UNIVERSITA’ DEGLI STUDI DI CAGLIARI
Facoltà di Ingegneria

Dipartimento di Ingegneria Elettrica ed Elettronica

DELFT UNIVERSITY OF TECHNOLOGY
Delft Center for Systems and Control

Freeway Traffic Modeling and Calibration for the Eindhoven Network

Relatori:
Prof. Alessandro Giua
Prof. Bart De Schutter
Dr Monique van den Berg

Tesi di Laurea di:
Federica Lamon

Ringraziamenti

Desidero ringraziare tutti coloro che mi hanno offerto il loro supporto nella preparazione di questa Tesi. Un ringraziamento particolare va ai miei relatori esteri Prof. Bart De Schutter e Monique van den Berg, del Delft Center for Systems and Control dell’ Università Tecnica di Delft, per avermi supportato durante lo svolgimento di questo lavoro e al Prof. Alessandro Giua, relatore italiano, ordinario del dipartimento di Ingegneria Elettrica ed Elettronica dell’Università degli Studi di Cagliari, per avermi dato sempre buoni consigli e per avermi permesso di poter svolgere la mia tesi all’estero in una delle università più prestigiose d’Europa.

Sommario

La congestione del traffico nelle autostrade è un serio problema nella società moderna. La creazione di nuove strade è una soluzione non sempre accettabile. La gestione dinamica del traffico è pertanto una valida alternativa che mira a migliorare l'efficienza delle reti autostradali esistenti. La tesi si apre con una breve introduzione che presenta le principali cause di congestione del traffico, gli strumenti di controllo (semafori, limitatori di velocità, messaggi di informazione sulla viabilità) e la strategia di controllo che permette una gestione dinamica ed efficiente del traffico mirata a prevenire gli ingorghi. Lo scopo principale di tale Tesi è stato analizzare una sezione di rete autostradale di Eindhoven (Paesi Bassi) e studiarne il modello analitico (METANET), appartenente alla famiglia dei modelli macroscopici, in cui il traffico viene descritto in termini aggregati, ossia in termini di velocità media, flusso medio e densità media. Tale modello è stato in seguito implementato in MATLAB e calibrato, ossia è stata effettuata una stima ottima dei parametri caratteristici che permette di poter ottenere delle uscite (flusso di traffico, densità di traffico e velocità media) congruenti con i dati provenienti dal software di simulazione di traffico Paramics Quadstone. Questo modello, opportunamente calibrato, potrà quindi essere utilizzato per poter studiare l'andamento del traffico e consentire una gestione efficiente degli strumenti di controllo.

Abstract

Traffic congestion in the freeways is a serious problem for modern society. Building new roads is a solution not always good. The dynamic traffic management is therefore a valid alternative that aims to improve the efficiency of the existing networks. The thesis starts with a brief introduction about the principal causes of the traffic congestion, the control tools (ramp metering, dynamic speed limits, route guidance) and the control strategy that allows the dynamic traffic management to improve the efficiency and to prevent the traffic jams. The main purpose of this thesis is to analyze a section of the freeway network of the city of Eindhoven (The Netherlands) and to study its analytical model (METANET) that belongs to the set of macroscopic freeway models where traffic is described in aggregate terms such as average speed, average flow, and average density. Then a MATLAB implementation of this model is given, in order to calculate and to optimize its characteristic parameters (calibration), or rather find optimal parameters that allow the model outputs (traffic outflow, traffic density, mean speed) to be in a good consistence with the data coming from the traffic simulation software Paramics Quadstone. This calibrated model can be used to study the traffic and to allow an efficient management of the control tools.

Contents

Chapter 1

Introduction

Traffic congestion is a serious problem in the modern society. The rapid increase of the vehicles number on the roads, due to an elevated need for mobility to which the civil plans and the systems of transport have not had the time to conform, leads to traffic jams that cause considerable costs not only due to the unproductive time losses, but they also decrease the safety of the roads with the possibility of accidents and causes air pollution. The areas that mostly suffer from these problems are the city centers and the freeways.

To give an answer to the problem of the traffic jams it is necessary to select a mobility management policy, that can include:

- Extend the road network.
- Integrated net of the means of public transport : bus, streetcar, trains, etc.
- Better use of existing infrastructure.

Adding lanes and creating alternative new freeway connections or integrating public transport is possible, but this global reorganization of the mobility is a long term project that requires big investments and in most of the cases involves numerous neighboring areas.

Dynamic traffic management is an alternative that aims to increase the safety and efficiency of the existing traffic networks. On the short term this can be seen as one of the better measures against traffic congestion.

The main focus of this thesis is on freeway traffic systems. Freeway networks offer to the network users a lot of possible routes connecting each origin-destination pair in the network. They are very complex and drivers who are not familiar with the daily traffic conditions in the network, usually follow the shortest path to their destination or they follow the indications of the direction signs installed at important bifurcation nodes. During rush hours this, however, may lead them to the tail of a traffic jam congestion while traffic flow on alternative paths is fluid.

Also, traffic flow on freeways is a complex process with many interacting components and random perturbations such as traffic jams, stop-and-go waves, hysteresis phenomena. These perturbations propagate from upstream to downstream road sections, forming forward waves. During traffic jams drivers are slowing down when they observe traffic congestion causing upstream propagation of the jam.

The purpose of the traffic systems is to help achieve full utilization of the highway network capacity and reduce trip times, congestion, and accidents against traffic congestion on freeways. Such systems can influence the pattern of route choice by providing early traffic incident detection and management, then redistribute traffic among the facilities of a corridor or a network by using the excess capacity in some parts of the network. The advanced traffic information systems can provide drivers with information on congestion, navigation and location, traffic conditions, and alternative routes.

The main purpose of this thesis is to analyze a section of the freeway network of the Dutch city of Eindhoven in order to give an analytical model that can represent in a faithful way the "real" system simulated by a traffic simulation software. This model, opportunely calibrated, will be used to study the traffic and to allow an efficient management of the control measures. This last subject in this thesis is not describe in detail, but we will give only some overview to explain the context problem.

The thesis starts with a brief introduction that presents the principal causes of the traffic congestion and it continues, in the second chapter, with a literature survey where a characterization of freeway traffic problems is given and a motivation for the traffic control problem statement is discussed. In particular we will talk about the main control measures (ramp metering, variable speed limits, route guidance) used to allow a dynamic and efficient traffic management policy aimed to prevent the traffic jams.

In the third chapter a setup of the case study for the network of Eindhoven is given. Here we will talk about the analytical model METANET used to describe the network and about the calibration technique used to have a very realistic model. We will introduce also the microscopic traffic simulation model Paramics, and we will give a brief introduction to MPC (Model Predictive Control).

In the fourth chapter we will describe the Matlab implementation of the freeway network, and we will give a characterization of the two main Matlab optimization function, *fmincon* and *patternsearch*, used to calibrate the model.

In the fifth chapter we will show the results of the optimization with both methods, *fmincon* and *patternsearch*, and we will highlight to the faults of the first and success of the second.

In the last part of the thesis we give the conclusion and we talk about the future research.

Chapter 2

Literature survey of freeway control systems

In this section we give an overview of freeway control systems. First, existing control measures are described and next, different control strategies are presented.

2.1 Traffic control measures

The main traffic control measures used to control freeway traffic network are ramp metering, variable speed limits and route guidance. In this section we give an overview of control measures that are used to improve traffic flow. For each control measure we present the control methods found in the literature.

2.1.1 Ramp metering

Ramp metering is one of the most applied freeway traffic control measures. Ramp metering determines the flow rate at which vehicles can enter the freeway. The flow at the on-ramp is controlled by a traffic light and the flow rate is determined by selecting appropriate red, green and amber light timings.

Ramp-metering can be used in two modes: the traffic spreading mode and the traffic restricting mode.

In the traffic spreading mode the metering rate equals the average arrival rate of the vehicles at the on-ramp and its purpose is to spread the vehicles that enter the freeway.

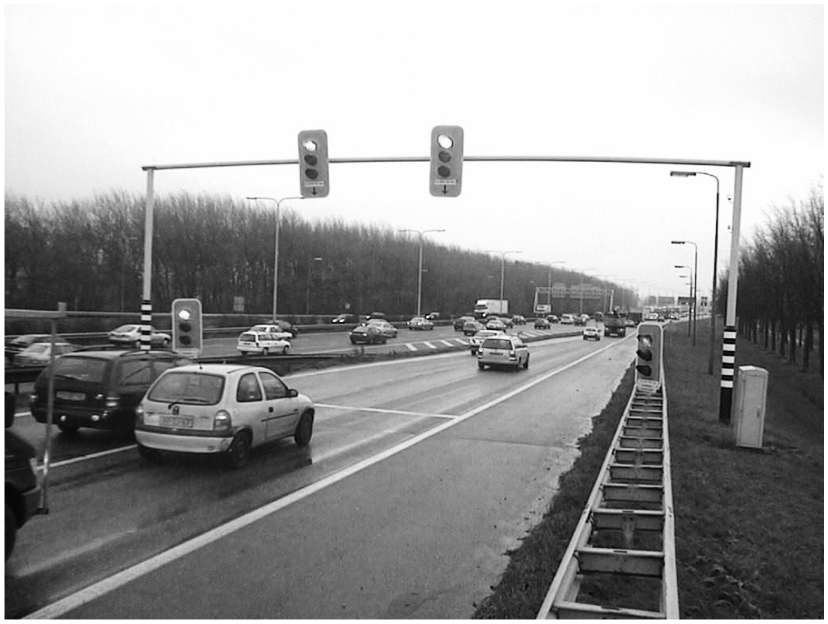


Figure 2.1: Ramp metering

Restrictive ramp metering can be used for two different purposes:

- When traffic is dense, ramp metering can prevent a traffic breakdown on the freeway by adjusting the metering rate such that the density on the freeway remains below the critical value [1].
- When drivers try to bypass congestion on a freeway by taking a local road ramp metering this can increase travel times and discourage the use of the bypass.

