluxes entering the junction in the presence of traffic lights.

Junctions will not typically only consist of one single traffic light, but rather a collection of traffic lights that interact with each other. The main point is that at time t, only some of the traffic lights should be green, whereas all others should be red to avoid collisions between incoming roads. We will consider the simplest case of two traffic lights interacting with each other. In such a case, it might be enough to use only one activation function. If the flux coming into the first traffic light is denoted by f_a^* and the flux coming into the second traffic light is denoted by f_b^* at time t, the actual flux coming into the junction can be calculated as $f_a^* \cdot \gamma(t)$ and $f_b^* \cdot (1 - \gamma(t))$ respectively.

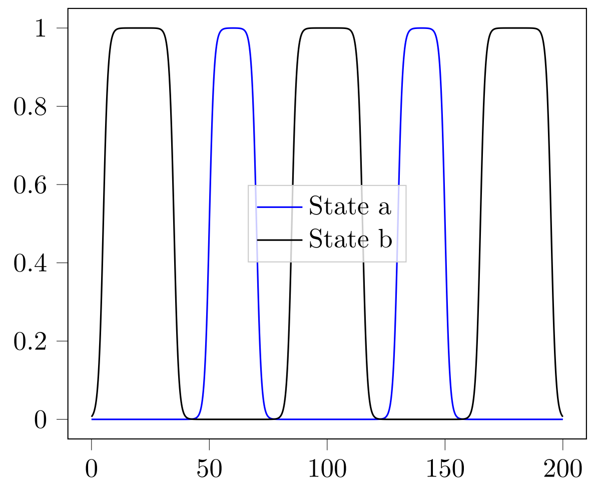


Figure 10: Activation functions for two coupled traffic lights.

One obvious issue with only using one activation function, is that at some point when the light switches state, half of the flux from both roads will be allowed to enter the junction. In real world traffic lights, however, the flux from one road is stopped before the cars from the other road are allowed to drive. To take this into account, we will instead use two activation functions, where there is some gap between a negative jump in the first activation function and a positive jump in the second activation function. A visualization of two suction activation functions can be seen in Figure 10. In the figure, we add a short period of time after each change, where both lights are red. This is a safety feature that can be seen in real world traffic lights, and is implemented to ensure all of the cars from the first road have enough time to exit the junction before allowing the cars from the second road to drive.

As discussed before, each time step is chosen to be the maximum allowed by the CFL condition (21). The problem with blindly choosing the time step as the maximum, is that there is no guarantee that any of the time points will be near each of the jumps in the activation function. If this is the case, the finite-volume scheme will have no way of knowing when exactly the jump occurred, and it will be as if the previous state was stayed in for longer than it should have been. To combat this effect, the next time should not be allowed to be chosen after the nearest jump, without first having been close to this time. We will now look at a simple example to visualize how this effect can be problematic.

Imagine two different activation functions with cycles (in units second) $[10, 20]$and $[11, 19]$, respectively, so that the two cycles are equally long. We will assume that the first state is red for both traffic lights. Suppose the time-stepping is run until time $t = 49$ is reached. Since the density of cars will be slightly different at time $t = 49$ for the two simulations, the next time step chosen will also be slightly different. Suppose that the time step in the first case was chosen to be $\Delta t = 4$ and the time step in the second case was chosen to be $\Delta t = 3$. In the first case, this will be as if the light was red until time $t = 53$ and in the second case it will be as if the light was red until $t = 52$. The actual times of changing are $t = 50$ and $t = 51$. In this scenario, the order of the jumps have been switched, and this effect will pose a problem when trying to optimize the control parameters.

To combat this effect, some method of forcing the simulation to simulate times close to the jumps is necessary. Suppose that the simulation has reached time t and that the next jump is centered around time t^*. Let Δt_{CFL} be the timestep chosen using the CFL condition. Then the actual time step should be picked as $\Delta t = \min\{\Delta t_{CFL}, t^* - t\}$.

5.3 Inflow and Outflow Boundary Conditions

So far, we have not discussed the boundary conditions of roads leading into the system. The roads that are missing boundary conditions are the ones only connected to the road network at their right or only at their left node. The roads only connected at their right node need some inflow conditions, and the roads only connected at their left node need some outflow conditions. If the road leading out of the system is a freeway, it might make sense to allow all cars to leave the system. If instead the final road leads to a road with smaller capacity, there might be a limit to how fast cars can leave the system.

The roads only connecting at their right node need some boundary condition on the left boundary. Using the same supply function as in (47), we get the capacity of the road. Assuming the desired inflow rate f^{in} of cars into the system is known, we can find the actual inflow to the road. If the desired inflow rate is higher than the capacity, a queue will start to form at the left node, while if the capacity is higher, all cars will be able to enter the system. We now define a new demand function that depends on the length l of the queue, similar as in [6]

$$D_i(l) = \begin{cases} f_i^{\max}, & \text{if } l > 0, \\ \min\{f_i^{\text{in}}, f_i^{\max}\}, & \text{if } l = 0. \end{cases} \tag{56}$$

The evolution of the length of the queue $l(t)$ satisfies

$$\frac{dl(t)}{dt} = f_i^{\text{in}} - f_i^{\text{actual}}, \tag{57}$$

where the actual inflow to the road is given by

$$f_i^{\text{actual}} = \min\{D_i(l), S_i(\rho)\}.$$

We note that whenever $l = 0$, we have that $f_i^{\text{actual}} \leq f_i^{\text{in}}$. This means that the queue length will never become negative. When optimizing the flow of traffic in the network, the length of the queues into the system should also be considered.

5.4 Variable Speed Limit

Another natural extension to the model is to allow the speed limit to vary in time. We will model the speed limits as a piecewise constant function that may only change at at finite number of control points, $0 \leq \mu_1 < \cdots < \mu_{N_{speed}} \leq T$. The distance $\mu_{k+1} - \mu_k$ may be different for each j, but since it is not very realistic that the speed limit will vary very rapidly, this distance should not be taken too small.

One important feature of speed limits is that the speed does not change instantly for the whole road. When a sign displaying the speed limit changes, the roads in front have no way of knowing that the speed limit changes until they reach a new speed limit sign. Therefore, the new speed limit will effectively move to the right with the same speed as the first car arriving at the sign. Thus, some time after the speed limit changes, the speed limit at some point on the road will be a function of both time and space. However, for the purposes of the discussions herein, we will assume that the change is instant over the whole road.

By a similar argument as for the traffic lights, it is important to ensure that the times of the jumps in speed are included as control points in the simulation. That is, if t_k is a control point, the time step of the simulation should be controlled as $\Delta t = \min\{\Delta t_{CFL}, t_k - t\}$ if t was the previous time of the simulation.

5.5 Limitations of Model

Even though we have extended the LWR model with several phenomena from real road networks, it is not perfect. We end the discussion bydescribing some of the limitations and looking at some parts of real road networks that we are not able to model accurately.

5.5.1 Duty to give way

As we have seen, the model used is a macroscopic one, meaning that it does not consider individual cars but instead looks at densities of cars. This is in contrast to a microscopic model that models each individual car separately. By looking at each car separately, it is easy to implement right of way. One simply needs to stop any cars from entering a junction if there is a car to the right.

In the macroscopic model however, this is not so trivial. One method to try to approximate the duty to give way is to specify the distribution parameters in Section 5.1.2 in such a way that more of the cars coming from the road with right of way than cars coming from other roads will reach the junction.

5.5.2 More complicated road networks

The model only supports a limited class of road networks and therefore cannot simulate any road network of interest. Some features that are missing from the model are

- Roundabouts: The project only considers cases where roads connect via junctions with or without traffic lights.
- Multi-lane roads: This project only considers single-lane roads. On a multi-lane freeway, the speeds may vary from lane to lane, and there will be some flux between the lanes. Even though this model does not support multi-lane roads, these roads can be approximated by choosing different maximum densities. The model offers the flexibility of specifying the maximum densities for each road, meaning that a two-lane freeway can be approximated by setting the maximum density twice that of a single-lane road.
- More complex junctions: We have only discussed the coupling of two traffic lights is considered. In real road networks, the traffic lights leading in to a junction might be more complicated than what was considered in this project.

6 Optimizing Flow of Traffic

In the previous sections, we have derived the underlying model equations, described a numerical method for obtaining unique solutions, and extended the model to be able to describe more advanced road network. The overall model contains a number of parameters, like the speed limits for each road, which we want to choose in a way that gives a good flow of traffic in the road network we are modelling. This section give possible interpretations of what a good flow of traffic could mean, as well as describing methods that can be used to find an optimal solution.

6.1 Objective Function

With the goal of achieving the “best flow of traffic” through the road network, one needs to first formulate what good flow means. This is a complex issue with no clear right answer.