Restrictive ramp metering can be classified as: fixed-time or traffic-responsive; static or dynamic; local or coordinated [2].

Fixed-time strategies are determined off-line (on modern freeways a large amount of data is available on-line and off-line that can serve as a basis for choices of appropriate control measures) based on historical demands, where the demands and splitting rates at off-ramps are assumed to be constant in a given time slot, e.g., in the morning rush hour. This approach typically considers on-ramps and off-ramps along one freeway stretch, but it is not difficult to extend to freeway networks. As control objective one may choose to maximize the number of served vehicles, to maximize the total traveled distance, or to balance ramp queues. The disadvantage of fixed-time strategies is that they do not take into account the traffic demand variations during a day or from day-to-day, which may result in underutilization of the freeway or inability to prevent congestion. Traffic-responsive strategies adjust on-line the metering rate as a function of the prevailing traffic conditions. These strategies typically aim at the same objectives as the fixed-time strategies, but also at preventing congestion. The traffic conditions are periodically fed into the controller to determine its control strategy.

Usually these measures operate based on local data, or rather, data obtained at the place where the measures devices operate, and therefore uncorrelated with the data obtained at the other places. However, considering the effect of coordinated strategies, coordinating all local data, with a global purpose, has many advantages compared to local control. Local control measures in fact, could influence the traffic flows in more distant parts of the network, for example, an improved flow may cause congestion somewhere else in the network or a reduced flow may prevent congestion somewhere else in the network. So, they should be coordinated such that they serve the same objectives. The measures in this way operate in a global level , or better in a network level, higher than local level. For example [2], solving a local congestion only, may have as consequence that the vehicles run faster into another (downstream) congestion, whereas still the same amount of vehicles have to pass the bottleneck (with a given capacity), and so the average travel time on the network level will still be the same. Furthermore, if dynamic origin-destination (OD) data is available, control on the network level can take advantage of the predictions of the flows in the network. For example, during peak hours the density on the main-stream (freeway) can be so high that the queue on an on-ramp spills back to the surface streets of the city, whereas (pro-active, coordinated) metering of upstream on-ramps could reduce the density of the main-stream flow and prevent spill back of the on-ramp queue. Local controllers are not able to use OD information because the flows arriving at the local controller depend on the actions of other controllers elsewhere in the network, which are unknown.

A number of studies have simulated ramp metering for different transportation networks and traffic scenarios, with different control approaches, and with the use of microscopic and macroscopic traffic flow models. Generally the total network travel time is considered as the performance measure and is improved until to 30% when using ramp metering. Since the total time spent in the network is strongly dependent on the combination of the scenario (which determines the inflow or demand of the network) and on the control method (which determines the outflow of the network), these figures are encouraging but no guarantee for success in general.

2.1.2 Dynamic speed limits

Another kind of traffic control measure is dynamic speed limits. Many modern freeways are equipped with variable speed limits signs (see Figure (3.14)).

The main purpose is to eliminate or to reduce the effects of shock waves that can bring to congestion and to increase the possibility of accidents. However, attempts are also made

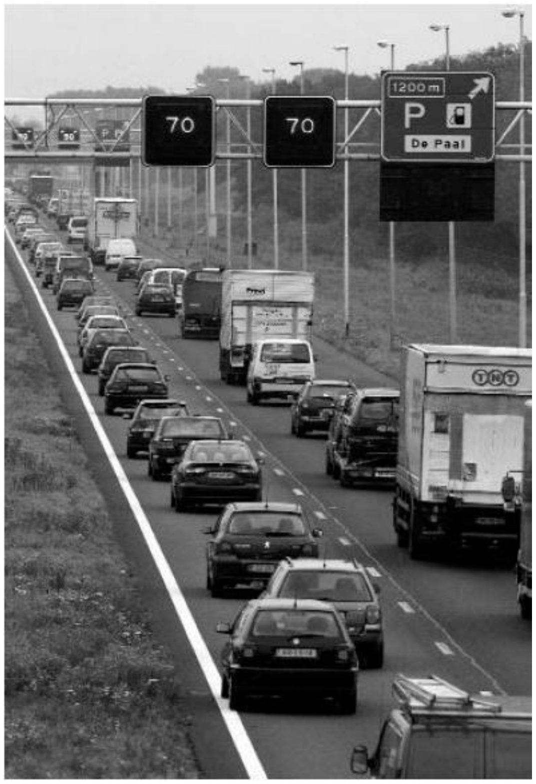


Figure 2.2: Speed limits sign in a modern freeway

to increase the traffic flow by more complex switching schemes. In literature basically two views on the use of speed limits are found:

- Homogenization effect [3],[4]: speed limits can reduce the speed differences between vehicles which is expected to result in a higher (and safer) traffic flow. The homogenization approach typically uses speed limits that are close to, but above the critical speed. This approach can increase the time to breakdown but it cannot suppress or resolve shock waves.
- Traffic breakdown prevention approach [5], [6]: it is more focused on the prevention of traffic breakdown and is developed using neural networks. It focuses more on preventing too high densities, and also allows speed limits that are lower than the critical speed in order to limit the inflow to these areas. By resolving the high density areas (bottlenecks) higher flow can be achieved. In contrast to the homogenization approach, this approach can also resolve existing jams.

2.1.3 Route guidance

Route guidance systems suggests the route when more alternative routes exist to a destination. The systems typically display traffic information on VMSs (Variable Message Sign) that point out the alternative routes or the delay on the alternative routes and on DRIPs (Dynamic Route Information Panel) that display messages about queue length or instantaneous travel time [7]. [8].

It is important underline that they do not directly determine the splitting rate: the drivers make their own decisions. One of the main difficulties associated with control by means of route guidance (but with variable speed signs too) is to assess the effect on driver behavior. For this reason the information given by the DRIP and VMS has to be credible. When this is not the case, many road users will stop complying to the advice, especially the road users who drive the route frequently.

In route guidance the notions system optimum and user equilibrium (or user optimum) play an important role [2]. The system optimum is achieved when the vehicles are guided such that the total costs (travel time or travel distance) of all drivers is minimized. However, the system optimum does not necessarily minimize the travel time for each individual driver. So, some drivers may have the choice for another route that has lower cost (shorter individual travel time). The traffic network is in user equilibrium when the costs on each utilized alternative route is equal and minimal, and on routes that are not utilized the cost is higher that on the utilized routes. This means that no driver has the possibility to find another route that reduces his individual cost. If the cost function is defined as the travel time it is typically