One possibility is to try to minimize the average time spent in the system. If you want to minimize traffic jams instead, one possibility could be to minimize the longest time spent in the system. A third way is to minimize the total time spent in the system or to maximize the total flow out of the system. If some roads are considered more important than others, it is possible to maximize the flow out of only some of the roads.

Combining these views, it is generally possible to search for a balance between minimal travel time and maximal flux out of the system.

6.1.1 Minimal total travel time

With the goal of minimizing the total travel time, we introduce the objective function

$$J(l,\rho) = \sum_i \beta_i \int_0^T \Big(l_i(t) + \int_0^{b_i} \rho_i(x,t)dx \Big) dt. \tag{58}$$

The parameter β_i serves the role of prioritizing certain roads over others. If $\beta_i = \beta \ \forall i$, all roads will have the same importance. If instead the β_i's are different for some i, the ones with the highest parameter will have a higher reduction in travel time under optimization. We could also have chosen different parameters β_i for the queue leading into the road and for the density on the road itself if the focus was to minimize queues.

In the numerical solution, we only have a finite number of points in time and space. In space, we use a uniform discretization for all roads, but since the length of the roads may differ, the number of gridpoints for each road may vary. Let N_x^i denote the number of cells for road i. The same time points are used for all the roads. Let the number of time points be denoted as N_t, and the time points as $0 = t_0 < t_1 < \cdots < t_{N_t-1} < t_{N_t} = T$, where the time differences $\Delta t^n = t_{n+1} - t_n$ vary with n. The integrals in (58) must be evaluated by some quadrature formula depending on the discrete points of the domain We write the numerical objective function as

$$J(\{p\}) = J\big(l(\{p\}), \rho(\{p\})\big) = \sum_i \beta_i Q\big(l_i + \tilde{Q}(\rho, [0,b_i]), [0,T]\big), \tag{59}$$

where $\{p\}$ is a set of parameters and Q and $\tilde{Q}$ are two quadrature formula that approximates the integrals in (58). These quadrature formula could be the trapezoidal rule, which is the quadrature formula that we will make use of in this project. In (59), both l_i and ρ_i will be 2D vectors, evaluated at the space and time points of the grid. Minimizing this objective function will give a minimal travel time.

6.1.2 Minimal total travel time and maximal outflow

If we want to maximize the outflow from the road network as well as minimizing the total outflow, we extend the objective function as

$$J(l,\rho) = \sum_i \Big\{ \beta_i \int_0^T \Big(l_i(t) + \int_0^{b_i} \rho_i(x,t)dx \Big) dt - \gamma_i \int_0^T f_i^{\text{out}}(t)dt \Big\} \tag{60}$$

Similar to the previous section, the integrals in (60) need to be approximated by a quadrature rule. Assuming the discrete points of the grid are as described in the previous section, the discrete objective function can be written as

$$J(\{p\}) = J\big(l(\{p\}), \rho(\{p\})\big) = \sum_i \big\{ \beta_i Q(l_i + \tilde{Q}(\rho, [0,b_i]), [0,T]) - \gamma_i \hat{Q}(f_i^{\text{out}}(\rho_i), [0,T]) \big\}. \tag{61}$$

The parameters β_i and γ_i can be chosen to get an appropriate balance between minimizing the travel time and maximizing the outflow. In this project, we will set all of these parameters equal to one.

6.2 Optimization Approach

In the previous section, we introduced objective functions that try to quantify the flow of traffic. The optimal solution is a set of parameters $\{p\}$ that minimize one of these objective functions. The subsequent sections will describe some methods one can use to estimate this optimal solution.

6.2.1 Gradient descent

Probably the simplest gradient-based iterative scheme that can be used to find the maximum of an objective function is the gradient descent algorithm, where steps are taken in the direction opposite of the gradient. The major challenge of this method is to determine a good step size since the scheme might not converge if the step size is chosen too small or too large. It is possible to find the optimal step size, but this includes solving a new optimization problem at each step. One way to combat this is to use a backtracking line search to choose a good step size. The idea is to first start with a maximum step-size value, and then iteratively choose a smaller one until some condition is fulfilled. This condition could, e.g., be

$$f(x_k + \alpha p) \leq f(x_k) + c\alpha p^T \nabla f(x_k), \tag{62}$$

where p is the search direction and α is the step size.

This method can be extended to allow bounds for each parameter. If component x_i is bounded as $l_i \leq x_i \leq u_i$, then at each iteration the new point will need to be projected onto the feasible set. We then get the new iteration as

$$x_{k+1} = P[x_k - \alpha_k \nabla f(x_k)],$$

with the projection operator defined as

$$[P(x)]_i = \text{median}\{x_i, l_i, u_i\}.$$

The condition on the step length is also updated in a similar manner, giving the condition

$$f(P[x_k + \alpha p]) \leq f(x_k) + c\alpha p^T \nabla f(x_k). \tag{63}$$

The starting guess for the step length is chosen so that the parameter that is updated most will be updated by 10 (km/h or s). To be able to get a correct starting guess for the step length, we ignore parameters that are at the boundary of the feasible domain, where taking a step in the direction of the gradient would send them out of the domain. To this end, let p be any parameter of the system, and let $\dot{p}$ be the derivative of the objective function with respect to p. If p is already at the boundary of the feasible domain, and $p - \alpha\dot{p}$ is outside the domain $\forall \alpha > 0$, we set $\dot{p} = 0$. We can now scale the gradient to get an updating step of 10 (km/h or s) as, assuming at least one non-zero component,

$$\alpha = \frac{10}{\max_i\{\nabla f(x_k)_i\}}.$$

6.2.2 Sequential quadratic programming

Sequential quadratic programming (SQP) is a class of method for solving a constrained nonlinear optimization problem on the form

$$\min_x \; f(x) \text{ s.t. } h(x) = 0 \text{ and } g(x) \geq 0, \tag{64}$$

where f, h, and g may be nonlinear functions. A detailed description of the method can be found in [19]. The strength of these methods is that they are able to handle nonlinearity in both the objective function as well as in the constraints. One drawback is that knowledge of the derivatives is required. In our case, the derivatives of the (linear) constraints can be easily calculated, and the derivatives of the objective function with respect to the control parameters are evaluated using automatic differentiation. The objective function will in general be nonlinear, but the constraints are simple linear functions. Before defining the SQP method, we need the Karush–Kuhn–Tucker (KKT) conditions and the Lagrangian function.

The Lagrangian function combines the objective function and the constraints into one function by introducing the Lagrangian multipliers λ and μ. The Lagrangian related to the problem (64) is defined as

$$\mathcal{L}(x, \lambda, \mu) = f(x) + \lambda^T h(x) + \mu^T g(x). \tag{65}$$

The first-order KKT conditions give criteria that any local solution of (64) need to satisfy. Supposing that x^* is a local solution and that some conditions are met. Let λ^* and μ^* be the Lagrangian multiplier vectors. Then the following conditions are met

$$\begin{aligned} \nabla_x \mathcal{L}(x^*, \lambda^*, \mu^*) &= 0, \\ h(x^*) &= 0, \\ g(x^*) &\geq 0, \\ \lambda_i^* &\geq 0, \\ \lambda_i^* h_i(x^*) &= 0, \\ \mu_i^* g_i(x^*) &= 0. \end{aligned} \tag{66}$$

Instead of trying to find critical points of the original function, the SQP algorithm tries to find critical points of the Lagrangian making use of the fact that any critical point of the Lagrangian will be a critical point of the objective function. The SQP method can be seen as an iterative method that uses Newton's method to find the critical points of the Lagrangian. Another viewpoint is that at each iteration, the objective function will be approximated by a quadratic sub-problem. This quadratic problem will then be solved to find the next guess. This gives rise to the name SQP, since a quadratic problem is solved in each iteration.

We can write the Lagrangian problem as

$$\begin{aligned} \min_p f_k + \nabla f_k^T p &+ \frac{1}{2} p^T \nabla_{xx}^2 \mathcal{L}_k p \\ \text{s.t. } \nabla h_i(x_k)^T p + h_i(x_k) &= 0 \\ \nabla g_i(x^k)^T p + c_i(x_k) &= 0 \\ \nabla g_i(x^k)^T p + c_i(x_k) &\geq 0, \end{aligned} \tag{67}$$

where p is the search direction, $p = x_{k+1} - x_k$. At each iteration this quadratic problem will be solved to obtain the new iterate.

We introduce the set of active inequality constraints $\mathcal{I}$ at the optimal solution as

$$\mathcal{I} = \{i : g_i(x^*) = 0\}.$$

Using the last of the KKT conditions we get that $\mu_i = 0 \ \forall i \notin \mathcal{I}$. This gives rise to the *active set method*. A detailed description of this method can be found in [19]. In each iteration of the algorithm, an estimate of the active set at the solution $\mathcal{I}$ is stored. This estimate is commonly know as a *working set* and denoted $\mathcal{W}$. When the active set is known, the problem reduces to an equality constrained problem, which can be solved more easily. At each iteration the estimate of the active set $\mathcal{W}$ is updated, and the resulting equality constrained quadratic problem is solved.

6.2.3 Variable continuous speed limits

We will now close the gap between the general optimization problem considered in the two previous sections and the specific optimization we are considering. Let the objective function be defined in one of the ways described in the previous section. Let the road network consist of n roads, and m^i discrete points in time for each road i on which the speed limits may change. In this case, we have $\sum_{i=1}^n m^i$ variable parameters that may impact the overall flux. These parameters are bounded, meaning there is an upper and a lower limit to the speed limit on each road. We want to choose the parameters in a way that minimizes the objective function, and to this end, we formulate it as a constrained minimization problem.

Let

$$L(\{v_{i,j^i}^{\max}\}_{i,j^i}), \; i = 1, \ldots, n, j^i = 1, \ldots, m^i,$$

be one of the objective functions defined in the previous section. We then formulate the minimization problem as

$$\begin{cases} \min \; L(\{v_{i,j^i}^{\max}\}_{i,j^i}), & i = 1, \ldots, n, \;\; j^i = 1, \ldots, m^i, \\ \text{s.t. } \; v_i^{\min} \leq v_{i,j^i}^{\max} \leq v_i^{\max}, & \end{cases} \tag{68}$$

where $v_i^{\min}$ denotes the minimum speed limit on road i and $v_i^{\max}$ the maximum speed limit on road i.

This minimization problem can be solved using either sequential quadratic programming (SQP), or as we will in do in this project, using projected gradient descent.

6.2.4 Variable speed limits and traffic lights

We now extend the model to also include traffic lights. The parameters we optimize with respect to are the time intervals the two states stay in for each traffic light. Each cycle has the parameters $\{\tau_{2i}, \tau_{2i+1}\}$ for $i \in \mathcal{I}$. If there are k traffic lights, we have the parameters $\{\tau^j_{2i}, \tau^j_{2i+1}\}$ for $i \in \mathcal{I}^j$, $j = 1, \dots, k$. These parameters will also be bounded in a similar way as the variable speed limits. The overall minimization problem will then be similar to the one for the variable speed limits, but with some extra parameters. We again let

$$L(\{v^{\max}_{i,j^i}\}_{i,j^i}, \{\tau^{j_t}_{2i_t}, \tau^{j_t}_{2i_t+1}\}_{i_t,j_t}),\ i = 1, \dots, n,\ j^i = 1, \dots, m^i, i_t \in \mathcal{I}^{j_t}, j_t = 1, \dots, k,$$

be one of the objective functions defined in Section 6.1 and define the minimization problem as

$$\begin{cases} \min L(\{v^{\max}_{i,j^i}\}_{i,j^i}, \{\tau^{j_t}_{2i_t}, \tau^{j_t}_{2i_t+1}\}_{i_t,j_t}),\ i = 1, \dots, n,\ j^i = 1, \dots, m^i, i_t \in \mathcal{I}^{j_t}, j_t = 1, \dots, k, \\ \text{s.t. } v_i^{\min} \le v^{\max}_{i,j^i} \le v_i^{\max} \text{ and } \tau^{\min}_{j_t} \le \tau^{j_t}_{2i_t}, \tau^{j_t}_{2i_t+1} \le \tau^{\max}_{j_t}, \end{cases} \tag{69}$$

where $v_i^{\min}$ and $v_i^{\max}$ again denote the minimum and maximum speed limit on road i, and $\tau^{\min}_{j_t}, \tau^{\max}_{j_t}$ denote the minimum and maximum times the state of traffic light j_t can be stayed in.

This minimization problem is qualitatively the same as (68) and can be solved either using SQP or projected gradient descent, as we will do herein.

7 Numerical Results

Combining the previous sections gives us the tools necessary to numerically find a set of parameters $\{p\}$ that give rise to best possible flow of traffic through a road network. This section considers three different road networks of increasing complexity. For each road network, the optimal set of parameters is estimated and the behaviour of the optimization algorithm is investigated. The first network (see Figure 11) is very simple and consists of only two lanes with a traffic light between them. The second network (Figure 23) is more complex and consists of four lanes connected in one junction, with two roads lead into and two roads leading out from the junction. The flux across the junction is controlled by two traffic lights that are coupled together. The third and final network (Figure 29) is more complicated and consists of six roads connected by two junctions.

7.1 Network A

The first network we consider is a simple network consisting of only two roads connected with a traffic light between them. This network is useful as a sanity check to ensure the optimization approach behaves as expected. If the two roads are equal, the best flow is achieved by setting the traffic light to be green for as long as possible. If the optimization method comes up with a different answer, it might indicate that something is wrong. In this section, we will order the roads as shown in Figure 11.

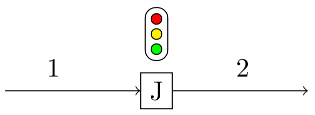


Figure 11: Graphical representation of road network A. The network consists of two roads connected in a junction with a traffic light between them.

7.1.1 Optimizing only control parameters of the traffic light

For the first example, we only optimize the parameters related to the traffic light and fix all other parameters. The two roads were both of length 1km with the same maximum capacity (set to 1 in this case). To isolate the optimization with respect to the traffic lights, speed limits for the two roads were fixed at 50km/h. We chose the grid spacing $\Delta x = 50$m while Δt^n was decided by a CFL condition at each step. The initial density of the two roads were constant at 0.8 and 0.1, respectively. The inflow to the first road was chosen to be constant with a influx equal to the flux from a road with density 0.6 and speed limit 50km/t.

The network was simulated for a total time of 2000 seconds with a traffic light having a cycle consisting of two times t_1 and t_2, with the starting state set to be green. Since traffic lights will change the light after some time has passed, the cycle times were bounded as $t_i \in [10, 120]$ for $i \in \{1, 2\}$. Moreover, we expect the traffic light to be green for as long as possible, and the expected behaviour is thus that t_1 approaches 120s and that t_2 approaches 10s. The optimization approach used was a projected gradient descent with line search.

The gradient descent is sensitive to the starting points and we therefore tested multiple starting points (in unit seconds): $(20, 20)$, $(50, 50)$, $(80, 80)$, $(30, 80)$, and $(80, 30)$. For all these cases, the optimal value was found at the point $(120, 10)$, which is the expected optimal value. In other words, for this specific problem the optimal solution was found independent of the starting point. Figure 12 shows the decrease of the objective value as a function of the number of iterations. We see that even though the stopping point was independent of the starting point, the number of iterations is greatly impacted by the starting point. Since we are only optimizing two parameters in this scenario, we can also visualize the path that was taken to obtain the optimal solution for the different starting points. Figure 13 shows the paths taken in each run of the optimization routine.

We can also try to visualize how the flow was improved by choosing different control parameters for the traffic light. Figure 14 shows the densities of the two roads at the end of the simulation for two different choices of cycle times for the traffic light. Even though these images only show the densities at the end point of the simulation, and the objective function considers the densities for all times, they give an indication of how choosing different control parameters can impact the flow of traffic. With the cycle times $(50, 50)$ as shown in Figure 14a, there is very heavy traffic on the first road. In Figure 14b, however, the traffic on the first road is much lighter. Thus we see that for this road network, a better choice of control parameters will reduce the buildup of traffic.

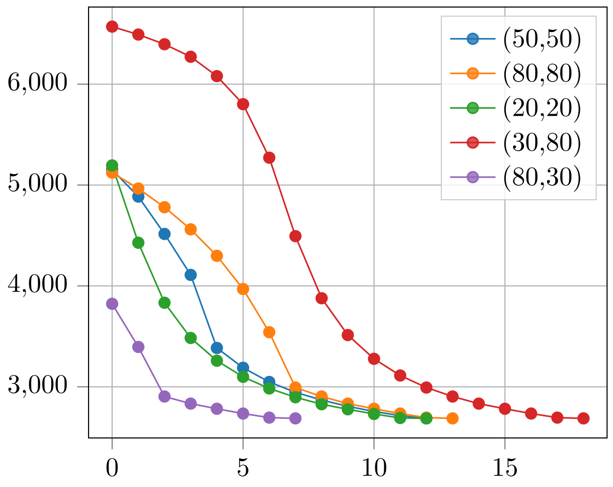


Figure 12: Objective value as a function of the number of iterations for different starting points.