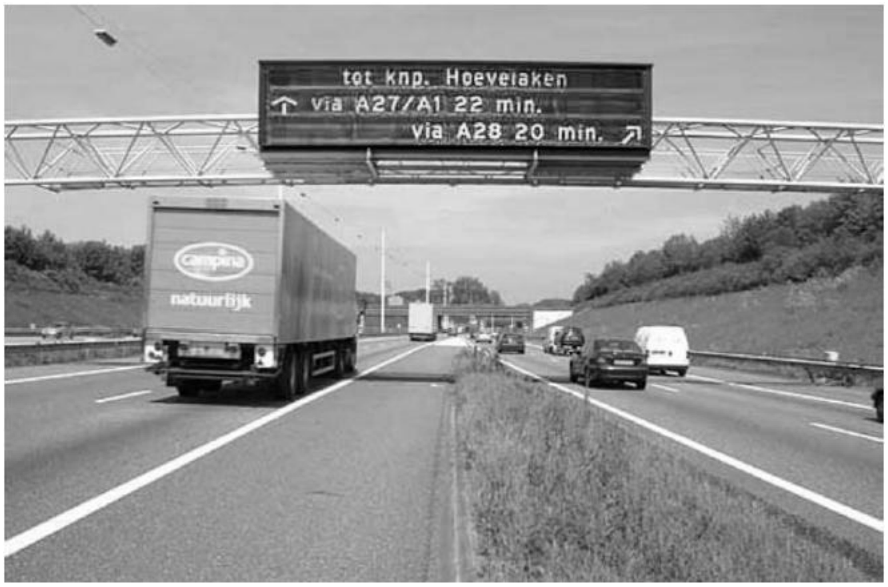


Figure 2.3: Route guidance in a modern freeway

defined as the predicted travel time or as the instantaneous travel time (or reactive travel time). The predicted travel time is the time that the driver will experience when he drives along the given route, while the instantaneous travel time is the travel time determined based on the current speeds on the route. In a dynamic setting these speeds may change when the driver travels over the route, and consequently the instantaneous travel time may be different from the predicted travel time. Papageorgiou in *Dynamic modeling assignment and route guidance in traffic networks* [9] and Papageorgiou and Messmer in *Dynamic network traffic assignment and route guidance via feedback regulation* [10] have developed a theoretical framework for route guidance in traffic networks. In the first paper a macroscopic modeling framework for dynamic traffic phenomena on multidestination freeway and/or road networks with time varying demands is developed. Key variables of the model at each network node are the splitting rates of each traffic subflow with a specific destination. Two approaches are investigated for resolving the dynamic assignment and the route guidance problem: first, an optimal control approach for achieving a dynamic system or user optimum; second, a feedback concept for establishing dynamic user optimal conditions. In the second paper a deterministic, macroscopic modeling framework for dynamic traffic phenomena on networks consisting of freeways and urban streets is presented for nonelastic but time-varying traffic demands. A feedback methodology is applied to the network model to establish dynamic traffic assignment conditions. Specifically, a multivariable feedback regulator are developed and tested for a particular network traffic model. So, three different traffic control problems are formulated: an optimal control problem to achieve system optimum (minimize travel time), an optimal control problem to achieve user optimum (equalize travel times), and a feedback control problem to achieve user optimum (equalize travel times). The resulting controller strategies are demonstrated on a test network with six pairs of alternative routes.

The feedback control strategy is tested with instantaneous travel times and results in a user equilibrium for most alternative routes, and the resulting travel time is very close to the system optimum.

Kraan et al. [11] present an extensive evaluation of the impact on network performance of VMSs on the freeway network around Amsterdam. Several performance indicators are compared before and after the installation of 14 new VMSs. The performance indicators used for comparison are:

- Total traveled distance (veh·km) by all vehicles in the network during the peak period.
- Total congestion length and duration (km·min) occurring in the network during the peak period, where congestion is defined as traffic traveling at speed of 35 km/h or lower.
- Instantaneous travel time delay (veh·h) the delay for all drivers during the peak period, based on instantaneous travel time calculations.

The performance indicators are compared for alternative routes and for most locations a small but statistically significant improvement is found. The day-to-day standard deviation of these indicators decreased after the installation of the VMSs, which indicates that the travel times have become more reliable. In the paper [11] the user response to VMSs messages (showing congestion lengths) is also analyzed. It is found that for each additional kilometer of queue length displayed for a route leads to a reduction of between 0.8% and 1.6%.

2.2 Freeway models

In this section we will talk about the different kinds of model that can describe a freeway.

Traffic models may be distinguished according to the level of detail at which they describe the microscopic, mesoscopic or macroscopic traffic process [2]:

- **Microscopic models** describe the behavior of vehicles individually. Important aspects of microscopic models are the so-called car-following and lane changing behavior. Car-following and lane-changing behavior is generally described as a function of the distance to and (relative) speed of the surrounding vehicles, and the desired speed of the actual vehicle. Since the vehicles are modeled individually in microscopic traffic models, it is easy to assign different characteristics to each vehicle. These characteristics can be related to the driving style of the driver (aggressive, patient), vehicle type (car, truck), its destination, and chosen route.

- **Mesoscopic models** do not track individual vehicles, but describe the behavior of individual vehicles in probabilistic terms.
- **Macroscopic models** use a high level of aggregation without distinguishing between individual vehicle actions such as a lane change. Instead traffic is described in aggregate terms such as average speed, average flow, and average density.

There is also another possible classification for traffic flow model based on the intended application:

- **Assessment of traffic control strategies** with a simulation model instead of a field operation test has several advantages. Above all, simulation is cheaper and faster, but it also provides an environment where the unpredictable disturbances of a field test, such as weather influences, traffic demand variations, and incidents, can be excluded, or if necessary simulations can be repeated under exactly the same disturbance scenario.
- **Model-based traffic control** makes use of an internal prediction model in order to find the best traffic control measures to be applied to the real traffic process. Since these models are operated in real-time, and are often used to evaluate several control scenarios, they need to be fast when executed on a computer [3].
- **Design of new transportation facilities**, e.g., geometric design of infrastructure can benefit from simulations that confirm that the design meets the specifications.
- **Training of traffic operators** in traffic control centers is supported by simulations that instantly give feedback about the consequences of the actions of the traffic operators in a certain situation.

In this thesis we used the model-based macroscopic traffic flow model METANET described by Kotsialos and Papageorgiou [12],[13]. This model will be used throughout this thesis for the simulation of freeways. This model was chosen because it provides a good trade-off between simulation speed and accuracy. The fact that this model is deterministic, discrete-time, discrete-space, and macroscopic makes it very suitable for model-based traffic control of which we will talk in the next section. In METANET [14] the freeway network is represented as a graph with nodes and links, where the links correspond to freeway stretches with uniform characteristics; the nodes are placed at on-ramps and off-ramps, and where two or more freeways connect, or where there is a change in the characteristics. Links are divided into one of more segments with a length of about 300 m. The evolution of the traffic system is characterized by macroscopic variables for each segment as traffic density, traffic flow and mean speed. The original METANET does not explicity describe the effect of speed limits, but an extension to the METANET model is provided [15],[16]. The METANET model describes how these state variables evolve over time but, before we can use it to predict the evolution of the traffic situation, the model needs to be calibrated and validated [17], [18]. Empirical calibration and validation using real-world traffic data is necessary to assess the accuracy of any macroscopic model. This study aims to construct an automated procedure to find the optimal parameters for a given network using systematic method. For this purpose we use an implemented optimization function in which the model parameters are variables to be estimated and the total error between model outputs and measured data is the objective function to be optimized. This subject, core of this thesis, will be explained better in the next chapters.

2.3 Control strategy

Network-oriented traffic control is based on coordination of control measures and prediction. In fact, determining the effects of control measures on distant parts of the network and coordinate them also involves prediction, because the effect of a control measure on more distant locations will only be visible after some time.

In the literature different approaches exist for coordinating traffic control measures:

- Model-based optimal traffic control technique. It uses the internal prediction model, of which we told before, in order to find the best traffic control measures to be applied to the real traffic process. It is based on:

  1. the current traffic state;
  2. the expected traffic demand on the network level, possibly including origin-destination relationships, and other possible external influences, such as weather conditions;
  3. the planned traffic control measures;

- Knowledge-based traffic control method [19]. It typically describes the knowledge about the traffic system in combination with the control system in terms that are comprehensible for humans. Via reasoning mechanisms the knowledge-based system generates a solution (control measure) given the current traffic situation. A typical motivation for these systems is to help traffic control center operators to find good (not necessarily the best) combinations of control measures. The operators often suffer from cognitive overload by the large number of possible actions (control measures) or by time pressure in case of incidents.The possibility for the operators to track the reasoning path of the knowledge-based system makes these systems attractive .

- Case-based reasoning, an emerging artificial intelligence (AI) paradigm, for real-time traffic routing [20]

- Nonlinear optimization methods, developed by A. Messmer and M. Papageorgiou [21].
- Expert Systems which determine the control outputs on the basis of decision rules extracted from expert knowledge [22] [23]. This includes also fuzzy control as a particular realization scheme.
- Neural Networks which are trained to produce the appropriate control inputs for typical traffic situations [24], [25].
- Optimal control procedure based on Powell's method [3].

The characteristics of nonlinear optimization techniques are very useful in the context of freeway network control because the nonlinear density-flow relationship of freeway stretches is an essential nonlinearity. Substantial variations in traffic dynamics between low and high traffic demand situations (free flow, congestion, shock waves) are mainly caused by this nonlinearity and because control measures are subject to strict control constraints. On the other hand, this method has some drawbacks:

- A mathematical state-space model of the process is necessary and should include the description of all relevant phenomena for adequate control behavior and performance. A high level of complexity is reached and it may be confronted by the utilization of an expert system or a neural network. Actually, in expert system design, the effort of specifying the process behavior is replaced by the necessity of collecting sufficient expert knowledge and of translating it into appropriate rules. Due to the lack of a general theoretical background, this may be a task with at least the same degree of difficulty as for deriving a mathematical process model. Some expert systems even use both on-line models and decision rules [22]. However, once a process model has been derived, it can be regarded as a great advantage that nonlinear optimization allows for consideration of the model in full detail.
- Compared to the other approaches, the computational effort in real time is very high. It can be really severe for complex processes with relatively fast process dynamics. The rapid development of the numerical performance of computer hardware, however, makes computationally intensive control concepts, such as on-line optimization, more and more feasible and also reasonable in terms of cost.

In this thesis we chose to use a particular model-based optimal traffic control technique, Model Predictive Control (MPC), to which is dedicated the next subsection.

2.3.1 Model Predictive Control

MPC (Model Predictive Control) is a model-based optimal control method applied in a rolling horizon framework [26],[27],[15] and it uses the METANET prediction model. With this method the optimal control signals is determined by minimizing an objective function, that expresses the performance of the traffic network (as a function of a given control input), over a given prediction horizon. This objective function is the total time spent (TTS) by the vehicles in the network. MPC uses a receding horizon framework in which only the first sample of control signals are implemented while the others are discarded and recalculated during the next iteration. Once the first sample is applied to the system, the state (and/or the model parameters) are updated using measurements and next the whole process is repeated with the control and the prediction horizon shifted one sample forward. In this way we obtain an adaptive control strategy that is robust for small changes in the system parameters, noise, and small disturbances and measurement errors.

Optimal control is successfully applied by Kotsialos and Papageorgiou [8] to coordinate or integrate traffic control measures. Both optimal control and MPC have the advantage that the controller generates control signals that are optimal according to a user-supplied objective function. However, MPC has some important advantages over the traditional optimal control, and to understand them it is important to divide traffic congestion into two types: recurrent and nonrecurrent [2]. Recurrent congestion occurs during rush hours, corresponding to traffic peaks. This type of congestion is easily predictable, and times-of-day or days-of-year control strategies can be adopted. It is more difficult to cope with nonrecurrent congestion, due to unpredictable circumstances such as, for instance, incidents. Recurrent congestion may be solved by an open-loop strategy: the past data are needed to find the best access rates. A closed-loop strategy is more reliable, because it also deals with nonrecurrent congestion. So a first important advantage of MPC compared to optimal control is that the last one has an open-loop structure, which means that the disturbances (in our case: the traffic demands) have to be completely and exactly known before the simulation, and the traffic model has to be very accurate to ensure sufficient precision for the whole simulation. MPC operates in closed-loop which means that the traffic state and the current demands are regularly fed back to the controller, and the controller can take disturbances into account and correct for prediction errors resulting from model mismatch. Second, adaptivity is easily implemented in MPC, because the prediction model can be changed or replaced during operation. This may be necessary when traffic behavior changes significantly (e.g., in case of incidents, changing weather conditions, lane closures for maintenance). Third, for MPC a shorter prediction horizon is usually sufficient, which reduces complexity, and makes the real-time application of MPC feasible.

In the Figure (2.4) a schematic representation of the dynamic traffic management systems realized by MPC is given [2]. The traffic sensors provide information about the current traffic

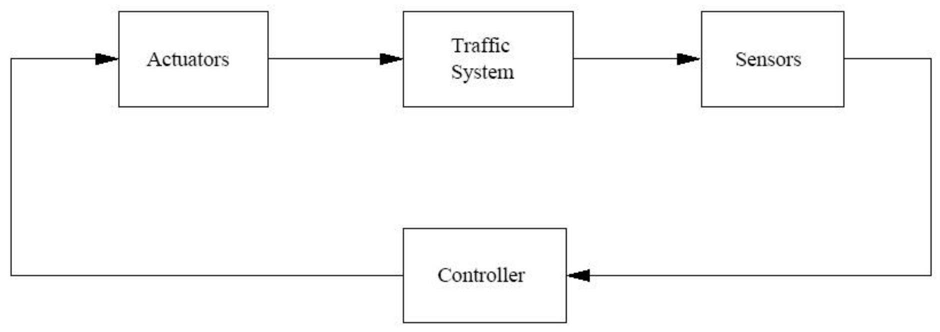


Figure 2.4: Schematic representation of the dynamic traffic management control loop [2]

state, such as speed, flow, density. The controller determines appropriate control signals that sent to the actuators. The reaction of the traffic system is measured by the sensors again, which closes the control loop. If new measurements show a deviation from the desired traffic system behavior (caused by unforeseen disturbances), the new control signals are adopted accordingly.

Chapter 3

Eindhoven: a case study

In this chapter a freeway traffic problem is considered. The case study concerns the freeway network of the Dutch city Eindhoven. In particular we consider the freeway around the three major junctions in the network (Batadorp, De Hogt and Leenderheide) (see Figures (3.1) and (3.2)).

In the following section we will talk about the analytical model METANET used to describe the network and the calibration technique used to make it more real possible. We will introduce the microscopic traffic simulation model Paramics (Quadstone, 2004), used to get the "traffic data" considerable as real, and finally we will talk about the optimal control method MPC, that will not be described in detail in this thesis.

3.1 METANET model

METANET [15] is the macroscopic analytical model used to describe the network. METANET represents the network as a direct graph with the links (indicated by the index *m*) corresponding to the freeway stretches. On-ramps, off-ramps are segments represented by node. Each link *m* is divided into N_m segments (indicated by the index *i*) of length L_m. In the Figures (3.3) and (3.4) we can observe in a very detail way the case study network. In the Figures (3.3) and (3.4) are shown all the subdivisions in links and segments of our case study network. We have two roadways, one in the Leenderheide-De Hogt direction, and the other one in the De Hogt-Leenderheide direction In this thesis all the results are referred to the road in the direction Leenderheide-De Hogt. The benchmark setup consist of an origin, 4 on-ramps, 4 off-ramps and 13 freeway links divided in segments for a total of 19 segments. The total length of the road is around 6.5 km. Each segment *i* of link *m* is characterized by three quantities:

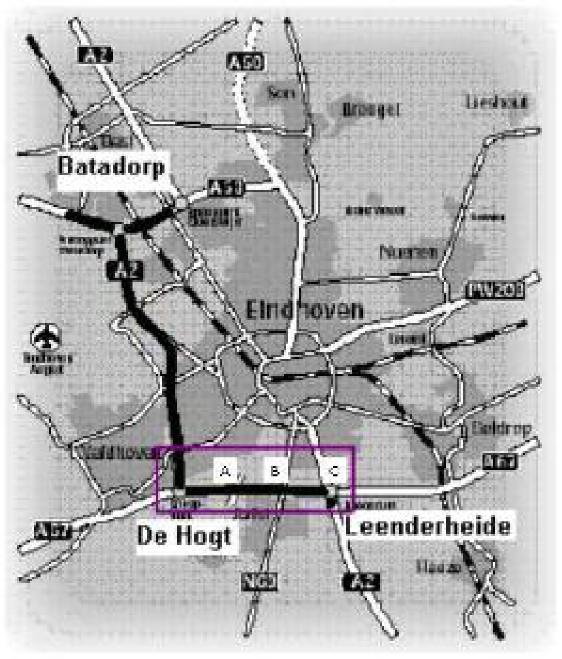


Figure 3.1: The case study network [28]

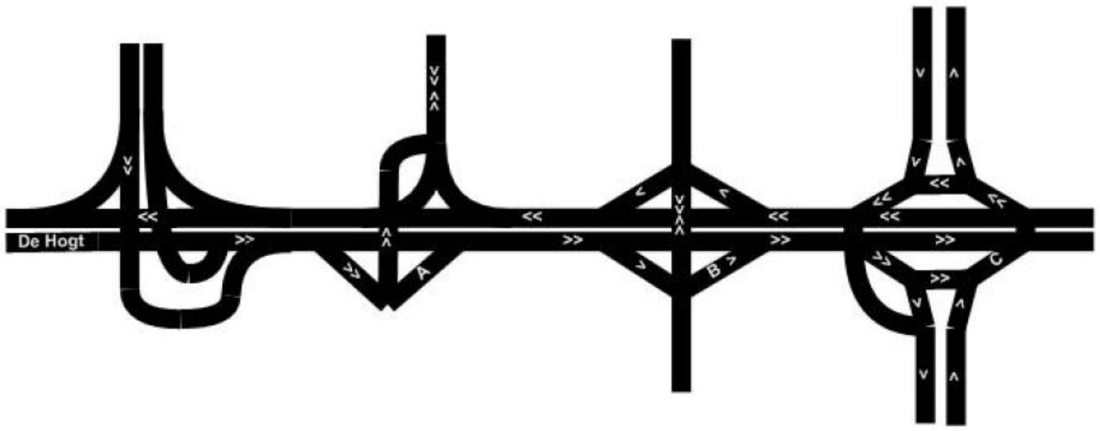


Figure 3.2: Schematic representation of the case study network

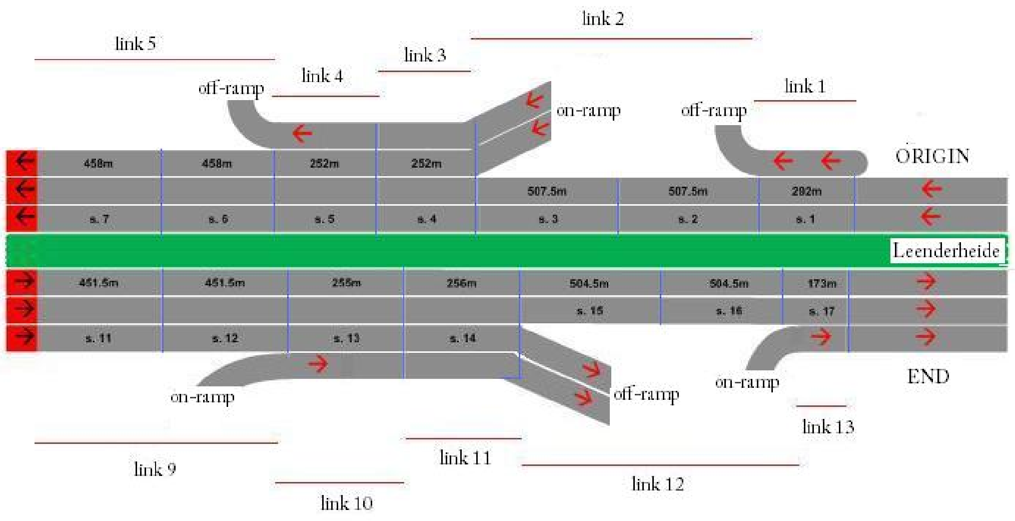


Figure 3.3: Detailed schematic representation of the case study network: first part