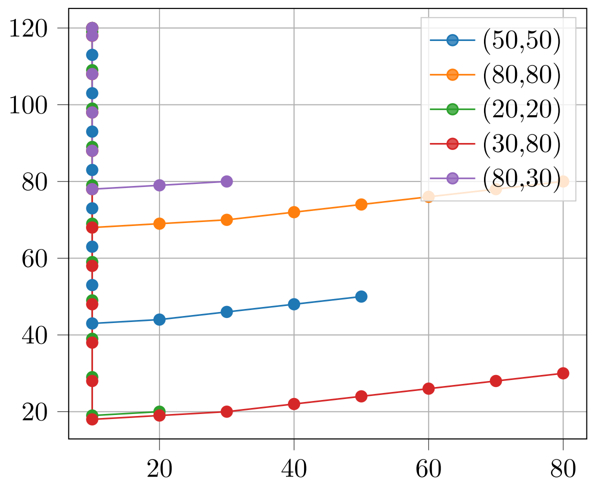
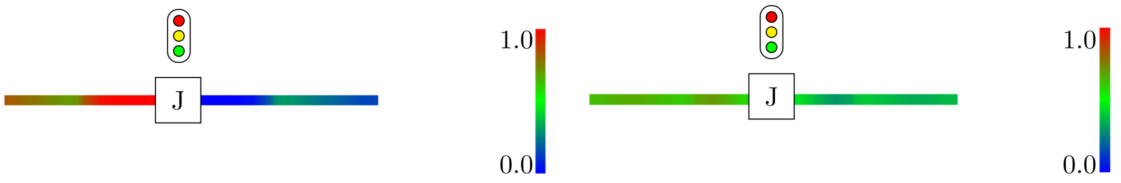


Figure 13: The path taken to obtain optimal solution for different starting points.



(a) Cycle times (50, 50).

(b) Cycle times (120, 10).

Figure 14: The figure shows the density at the end of the simulation for two different choices of control parameters for the traffic light connecting the two roads.

7.1.2 Optimizing constant speed limits on roads

Another case where this simple road network is interesting is the case where the first road has a higher maximal capacity than the second road. This case could represent a multi-lane freeway leading into a smaller road. In this case, setting the speed limit to be as high as possible might not be the best choice as this might lead to a buildup of traffic leading in to the junction. Instead, it might be better to choose a speed limit for the first road so that not too many cars reach the junction in a too short time period to avoid the buildup of traffic into the junction. As before, the lengths of the two roads were both set to equal 1km. The first road had a maximum density of 2, and the second road had a maximum density of 1. The initial densities for the two roads were both set to 80% of the maximum capacity, and the inflow to the first road was chosen to mimic a road with density 0.2 and speed limit 80km/t leading into the system; that is, we put a constant flux into the first road corresponding to the flux coming from a road with density 0.2 and speed limit 80km/t. In this scenario, we do not allow the speed limits on the roads to vary in time, and we assume that they are bounded in the interval [30km/t, 80km/t]. Between the two roads there is a traffic light with a fixed cycle, where green and red lights are both turned on for 100 seconds at a time.

As in Section 7.1.1, we will consider multiple starting points for the optimization algorithm, and since we are only optimizing two parameters, we will visualize the paths taken for each starting point. Figure 15 shows the number of iterations needed before a solution was reached for the different starting points, while Figure 16 shows the paths taken to reach the optimal solution in the different cases. The optimal solution that was reached for all starting points was to choose a 42km/t speed limit for the first road, and the maximum possible 80km/t speed limit for the second road. Thus, in this case, choosing a lower speed limit helped reduce the road congestion and improved the overall flow of traffic.

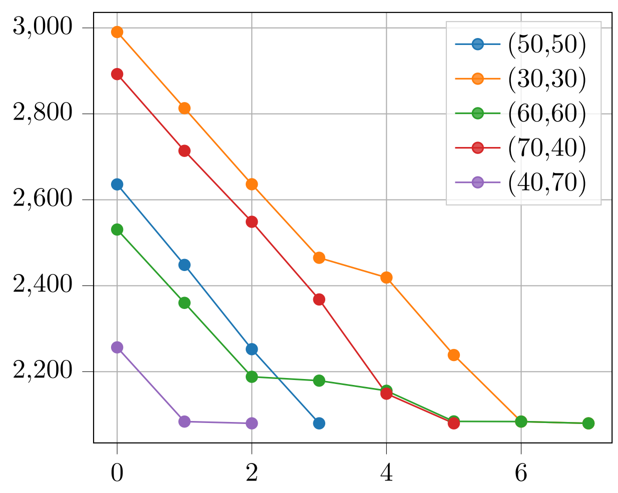


Figure 15: Objective value as a function of the number of iterations for different starting points.

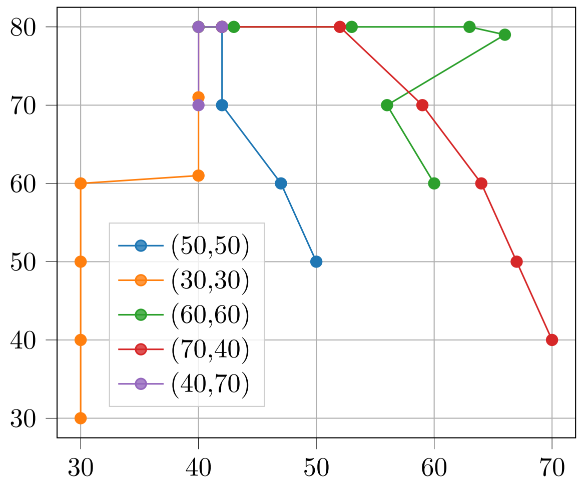


Figure 16: The path taken to obtain optimal solution for different starting points.

7.1.3 Optimizing variable speed limits on the roads

We will now extend the previous example by allowing the speed limits to vary also in time. As discussed in Section 5.4, we assume that the speed limit is a piecewise constant function of time, and that the speed limit changes instantly over the whole road. Since the road is congested at the beginning, it might be that a better overall flow of traffic can be achieved by setting a higher speed limit after the traffic has cleared. We will try to find the optimal choice of speed limits while allowing the speed limit to vary at an increasing number of points.

Figure 17 shows the best choice of speed limits for the incoming road for some different choices of control points. When the speed limit is not allowed to vary in time, we get the optimal solution from the previous section, with 42km/t as the best speed limit. By adding one control point at $t = 500$s, the optimal choice changes, and the best possibility now is to have the speed limit 42km/t for the first interval, before increasing the speed limit to 80km/t. This change can come from the fact that after the first interval, the congested road has cleared up, meaning that the speed limit can be increased without risking a new buildup of traffic. By adding more control points, the optimal choice changes slightly, but in general, a speed limit close to 42km/t for the first 500 seconds, followed by a speed limit close to 80km/t will be preferred in all cases.

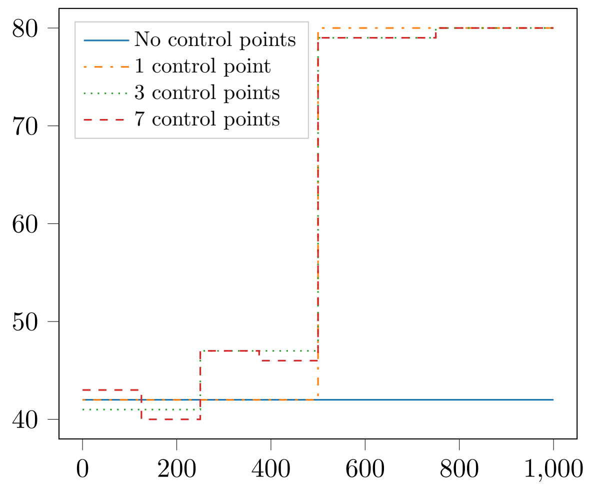


Figure 17: Optimal choice of speed limits for different number of control points.

Figure 18 shows how the objective at the optimal point improves when the speed limit can also vary in time. In this specific case, the main contributor to a high objective value is the congested road in the beginning of the simulation. Since the speed limit is almost equal in the beginning of the simulation for all of the cases, the heavy traffic is being cleared up almost equally quickly in all cases. This explains why the objective value at optimum is similar for the different number of control points. However, we still see that increasing the number of control points helps reduce the objective value, resulting in a better flow of traffic.

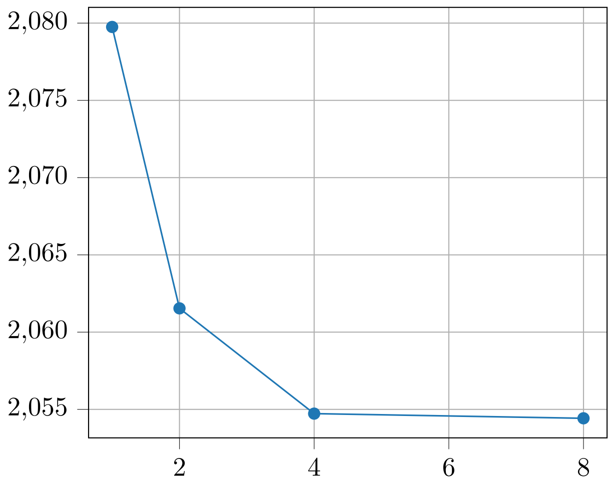


Figure 18: Objective value at the optimal point as a function of the number of levels for the speed limit.