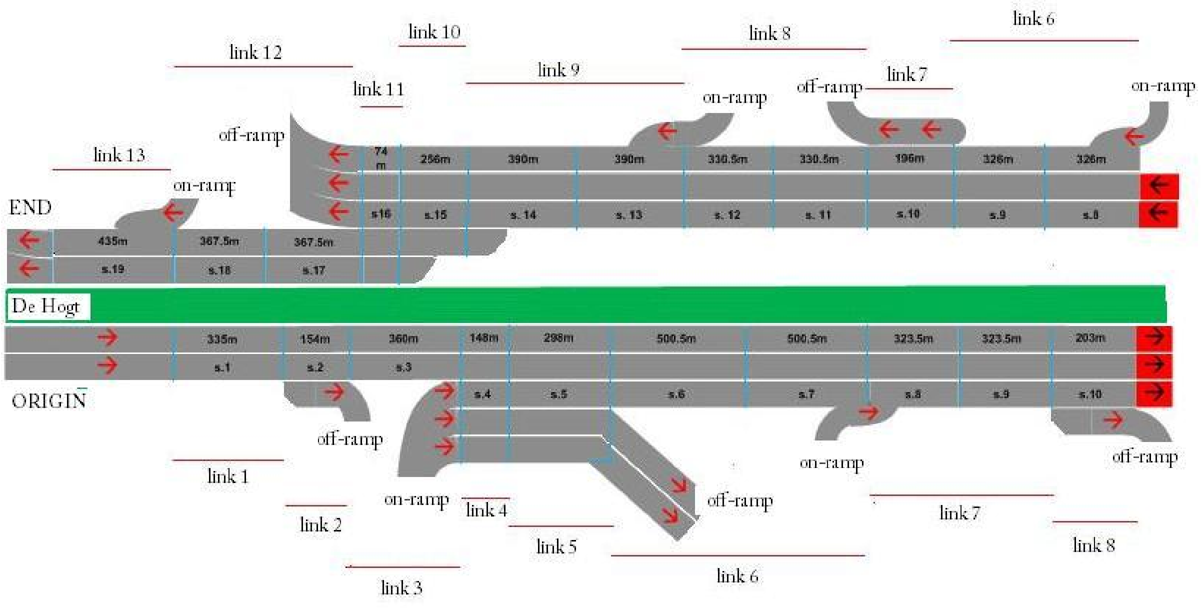


Figure 3.4: Detailed schematic representation of the case study network: second part

- traffic density $\rho_{m,i}(k)$ (veh/km/lane),
- mean speed $v_{m,i}(k)$ (km/h),
- traffic volume or outflow $q_{m,i}(k)$ (veh/h)

where *k* indicates the time instant, and *T* is the time step used for the simulation of the traffic flow. For stability the following relation must be respected:

$$L_m > v_{free,m}T \tag{3.1}$$

where $v_{free,m}$ is the average speed for each link that the drivers assume if traffic is freely flowing. In this project we assume that $v_{free,m}$ is the same for all the links.

The following equationsndescribe the evolution of the network over time.

The outflow of each segment is equal to the traffic density multiplied by the mean speed and the number of lanes on that segment (denoted by λ_m):

$$q_{m,i}(k) = \rho_{m,i}(k) v_{m,i}(k) \lambda_m \tag{3.2}$$

and the density of a segment at the time step $(k+1)$, due to the law of conservation of vehicles is:

$$\rho_{m,i}(k+1) = \rho_{m,i}(k) + \frac{T}{L_m \lambda_m}(q_{m,i-1}(k) - q_{m,i}(k)) \tag{3.3}$$

Both equations (3.2) and (3.3) are based on physical principles and are for this reason exact. The relation between density and desired speed (see equation (3.4)) is instead based on heuristic principles. The mean speed at the simulation step $(k+1)$ is taken to be the mean speed at time instant *k* plus a *relaxation term* that expresses that the drivers try to achieve a desired speed $V(\rho)$, a *convection term* that expresses the speed increase (or decrease) caused by the inflow of vehicles, and an *anticipation term* that expresses the speed decrease (increase) as drivers experience a density increase (decrease) downstream:

$$v_{m,i}(k+1) = v_{m,i}(k) + \frac{T}{\tau}(V(\rho_{m,i}(k)) - v_{m,i}(k)) + \frac{T}{L_m} v_{m,i}(k)(v_{m,i-1}(k) - v_{m,i}(k))$$

$$-\frac{\eta T}{\tau L_m} \frac{\rho_{m,i+1}(k) - \rho_{m,i}(k)}{\rho_{m,i}(k) + \kappa} \tag{3.4}$$

where τ, η and κ are model parameters and where:

$$V(\rho_{m,i}(k)) = v_{free,m} \exp\Big[-\frac{1}{a_m}\Big(\frac{\rho_{m,i}(k)}{\rho_{crit,m}}\Big)^{a_m}\Big] \tag{3.5}$$

is the desired speed, with a_m a model parameter and with the critical density $\rho_{crit,m}$ that is the density at which the traffic flow is maximal. For our model we assume that a_m and $\rho_{crit,m}$ are the same for all the links.

Origins are modeled with a simple queue model. The length of the queue equals the previous queue length plus the demand $d_0(k)$, minus the outflow $q_0(k)$:

$$w_0(k+1) = w_0(k) + T(d_0(k) - q_0(k)) \tag{3.6}$$

The outflow of the origin depends on the traffic conditions on the mainstream and, for the metered on-ramp, on the ramp metering rate $r_0(k)$, where $r_0(k) \in [0; 1]$. More specifically, $q_0(k)$ is the minimum of three quantities: the available traffic in time period *k* (queue plus demand), the maximal flow allowed by the metering rate and the maximal flow that could enter the freeway because of the mainstream conditions:

$$q_0(k) = \min\Big[d_0(k) + \frac{w_0(k)}{T}, Q_0 r_0(k), Q_0\Big(\frac{\rho_{max,m} - \rho_{m,i}(k)}{\rho_{max,m} - \rho_{crit,m}}\Big)\Big] \tag{3.7}$$

where Q_0 is the on-ramp capacity(veh/h) under free-flow conditions, the global parameter ρ_{max} (veh/km/lane) is the maximum density of a segment (also called jam density), and *m* is the index of the link to which the on-ramp is connected. In our case study, we have not ramp metering, so we consider $r_0(k) = 1$ and $Q_0 r_0(k)$ become Q_0.

In order to account for the speed drop caused by merging phenomena, if there is an on ramp the following term is added to (3.4):

$$-\frac{\delta T q_0(k) v_{m,1}(k)}{L_m \lambda_m \rho_{crit,m}} \tag{3.8}$$

where δ is a model parameter.

When there is a lane drop, the speed reduction due to the weaving phenomena is expressed by:

$$-\frac{\phi T \Delta\lambda \rho_{m,N_m}(k) v^2_{m,N_m}(k)}{L_m \lambda_m \rho_{crit,m}} \tag{3.9}$$

where $\Delta\lambda = \lambda_m - \lambda_{m+1}$ and ϕ is a model parameter. Also this term is added to (3.4). In our model we will not use this factor, because our case study network has not lane drops.

If there is a junction or a bifurcation, a node is placed between the links. This node provides the incoming links with a (virtual, when there are more leaving links) downstream density, and the leaving links with an inflow and a (virtual, when there are more entering links) upstream speed.

The flow that enters node *n* is distributed among the leaving links according to:

$$Q_n(k) = \sum_{\mu \in I_n} q_{\mu,N_\mu}(k) \tag{3.10}$$

$$q_{m,0}(k) = \beta_{n,m}(k) \cdot Q_n(k) \tag{3.11}$$

where $Q_n(k)$ is the total flow that enters the node at time *k*, I_n is the set of links that enter node *n*, $\beta_{n,m}(k)$ are the turning rate (the fraction of the total flow through node *n* that leaves via link *m*), and $q_{m,0}(k)$ is the flow that leaves node *n* via link *m*.

When node *n* has more than one leaving link as shown in Figure (3.5), the virtual downstream density $\rho_{m,N_m+1}(k)$ of the entering link *m* is given by 3.12:

$$\rho_{m,N_m+1}(k) = \frac{\sum_{\mu \in O_n} \rho^2_{\mu,1}(k)}{\sum_{\mu \in O_n} \rho_{\mu,1}(k)} \tag{3.12}$$

When node *n* has more than one entering link as shown in Figure (3.6), the virtual upstream speed $v_{m,0}(k)$ of leaving link m is given by 3.13:

$$v_{m,0}(k) = \frac{\sum_{\mu \in I_n} v_{\mu,N_\mu}(k) q_{\mu,N_\mu}(k)}{\sum_{\mu \in I_n} q_{\mu,N_\mu}(k)} \tag{3.13}$$

3.2 Extensions of the METANET model

The original METANET model does not describe the effect of speed limits. For this reason an extension of the original METANET [15] model is necessary. We extend the desired speed equation to incorporate speed limits as in [15].