Figures 19 and 20 show the decrease of the objective value for different starting points. Figure 19 compares results for different numbers of control points, while Figure 20 only considers the case of three control points. The comparison between the number of points shows that the initial decrease is more or less equal per iteration for each number of control points. However, choosing more control points makes it possible to reduce the optimal value slightly more. Figure 20 shows that in the case with three control points, a similar optimal value was reached regardless of the starting point.

Table 3: Starting points for the optimization method for different number of control points. The number in the left column denotes the number of control points, and the letter specifies which starting point it is. **1***, **3*** and **7*** are the optimal solutions for one control point, three control points and seven control points respectively.

	Objective value	**Speed Limits**	
		Road 1	**Road 2**
1A	2125.6	[50, 70]	80
1B	2122.7	[50, 80]	80
3A	2122.7	[50, 50, 80, 80]	80
3B	2122.7	[50, 50, 80, 80]	80
7A	2122.6	[50, 50, 50, 50, 80, 80, 80, 80]	80
1*	2061.5	[42, 80]	80
3*	2054.7	[41, 47, 80, 80]	80
7*	2054.4	[43, 40, 47, 46, 79, 79]	80

Table 4: Starting points for the optimization method with three control points. If only speed limit is specified for the second road, it means that this speed limit is chosen for all times.

	Objective value	**Speed Limits**	
		Road 1	**Road 2**
A	2122.7	[50, 50, 80, 80]	80
B	2122.7	[50, 50, 80, 80]	[80, 80, 70, 70]
C	2360.6	[70, 70, 70, 70]	70
D	2359.0	[40, 40, 60, 60]	[60, 60, 80, 80]
E	2080.0	[45, 45, 80, 80]	80
F	2122.7	[50, 50, 70, 70]	80
G	2635.8	[50, 50, 50, 50]	50

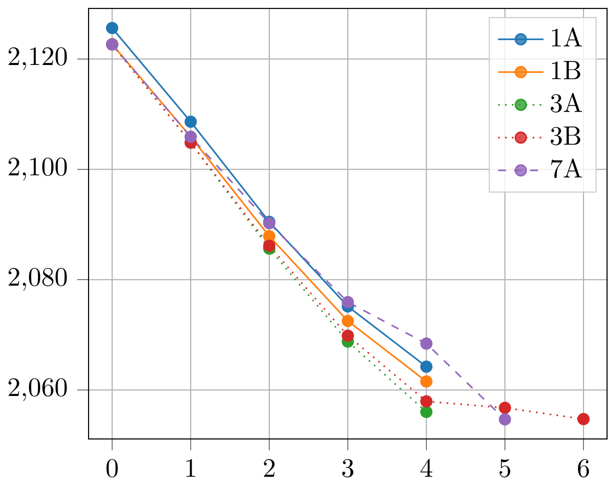


Figure 19: Objective value as a function of number of iterations for some different starting points and number of control points. The starting points are given in Table 3

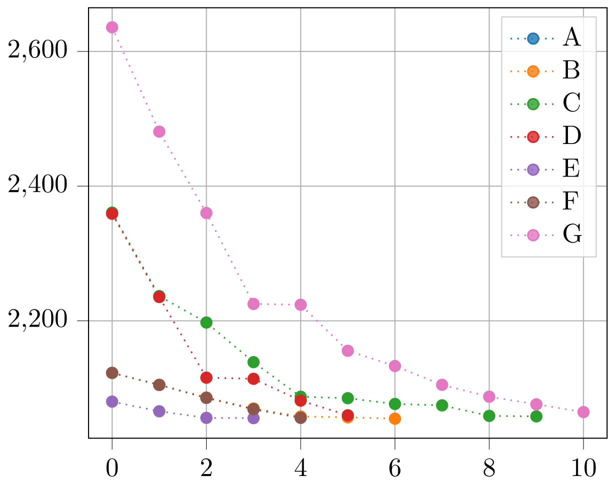


Figure 20: Objective value as a function of number of iterations for some different starting points with three control points. The starting points are given in Table 4

7.1.4 Optimizing controls for both traffic lights and speed limits

Finally, we will combine the optimization of the traffic light and the speed limits. In this case, we consider one control point at $t = 500$s for both the roads. The specifics of the network and the simulation are equal to those of the previous section. Figure 21 shows the reduction of the objective value as a function of the number of iterations for some different starting points. We see that although the optimal value was not reached in all cases, a similar optimal value was reached. We can also note that allowing the traffic light to be optimized in conjunction with the speed limits results in a much lower optimal value than when the traffic light is fixed. This seem reasonable, as setting the traffic light to be green for a longer period of time will allow more flux to cross the junction.

Table 5: Starting point for the speed limits and cycle times for the traffic light. The optimal solution and the objective values at each starting point is also included.

	Objective value	**Speed Limits**		**Traffic Light**	
		Road 1	**Road 2**	**Green light**	**Red light**
A	1456.6	[50, 70]	80	110	20
B	1807.8	[60, 75]	80	115	30
C	1427.0	[50, 80]	80	120	20
Optimal	1253.8	[43, 80]	80	120	10

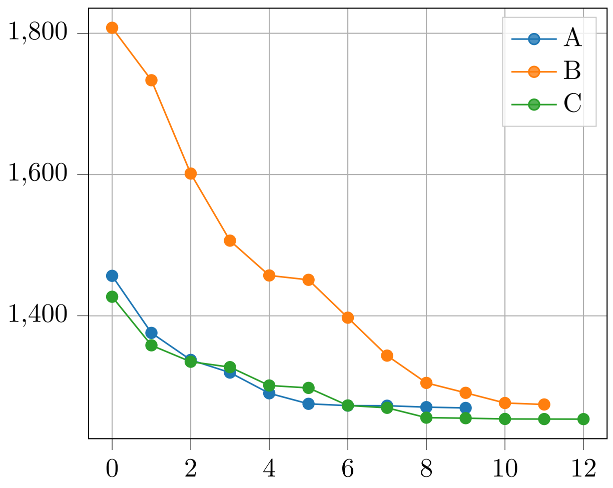
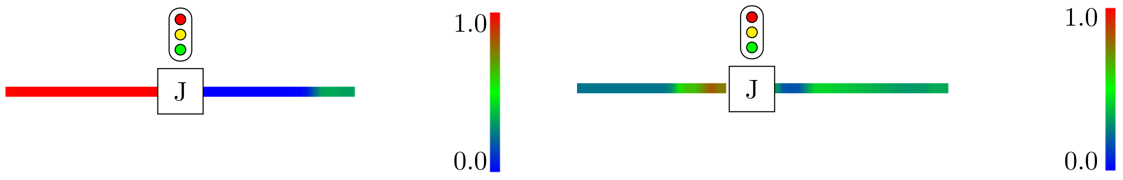


Figure 21: Objective function as a function of number of iterations for some different starting points. The starting points are given in Table 5

Figure 22 shows how the flow in the network at the endpoint of the simulation was improved by choosing a better set of control points. Figure 22a shows that there is heavy traffic on the entirety of the first road. On the other hand, Figure 22b shows that a better choice of control parameters resulted in much lighter traffic on the roads at then end of the simulation.



(a) Bad choice of parameters.

(b) Optimal choice of parameters.

Figure 22: The figure shows the density at the end of the simulation for the parameters given by starting point A and the optimal solution in Table 5.

7.2 Network B

The second network is slightly more complex than the network we considered in the previous section. It consists of four roads connected in one junction. As discussed in Section 5.2, the junction has two traffic lights that are coupled together, meaning that only one of them can be active at a time. We will start by only optimizing the control parameters of the coupled traffic lights, before combining the optimization of the speed limits with the optimization of the coupled traffic lights. A graphical representation of the network is shown in Figure 23. The figure also shows the ordering of the roads that will be used in this section.

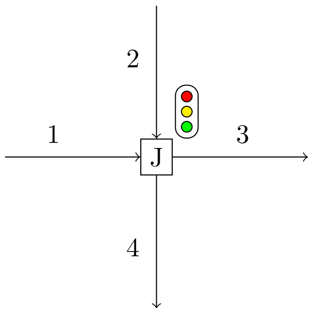


Figure 23: Graphical representation of road network B. The network consists of four roads connected in a junction with two coupled traffic lights between them.

7.2.1 Optimizing only control parameters of the coupled traffic lights

As in Network A, a step length of $\Delta x = 50$m is used, and the time step length is determined using the CFL condition. All of the roads in the network are 1km long. One of the incoming roads has a maximum density of 2, and all the other roads have a maximum density of 1. In the first test, we only look at the optimization of the coupled traffic lights, and therefore we fix the speed limit for road 1 to 80km/h, and 50km/h for the other three roads. The initial densities of the two incoming roads are equal and set to 0.8 and the initial densities of the two outgoing roads are both set to 0.1. The network is simulated for a total of 1000 seconds. The distribution parameters are as follows

$$\alpha = \begin{bmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{bmatrix}$$

meaning that most of the flux from the first incoming road will go to the first outgoing road, and most of the flux from the second incoming road will go to the second outgoing road.

The starting state of the coupled light is green for the first incoming road and red for the second incoming road. The cycle defining the change in states for the coupled traffic light consists of two times t_1 and t_2. As before, the starting point might impact the optimal solution found, so multiple different starting points will be considered. For the first test, we are only optimizing the control parameters of the coupled traffic lights, and we will fix the speed limit of the first incoming road at 80km/t and all other speed limits at 50km/t. This case could represent a two-lane freeway and a smaller road leading into a junction with two smaller roads going out. We assume the influx to the roads to be constant in time.

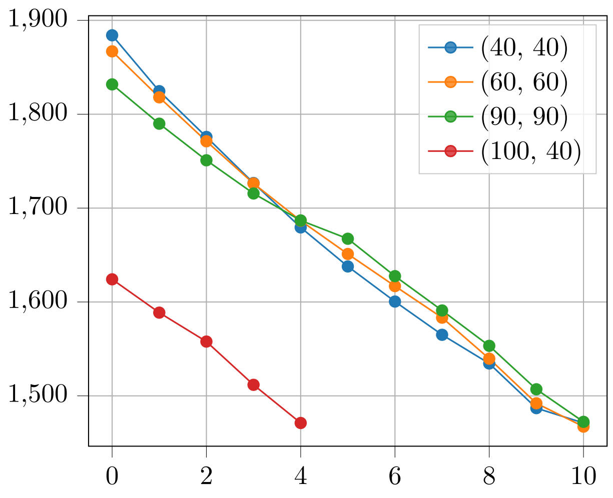


Figure 24: Reduction in objective value as a function for the number of iterations for some different starting points.

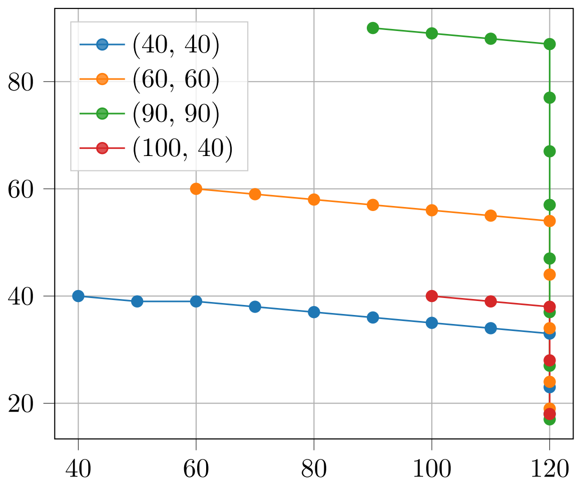
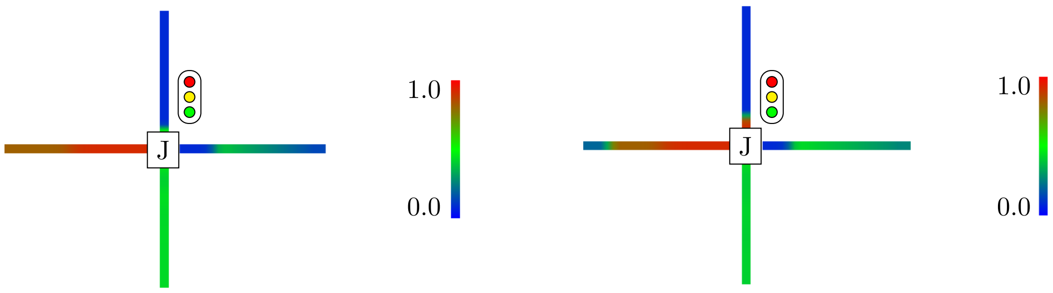


Figure 25: Path to optimal solution for different staring points.



(a) Cycle times (60, 60).

(b) Cycle times (120, 19).

Figure 26: The figure shows the density at the end of the simulation for two different choices of control parameters for the traffic light connecting the two roads.

7.2.2 Optimizing controls for both traffic lights and speed limits

Similar to the first network we considered, we will now extend the optimization to also consider the speed limits of the roads. We assume that the roads and boundary conditions are the same as in the previous case, and that the only difference is that the speed limits are allowed to vary in time. We will still assume the speed limits of the two outgoing roads to be fixed at 50 km/t, but we allow the speed limit of the two incoming roads to take two values, with a control point a $t = 500$s.

Figure 27 shows the reduction in the objective value as a function of the number of iterations. The specific starting points of the graph can be seen in Table 6. As in the previous section, we will visualize the densities of the roads at the end of the simulation. Figure 26 shows the end densities for two different choices of control parameters. From Table 6, it can be seen that the objective value can be reduced drastically by choosing a better set of control parameters. This is also supported by Figure 26, which shows that at the end of the simulation, there is a traffic jam spanning the entire length of road 1. On the other hand, when choosing a better set of control parameters, there is little traffic on the road network at the end of the simulation.

Table 6: Starting point for the speed limits and cycle times for the traffic light. The optimal solution and the objective values at each starting point is also included. Note that the time under green light is the time the there is green light for the traffic light that road 1 leads into.

	Objective value	**Speed Limits**				**Traffic Light**	
		Road 1	**Road 2**	**Road 3**	**Road 4**	**Green light**	**Red light**
A	1867.0	[80, 80]	[50, 50]	50	50	60	60
B	2094.8	[60, 60]	[60, 60]	50	50	90	90
C	1826.2	[70, 70]	[70, 70]	50	50	40	40
Optimal	943.0	[80, 80]	[80, 80]	50	50	120	10

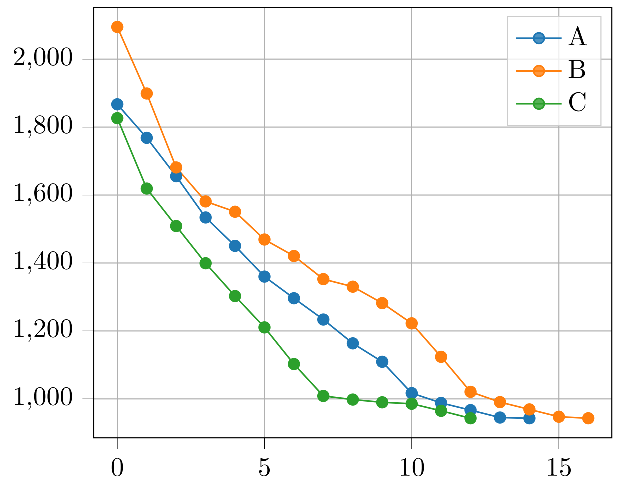
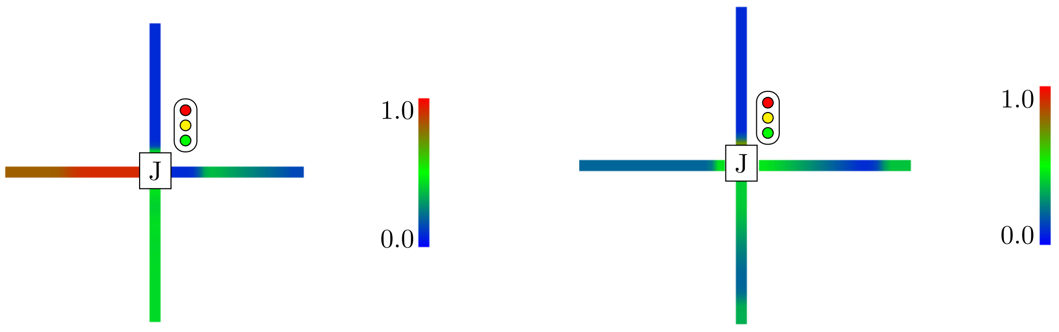


Figure 27: Reduction in objective value as a function for the number of iterations for some different starting points. The starting points are given in Table 6



(a) Unoptimal choice of control parameters.

(b) Optimal set of parameters.

Figure 28: The figure shows the density at the end of the simulation for the control parameters of starting point A and the optimal solution in Table 6.

7.3 Network C

The final road network we consider consists of multiple roads, coupled traffic lights and junctions. Two incoming roads leads to the first junction, which only has one outgoing road. This outgoing road leads to the second junction, which has one other incoming road. Finally, the network has two outgoing roads, leading out from the second junction. A graphical presentation of this network can be seen in Figure 29.