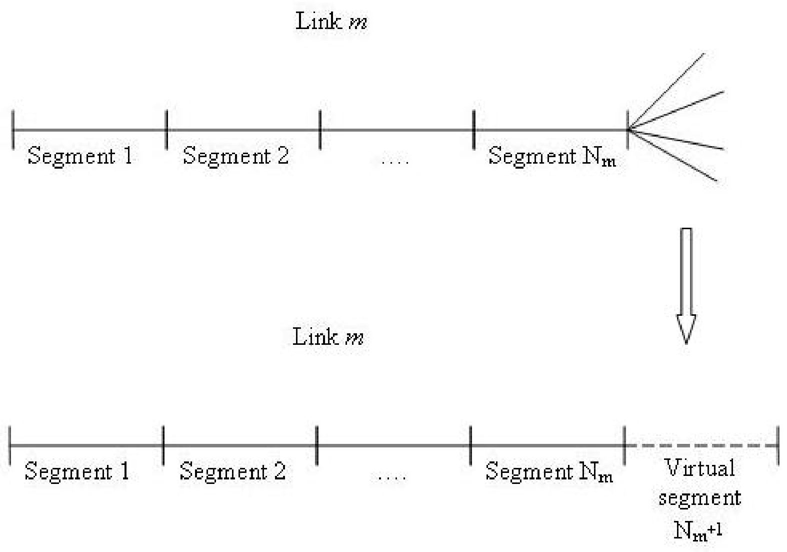


Figure 3.5: A node with one entering link *m* and several leaving links

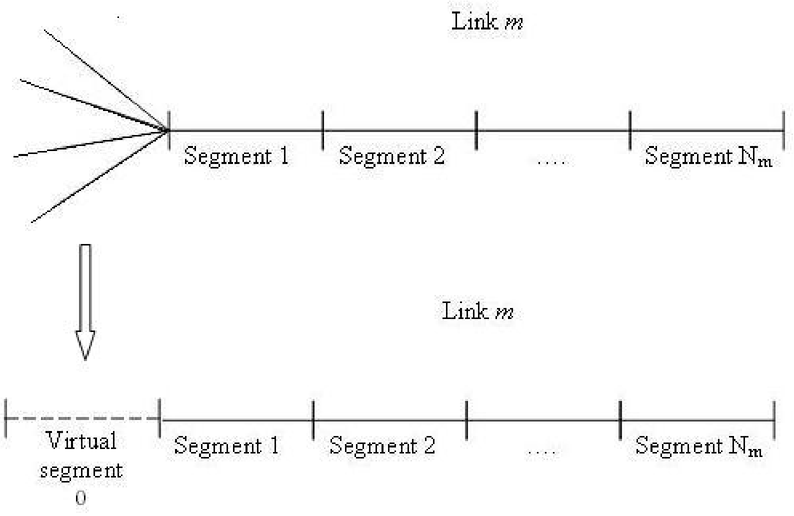


Figure 3.6: A node with one leaving link *m* and several entering links

To get a realistic model, we assume that the desired speed is the minimum of the following two quantities: the desired speed based on the experienced density, and the desired speed caused by the speed limit displayed on the variable message sign (VMS).

$$V(\rho_{m,i}(k)) = \min(v_{free,m} \cdot \exp\Big[-\frac{1}{a_m}\Big(\frac{\rho_{m,i}(k)}{\rho_{crit,m}}\Big)^{a_m}\Big], (1+\alpha)v_{control,m,i}(k)) \tag{3.14}$$

where $v_{control,m,i}(k)$ is the speed limits imposed on segment *i*, link *m*, at time *k*, and $1+\alpha$ is the non compliance factor that expresses that drivers usually do not fully comply with the displayed speed limit and their target speed is usually higher than what is displayed.

The second extension regards the modeling of a mainstream origin, which has a different nature than an on-ramp origin. It is introduced to express the different natures of a mainstream origin link O and a regular on-ramp. To this end, we use a modified version of 3.7 with another flow constraint to model a mainstream origin link, because the inflow of a segment (and thus the outflow of the mainstream origin) can be limited by an active speed limit or by the actual speed in the first segment (when either of them is lower than the speed at critical density). Hence, we assume that the maximal flow equals the flow that follows from the speed-flow relationship from 3.2 and 3.5 and with the speed equal to the speed limit or the actual speed in the first segment, whichever is smaller. So if O is the origin of mainstream link μ, then we have :

$$q_0(k) = \min\Big[d_0(k) + \frac{w_0(k)}{T}, q_{lim,\mu,1}(k)\Big] \tag{3.15}$$

where $q_{lim,\mu,1}(k)$ is the maximal inflow determined by the limiting speed in the first segment of link μ:

$$q_{lim,\mu,1}(k) = \begin{cases} \lambda_\mu \cdot v_{lim,\mu,1}(k) \cdot \rho_{crit,\mu}\Big[a_\mu \ln\Big(\frac{v_{lim,\mu,1}(k)}{v_{free,m}}\Big)\Big] & \text{if} v_{lim,\mu,1}(k) < V(\rho_{crit,\mu}); \\ q_{cap,\mu} & \text{if} v_{lim,\mu,1}(k) \geqslant V(\rho_{crit,\mu}). \end{cases}$$

where

$$v_{lim,\mu,1} = \min(v_{control,\mu,1}(k), v_{\mu,1}(k))$$

is the speed that limits the flow, and

$$q_{cap,\mu} = \lambda_\mu V(\rho_{crit,\mu})\rho_{crit,\mu}$$

is the capacity flow.

We can see in the Figure (3.3) and 3.4 that the case study model has one mainstream origin O, and four on-ramp.

3.3 Calibration and Validation

Before a traffic model can be used to predict the evolution of the traffic situation, it needs to be calibrated and validated [18], or rather the model parameters have to be chose in order to make the state variables of the model in a good consistence with the "real" values. For this case study the microscopic traffic simulation model Paramics (Quadstone, 2004) will be use. Paramics is a suite of software tools used to simulate the movement and behavior of individual vehicles on urban and freeway road networks. Data deriving from these simulation will be used as "real world" data in order to be confronted with the model data. The purpose of the calibration is just minimize the difference between "real" data and model data. To do this we used an optimization function of the optimization toolbox of MATLAB, of which we will talk in Chapter 4.

The state variables of the model are:

- traffic density $\rho_{m,i}(k)$,
- traffic outflow $q_{m,i}(k)$,
- mean speed $v_{m,i}(k)$

at all network locations.

The parameters to estimate are:

- v_{free} is the average speed that drivers assume if traffic is freely flowing,
- a involved in $V(\rho_{m,i}(k))$,
- $\rho_{crit,m}$ is the critical density at which the traffic flow is maximal,
- α is involved to *(*$1+\alpha$*)*, the no-compliance factor,
- η (anticipation factor), τ (relaxation time), κ that are involved to the anticipation term in the dynamical evolution of mean speed $v_{m,i}(k+1)$,
- δ presents the effect of merging phenomena on speed,

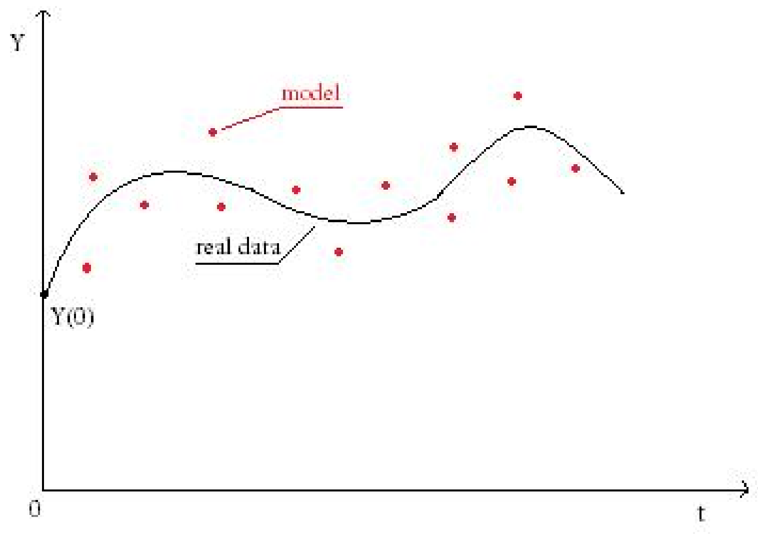


Figure 3.7: Graphical example about calibration method: the differences between model and Paramics data are minimized.

- ρ_{max} is the maximal density.