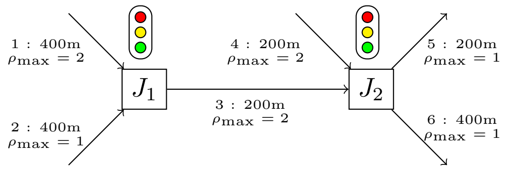


Figure 29: Graphical representation of road network C.

7.3.1 Optimizing flow

When optimizing network A and B, we started by optimizing only the speed limits or only the control parameters related to traffic lights. In this section, we will skip these steps, and instead directly optimize all of the control parameters at the same time.

As before we set the step length $\Delta x = 50$m and find the time step using a CFL condition. The lengths and maximum densities of the roads in the network can be seen in Figure 29. This Figure is a simplified visualization of the network, and the lengths of the roads are not to scale. The two incoming roads to the first junction both have lengths of 400 meters. The road connecting the two junctions has a length of 200 meters.The second incoming road to the second junction has a length of 200 meters. Finally, the two outgoing roads have lengths of 200 and 400 meters. Roads 1, 3 and 4 have a maximum capacity of 2, and roads 2, 5 and 6 have a maximum capacity of 1. The initial densities are constant for all roads, and equal to 10%, 80%, 90%, 90%, 10% and 10% of the respective maximum densities of the six roads. For both of the outgoing roads we assume that all of the flux is allowed to exit. Road 1 has a constant influx of corresponding to a road of maximum density 2 and speed limit 80km/t with density 0.06. Road 2 has a constant influx of corresponding to a road of maximum density 1 and speed limit 80km/t with density 0.15. Road 4 has a constant influx of corresponding to a road of maximum density 2 and speed limit 80km/t with density 0.05. Instead of the influxes being constant over the entire simulation, all three of them will be doubled in the interval $[300s, 600s]$. and multiplied by 0.8 in the interval $[800s, 1000s]$. The simulation will be run for a total of 1000 seconds.

Figure 30 shows how the objective value is reduced starting from the points specified in Table 7. We can see from the figure that starting from the same point, but allowing the speed limits to vary in time, results in a better overall flux, and in this case also a faster convergence. Figure 31 shows the densities on the six roads at the end of the simulation. In Figure 31a, which is simulated using the control parameters of starting point A in Table 7, we see that there is heavy traffic on multiple of the roads, especially on the road connecting the two junctions. In Figure 31b, which is for the optimal choice of control parameters, we see that there is less traffic on the roads. This is particularly apparent for the road connecting the junctions.

Table 7 also specifies the optimal solution that was reached from the two starting points using one and no control points. Interestingly, we see that when using no control points, the constant speed limit will be between the two speed limits when they are allowed to change for all roads. We can also note that the cycle times of the traffic lights differ when using one or no control points.

Table 7: Starting point for the speed limits and cycle times for the two coupled traffic lights. The optimal solution and the objective values at each starting point is also included. A and B correspond to using one control point at $t = 500$s for the speed limits and using no control points respectively. The subcategories **A** and **B** under **Traffic Lights** denote how long each traffic lights stay in state A and B respectively. State A for the first coupled traffic lights is to have green light for road 1 and red light for road 2. State A for the last coupled traffic lights is to have green light for road 3 and red light for road 4.

	Objective value	**Speed Limits of Roads**						**Traffic Lights**			
		1	**2**	**3**	**4**	**5**	**6**	**A**	**B**	**A**	**B**
A	14481.6	[60, 60]	[60, 60]	[60, 60]	[60,60]	50	50	60	60	60	60
B	14481.9	60	60	60	60	50	50	60	60	60	60
Optimal B	12961.9	80	62	59	80	50	50	74	56	84	85

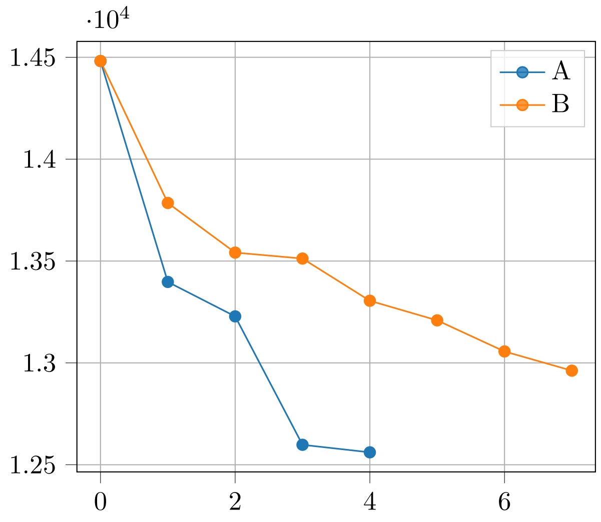
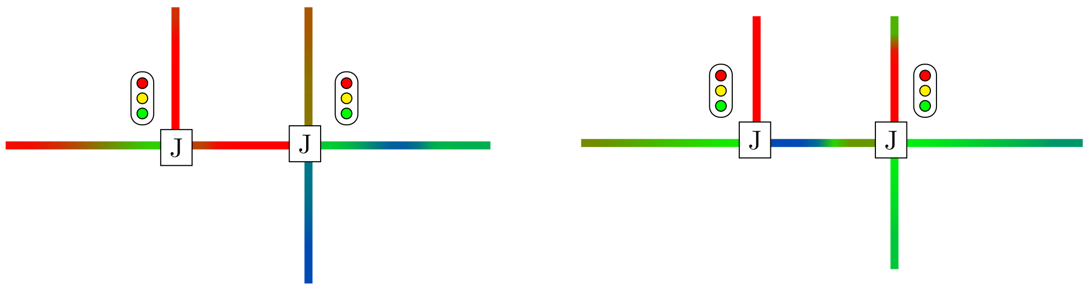


Figure 30: Reduction in objective value as a function of the number of iterations for different starting points. The path starting from A has one control point for the speed limits at $t = 500$s and the path starting from B only allows the speed limit to be constant in time.



(a) An unoptimal choice of control parameters. (b) Optimal choice of parameters

Figure 31: The figure shows the density at the end of the simulation for the control parameters of starting point A and the optimal solution specified in Table 7.

8 Conclusion

In Section 2, we have shown how to use conservation laws to model the flow of traffic in a traffic network while we in Section 3 introduced a finite-volume discretization as a method of solving these models numerically. We saw in Section 4 how automatic differentiation can be used to construct a gradient-based optimization method with the goal of optimizing the flow in a road network.

Using these tools, we optimized the flow in three different networks. We saw that in some cases, the best flow was achieved by reducing the speed limit. The optimization method in use was a projected gradient descent, which did not always find the optimal solution for all starting points. Therefore it would be interesting to compare the results with those from a different gradient-based optimization method, as well as with results from non-gradient based optimization methods. It would also be interesting to utilize a more accurate model, like the one discussed in Section 2.2.1, in place of the LWR model.

Bibliography

[1] A. Aw and M. Rascle. 'Resurrection of "Second Order" Models of Traffic Flow'. In: *SIAM Journal on Applied Mathematics* 60.3 (2000), pp. 916–938. DOI: 10.1137/S0036139997332099.

[2] M. Burger, S. Göttlich and T. Jung. 'Derivation of a first order traffic flow model of Lighthill-Whitham-Richards type'. en. In: *IFAC-PapersOnLine* 51.9 (2018), pp. 49–54. ISSN: 24058963. DOI: 10.1016/j.ifacol.2018.07.009.

[3] R. Colombo. 'A 2 × 2 hyperbolic traffic flow model'. In: *Mathematical and Computer Modelling - MATH COMPUT MODELLING* 35 (Mar. 2002), pp. 683–688. DOI: 10.1016/S0895-7177(02)80029-2.

[4] U. S. Fjordholm and K. O. Lye. *Convergence rates of monotone schemes for conservation laws for data with unbounded total variation.* 2020. DOI: 10.48550/arXiv.2010.07642.

[5] P. Goatin. 'The Aw–Rascle vehicular traffic flow model with phase transitions'. In: *Mathematical and Computer Modelling* 44.3 (2006), pp. 287–303. ISSN: 0895-7177. DOI: 10.1016/j.mcm.2006.01.016.

[6] P. Goatin, S. Göttlich and O. Kolb. 'Speed limit and ramp meter control for traffic flow networks'. In: *Engineering Optimization* 48.7 (2nd July 2016), pp. 1121–1144. ISSN: 0305-215X, 1029-0273. DOI: 10.1080/0305215X.2015.1097099.

[7] B. Goñi-Ros et al. 'Empirical analysis of the causes of stop-and-go waves at sags'. en. In: *IET Intelligent Transport Systems* 8.5 (2014), pp. 499–506. ISSN: 1751-9578. DOI: 10.1049/iet-its.2013.0102.