The calibration is an optimization procedure that minimize the difference between the "real data" coming from Paramics and the data coming from our model. In particular we try to minimize the following objective function:

$$\sum_{h=0}^{N_{samp}} \sum_{m,i \in I_{all}} (q_{m,i}^{model}(h) - q_{m,i}^{sim}(h))^2 + \xi(v_{m,i}^{model}(h) - v_{m,i}^{sim}(h))^2 \tag{3.16}$$

where $N_{samp}$is the number of simulation time step into the entire simulation period,I_{all} is the set of indexes of all pairs of links and segments, and ξ is a tuning weight.

We give an example in the Figure (3.7) to explain better what we are going to do to calibrate the parameters.

The automated calibration process is performed as described in Figure (3.8). The input of the METANET model are the initial values of the parameters, before the calibration, chosen among a range of possible values but in casual way, and the initial condition of the network system, or rather the "real" value coming from Paramics of the state variable at

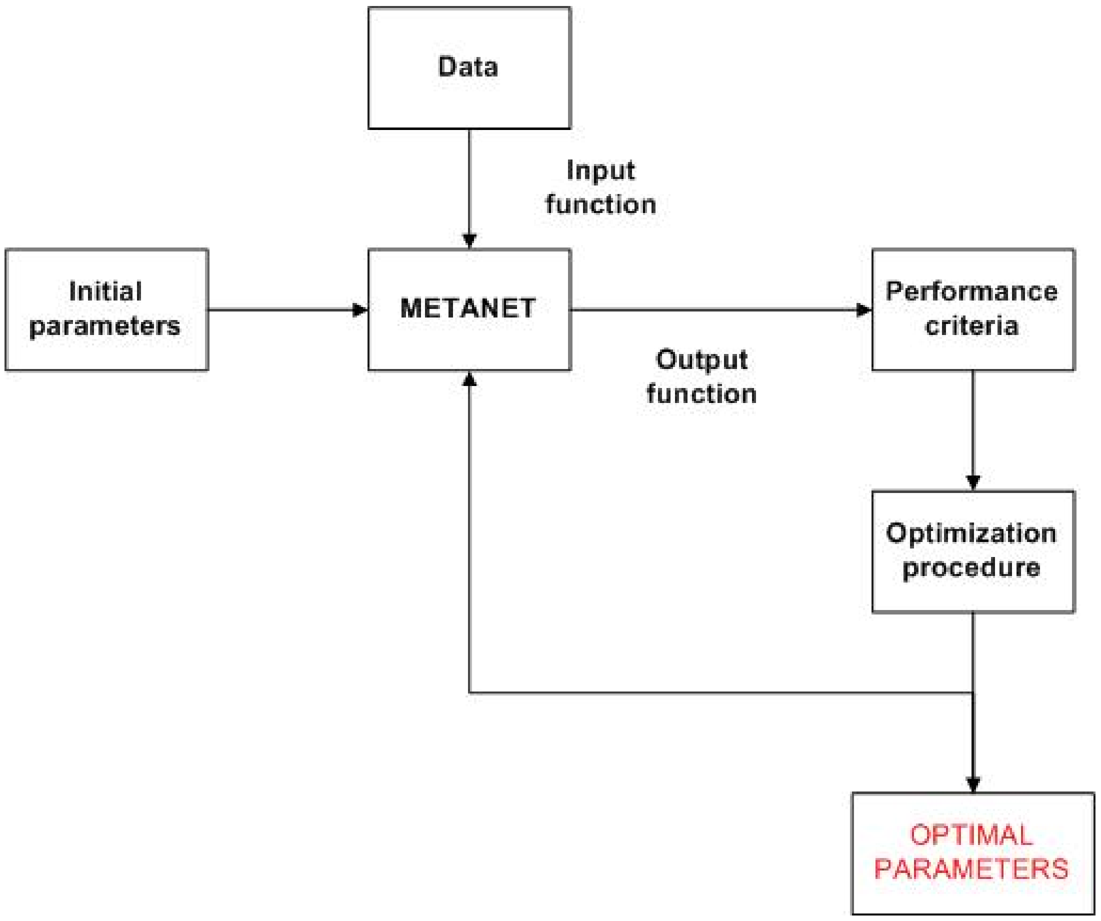


Figure 3.8: Automated calibration procedure

the mainstream origin and on the on-ramps, taken step by step during the whole simulation period. From this condition the METANET model gives the outputs, or rather the values of the state variables for all segments that are confronted time step by time step with the "real" value (performance criteria). They are submitted to the iterative optimization procedure (calibration), that gives for each iteration new values of the parameters until they are optimal and the objective function is minimized.

3.4 Model Predictive Control

When the model that describes the network is perfectly calibrated, we can continue with the control design of the controller. The aim of the controller is to find the control signal that results in an optimal process traffic behavior. In this thesis we will not present this approach in detail. We give only an introduction about it.

In MPC [26], [27], [15], the control is applied in a rolling horizon scheme: at each time instant *k* a new optimization is performed over the prediction horizon $[k, ...; k + N_p - 1]$, and only the first value of the resulting control signal is applied to the process. The next time instant *k+1* this procedure is repeated.To reduce complexity and improve stability often a control horizon N_c $(\leq N_p)$ is introduced, and after the control horizon has been passed the control signal is taken to be constant. So there are two loops: the rolling horizon loop and the optimization loop inside the controller. The loop inside the controller of Figure (3.9) is executed as many times as needed to find the optimal control signals at time instant k, for given N_p, N_c, traffic state and expected demand. The loop connecting the controller and the traffic system is performed once for each k and provides the state feedback to the controller. Recall that this feedback is necessary to correct for (the ever present) prediction errors, and disturbance rejection (compensation for unexpected traffic demand variations). The advantage of this rolling horizon approach is that it results in an on-line adaptive control scheme that allows us to take changes in the system or in the system parameters into account by regularly updating the model of the system. In conventional MPC heuristic tuning rules have been developed to select appropriate values for N_p and N_c. One of the main parameters of MPC is the length of the prediction horizon N_p, the number of samples for which the behavior model is predicted. One should choose N_p long enough to include all relevant system dynamics in the prediction, but a too large N_p unnecessarily increases the computational demand.

The MPC finds the control signal $r_0(j)$ and $v_{control,m,i}(j)$ for $j \in k, ...k + N_c - 1$ that minimize the Total Time Spent (TTS) by the vehicles in the network:

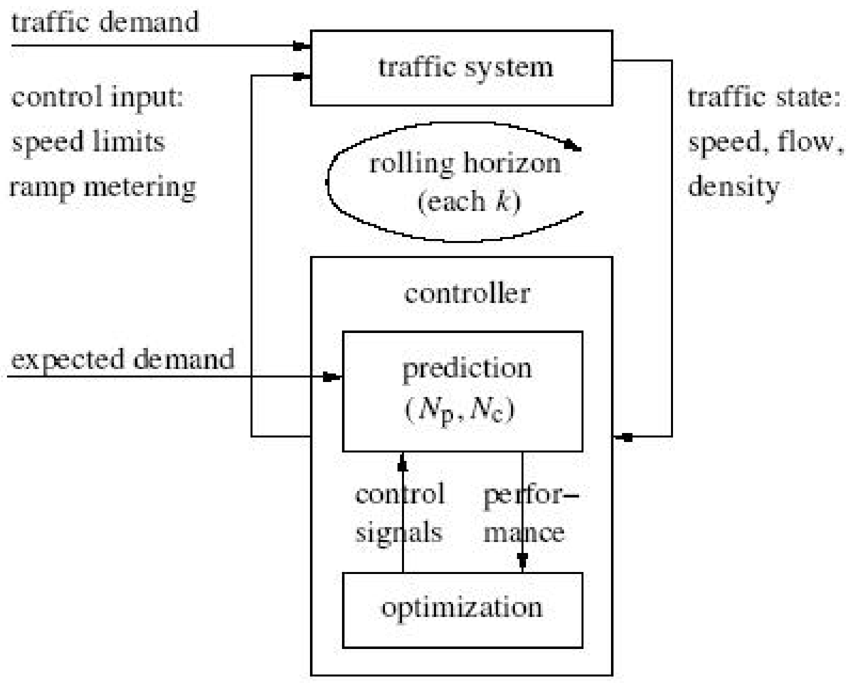


Figure 3.9: Schematic view of the model predictive control structure [15]

$$J(k) = T \sum_{j=k}^{k+N_p-1} \left\{ \sum_{m,i} \rho_{m,i}(j) L_m \lambda_m + \sum_{0} w_0(j) \right\} + \sum_{j=k}^{k+N_c-1} \left\{ a_{ramp} \sum_{o \in O_{ramp}} (r_0(j) \right.$$

$$\left. - r_0(j-1))^2 + a_{speed} \sum_{(m,i) \in I_{speed}} \left(\frac{v_{control,m,i}(j) - v_{control,m,i}(j-1)}{v_{free,m}} \right)^2 \right\} \qquad (3.17)$$

where O_{ramp} is the set of indexes *o* of those on-ramps where ramp metering is present, and I_{speed} is the set of pairs of indexes (m, i) of the links and segments where speed control is present.