[8] S. Göttlich, E. Iacomini and T. Jung. 'Properties of the LWR model with time delay'. In: *Networks and Heterogeneous Media* 16.1 (2021), pp. 31–47. ISSN: 1556-1801. DOI: 10.3934/nhm.2020032.

[9] S. Gottlieb, CW Shu and E. Tadmor. 'Strong Stability-Preserving High-Order Time Discretization Methods'. In: *SIAM Review* 43.1 (2001), pp. 89–112. DOI: 10.1137/S003614450036757X.

[10] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual.* 2023. URL: https://www.gurobi.com.

[11] D. Harrison. *A Brief Introduction to Automatic Differentiation for Machine Learning.* 2021. DOI: 10.48550/arXiv.2110.06209.

[12] H. Holden and N. H. Risebro. 'A Mathematical Model of Traffic Flow on a Network of Unidirectional Roads'. In: *SIAM Journal on Mathematical Analysis* 26.4 (July 1995), pp. 999–1017. ISSN: 0036-1410, 1095-7154. DOI: 10.1137/S0036141093243289.

[13] H. Holden and N. H. Risebro. *Front Tracking for Hyperbolic Conservation Laws.* Vol. 152. Applied Mathematical Sciences. Berlin, Heidelberg: Springer, 2015. ISBN: 978-3-662-47506-5. DOI: 10.1007/978-3-662-47507-2.

[14] J. A. Laval and L. Leclercq. 'A mechanism to describe the formation and propagation of stop-and-go waves in congested freeway traffic'. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 368.1928 (Oct. 2010). Publisher: Royal Society, pp. 4519–4541. DOI: 10.1098/rsta.2010.0138.

[15] R. J. LeVeque. *Finite Volume Methods for Hyperbolic Problems.* Cambridge Texts in Applied Mathematics. Cambridge: Cambridge University Press, 2002. ISBN: 978-0-521-00924-9. DOI: 10.1017/CBO9780511791253.

[16] M. J. Lighthill and G. B. Whitham. 'On kinematic waves II. A theory of traffic flow on long crowded roads'. In: *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 229.1178 (Jan. 1997). Publisher: Royal Society, pp. 317–345. DOI: 10.1098/rspa.1955.0089.

[17] M. J. Lighthill and G. B. Whitham. 'On Kinematic Waves. I. Flood Movement in Long Rivers'. In: *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences* 229.1178 (1955), pp. 281–316. ISSN: 00804630. URL: http://www.jstor.org/stable/99768.

[18] G. F. Newell. ‘Nonlinear Effects in the Dynamics of Car Following’. en. In: *Operations Research* (Apr. 1961). Publisher: INFORMS. doi: 10.1287/opre.9.2.209.

[19] J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer New York, NY, 2006. doi: https://doi.org/10.1007/978-0-387-40065-5.

[20] A. Paszke et al. ‘Automatic Differentiation in PyTorch’. In: *NIPS 2017 Workshop on Autodiff.* Long Beach, California, USA, 2017. url: https://openreview.net/forum?id=BJJsrmfCZ.

[21] P. I. Richards. ‘Shock Waves on the Highway’. In: *Operations Research* 4.1 (Feb. 1956). Publisher: INFORMS, pp. 42–51. issn: 0030-364X. doi: 10.1287/opre.4.1.42.

[22] A. Verma. ‘An introduction to automatic differentiation’. In: *CURRENT SCIENCE* 78.7 (2000).

Appendix

A Analytical Solution of the Transport Equation

We will in this section derive an analytical solution of the problem

$$\rho_t + (\rho(1-\rho))_x = \rho_t + (1-2\rho)\rho_x = 0, \tag{70}$$

for a specific choice of initial data.

Let ρ be defined on the domain $[-1,2]\times\mathbb{R}^+$ and the initial data be given as

$$\rho(x,0) = \rho_0(x) = \begin{cases} \rho^L & \text{if } x \leq 0, \\ \rho^L + (\rho^R - \rho^L)x & \text{if } 0 < x \leq 1 \\ \rho^R & \text{if } 1 < x. \end{cases}$$

We will make use of the method of characteristics to obtain the analytical solution. To this end we let $z(t) = \rho(t, x(t))$ along some curve $\mathcal{C} = (t, x(t)) \in \mathbb{R}^2$.

The derivative of z will be

$$z'(t) = \rho_t(t, x(t)) + \rho_x(t, x(t))x'(t).$$

If we let $x'(t) = (1 - 2\rho(t, x(t))) = 1 - 2z(t)$, it is clear that $z'(t) = 0$. In other words ρ is constant along the curve $\mathcal{C}$.

We now consider the domain $[-1,2]\times\mathbb{R}^+$, and let the initial data of the density be given as

$$\rho(x,0) = \rho_0(x) = \begin{cases} \rho^L & \text{if } x \leq 0, \\ \rho^L + (\rho^R - \rho^L)x & \text{if } 0 < x \leq 1 \\ \rho^R & \text{if } 1 < x, \end{cases}$$

where we assume that $\rho^L < \rho^R$.

The expression for the characteristics is given as

$$x = \begin{cases} (1-2\rho^L)t + x_0 & \text{for } x_0 < 0, \\ (1 - 2\big(\rho^L + (\rho^R - \rho^L)x_0\big))t + x_0 & \text{for } 0 < x_0 < 1, \\ (1-2\rho^R)t + x_0 & \text{for } 1 < x_0. \end{cases}$$

Since $\rho^L < \rho^R$, the characteristics coming from $x = 0$

$$x = (1-2\rho^L)t$$

and from $x = 1$

$$x = (1-2\rho^R)t + 1$$

collide at a point (x_s, t_s). By inserting $t = t_s$ in the two expressions and setting them equal to each other we obtain

$$(1-2\rho^L)t_s = (1-2\rho^R)t_s + 1 \Leftrightarrow t_s = \frac{1}{2(\rho^R - \rho^L}).$$

Thus, the point and time of collision will be

$$(x_s, t_s) = \Big(\frac{1-2\rho^L}{2(\rho^R-\rho^L)}, \frac{1}{2(\rho^R-\rho^L)}\Big).$$

It can also be shown that the characteristics coming from $0 < x < 1$ will collide in this point at time $t = t_s$.

Thus, for times $t < t_s$ we have the solution

$$\rho(x,t) = \begin{cases} \rho^L & , x < (1-2\rho^L)t \\ \rho^L + (\rho^R - \rho^L)\Big(\frac{x-(1-2\rho^L)t}{1-2(\rho^R-\rho^L)t}\Big) & , (1-2\rho^L)t < x < (1-2\rho^R)t + 1 \\ \rho^R & , (1-2\rho^R)t + 1 < x. \end{cases}$$

For times $t >= t_s$, a shock will form, and we use the Rankine-Hugoniot condition to get the shock solution.

The Rankine-Hugoniot condition is a condition on the speed of the shock, which says that the speed must satisfy

$$s'(t) = \frac{f(\rho^-) - f(\rho^+)}{\rho^- - \rho^+} = \frac{\rho^L - (\rho^L)^2 - (\rho^R - (\rho^R)^2)}{\rho^L - \rho^R} = 1 - (\rho^L + \rho^R).$$

As mentioned before, the shock starts at position (x_s, t_s), so the equation for the shock is given as

$$s(t) = (1 - (\rho^L + \rho^R))(t - t_s) + x_s.$$

With $\rho^L = \frac{1}{3}$ and $\rho^R = \frac{3}{4}$ the initial data is given as

$$\rho_0(x) = \begin{cases} \frac{1}{3} & \text{if } x \le 0, \\ \frac{1}{3} + \frac{5}{12}x & \text{if } 0 < x \le 1 \\ \frac{3}{4} & \text{if } 1 < x. \end{cases}$$

The shock starts at the point

$$(x_s, t_s) = \Big(\frac{2}{5}, \frac{6}{5}\Big),$$

and the solution for times $t < t_s$ is given by

$$\rho(x,t) = \begin{cases} \frac{1}{3} & , x < \frac{1}{3}t \\ \frac{1}{3} + \frac{5}{12}\Big(\frac{x-\frac{1}{3}t}{1-\frac{5}{6}t}\Big) & , \frac{1}{3}t < x < 1 - \frac{1}{2}t \\ \frac{3}{4} & , 1 - \frac{1}{2}t < x. \end{cases} \tag{71}$$

The shock starting at position (x_s, t_s) is given by the equation

$$s(t) = -\frac{1}{12}\Big(t - \frac{6}{5}\Big) + \frac{2}{5},$$

and the solution for times $t > t_s$ has the form

$$\rho(x,t) = \begin{cases} \frac{1}{3} & \text{if } x < s(t) \\ \frac{3}{4} & \text{if } x > s(t). \end{cases}$$