Chapter 4

Eindhoven, model implementation

In this chapter we describe the Matlab implementation of the freeway network model of Eindhoven. In the first section we will illustrate the Matlab function that represents the model and the problems that occurred during the creation. In the second section we will discuss the calibration, or rather, the choice of the parameters that characterize the model that approximate the real network. After this, we will talk about Paramics, the simulation program from which we can obtain data, considerable as real, that can be compared with the data coming from our model, and then we will talk about the heart of this thesis, the optimization function "fmincon" and "patternsearch" (Matlab) used to find the optimal parameters that calibrate the model. In the last section we will give the results of the optimization.

4.1 Model implementation in Matlab

In this section we describe the Matlab function that implements the freeway network META NET model. It follows the procedure introduced in the work of Bart De Schutter and Andreas Hegyi [2] described in the Chapter 2 and 3 and it is adapted for the case study network of Eindhoven.

4.1.1 Function "model_eindhoven": syntax

```
[Q_model,RHO_model,V_model]=model_eindhoven(v_free,a,rho_crit,alpha,
tau,eta,kappa,delta,vmin,vmax,rho_max)
```

The input parameters are the model parameters of the METANET model described in the chapter 3 (inside the parenthesis there is the correspondence with METANET model), which we selected for the calibration and that have the following functions:

- `v_free`: (v_{free}) It is the average speed that the drivers assume if the traffic is freely flowing.
- `a`: (a) It is involved in the desired speed $V(\rho_{m,i}(k))$.
- `rho_crit`: (ρ_{crit}) It is the critical density at which the traffic flow is maximal. For densities above this critical density a traffic jam is very probable.
- `alpha`: (α) It is involved to $1 + \alpha$, the no-compliance factor,
- `tau`: (τ) It is the "relaxation time" and it is involved in the anticipation term in the dynamical evolution of mean speed $v_{m,i}(k + 1)$.
- `eta`: (η) It is the "anticipation factor" and it is also involved in the anticipation term in the dynamical evolution of mean speed $v_{m,i}(k + 1)$.
- `kappa`: (κ) Parameter that is also involved in the anticipation term in the dynamical evolution of mean speed $v_{m,i}(k + 1)$.
- `delta`: (δ) This parameter represents the effect of merging phenomena on the speed.
- `rho_max`: Is the maximal density, so the capacity of the freeway.

The following parameters are not described in Chapter 3 because they do not appear in the METANET model, but it was necessary to introduce them in the Matlab implementation of the model and so to calibrate them too.

- `vmin`: It is the minimum speed that we can consider.
- `vmax`: It is the maximal speed that we can consider.

The outputs of the function `model_eindhoven` are:

- `Q_model`: This matrix contains the value of the flow q for each segment and each time step.
- `RHO_model`: This matrix contains the value of the density ρ for each segment and each time step.
- `V_model`: This matrix contains the value of the speed v for each segment and each time step.

The dimensions of each matrix are 19×60, because 19 is the number of the segments of the network considered, and 60 are the number of time steps, in order to have a simulation of 10 minutes ($T = 10s$).

4.1.2 Function "model_eindhoven": program description

The function starts with the initialization of some vectors and parameters. The most important parameter is `T` that defines the duration of the time step (`T=10s`) at 10 seconds. Then we define:

- `L`: This vector contains the length of the segments of each link in km.

  `L=[292 507.5 252 252 458 326 196 330.5 390 256 74 367.5 436]/1000`

- `numsegment`: This vector contains the number of segments of each link.

  `numsegment=[1 2 1 1 2 2 1 2 2 1 1 2 1];.`

- `lambda`: This vector contains the number of lanes of each link.

  `lambda=[3 2 4 4 3 3 4 3 3 4 5 2 2 2]`

- `kmax`: is the maximal number of steps. It is fixed to 60 because we decided to have 10 minutes of simulation.

- `v_control`: This matrix contains for each link and each segment the control speed. In this thesis we can not develop the controller that generates the input `v_control`, but we told about it in the Chapter 3.4, "Model Predictive Control".

- `rho` and `v`: These matrices contain respectively the density and the speed at the current time step for each segment. `rho` is initialized at zero, instead `v` at `v_free`. From them we can calculate the value of the outflow `q` for each segment, and the matrices `rho_pred` and `v_pred` that contain respectively the values of the density and the speed valid for the next step `(k+1)` for each segment. The following equations have a general validity and correspond respectively to the equations 3.3 and 3.4:

  ```
  rho_pred(m,i)=rho(m,i)+(T/(L(m)*lambda(m)))*(q(m,i-1)-q(m,i))
  ```

  ```
  v_pred(m,i)=v(m,i)+(T/tau)*(V-v(m,i))
  +(T/L(m))*v(m,i)*(v(m,i-1)-v(m,i))-
  ((eta*T)/(tau*L(m))*(rho(m,i+1)-rho(m,i))/(rho(m,i)+kappa))
  ```

The other initial conditions are the vehicles coming from other roads into the freeway,or rather the outflow of the mainstream origin and the outflow of the on-ramps. They are "real data" obtained with Paramics, and they are introduced into the function thanks to the function `data_in`. With this function we can introduce the flow, the density and the speed at the beginning of the freeway (`q_0,rho_0,V_0`) and the flow, the density and the speed (`q_ramp,rho_ramp,v_ramp`) coming from the on-ramps for each time step. As we describe in Chapter 3, there is a difference between flow coming from the mainstream origin and flow coming from on-ramps. For the mainstream origin outflow we adopted this solution:

```
q_0(k)=min(q_0(k),q_limit);
```

where:

```
 if v_limit< (v_free*exp(-1/a))

     q_limit=lambda(m)*v_limit*rho_crit*(-a*log(v_limit/v_free))
             ^(1/a);
else
     q_limit=q_cap;
end
```

as described in the equation 3.15.

For the on-ramps we adopted the following solution:

```
q_ramp(k,m)=min(q_ramp(k,m),C*lambda_ramp(m),
C*lambda_ramp(m)*((rho_max-rho(m,1))/(rho_max-rho_crit)));
```

as described in equation 3.7.

These initial conditions are considered when we calculate `rho_pred` and `v_pred` for the first segment of each link. In particular `rho_pred` and `v_pred` become respectively:

```
rho_pred(m,i)=rho(m,i)+(T/(L(m)*lambda(m)))*(q_0(k)-q(m,i))

v_pred(m,i)=v(m,i)+(T/tau)*(V-v(m,i))+(T/L(m))*v(m,i)*(V_0(k)-v(m,i))
-((eta*T)/(tau*L(m)))*((rho(m,i+1)-rho(m,i))/(rho(m,i)+kappa))
```

if we consider the first link and the first segment, or rather if we consider the beginning of the freeway, otherwise they become respectively:

```
rho_pred(m,i)=rho(m,i)+(T/(L(m)*lambda(m)))*(q0(m)-q(m,i))
```

```
v_pred(m,i)=v(m,i)+(T/tau)*(V-v(m,i))+(T/L(m))*v(m,i)*(V0(m)-v(m,i))
-((eta*T)/(tau*L(m)))*((rho(m,i+1)-rho(m,i))/(rho(m,i)+kappa))
-(delta*T*q_ramp(k,m)*v(m,1))/(L(m)*lambda(m)*(rho(m,1)+kappa))
```

where `q0(m)` is calculated as:

```
q0(m+1)=beta*q(m,numsegment(m)) + q_ramp(k,m+1)
```

and `V0(m)` as:

```
V0(m+1)=(v(m,numsegment(m))*q(m,numsegment(m))*beta
+v_ramp(k,m+1)*q_ramp(k,m+1))/(q(m,numsegment(m))*beta
+q_ramp(k,m+1))
```

if we consider the first segment of the other links, because is here that is possible to have an on-ramp. In the last two equations we can see the parameter `beta` that correspond to the turning rate $\beta_{n,m}(k)$ of the METANET model described in Chapter 3, (the fraction of the total flow through node *n* that leaves via link *m*), or rather the ratio between the flow of the vehicles remaining in the freeway and those that leave it and go to the off-ramp. We consider only one value of β valid for all the segments and for all time steps. It is found through the function `data_beta`, that computes a kind of mean value between $\beta_{n,m}(k)$ values using the "real" data coming from Paramics.
Another important step is calculate `rho_pred(m-1,numsegment(m-1)+1)`, that is used to calculate both `rho_pred` and `v_pred`:

```
rho_pred(m-1,numsegment(m-1)+1)=(rho_pred(m,1)+

+rho_offramp(k,m))^2/(rho_pred(m,1)+rho_offramp(k,m))
```

where `rho_offramp` is the density of the off-ramp, and it is calculated from the "real" values of Paramics simulation, with the function `offramp_data`. If `rho_offramp` is bigger than `rho_crit` of the off-ramp, `rho_offramp=rho_crit`

Finally, we can calculate the other step in a recursive way following the same reasoning done until here, in order to have a total simulation of 10 minutes, considering:

```
rho=rho_pred
v=v_pred
```

At the end we will obtain with other passages the output matrices `Q_model`, `RHO_model`, `V_model` already discussed.

4.2 Model calibration and validation

In this section we describe the process of calibration and validation of the network model. The purpose of the calibration is to estimate the optimal global parameters of the model that we have already described in order to obtain outputs that are in accordance with the real data (validation). To do it two important tools are necessary: Paramics, from which we take the "real data" to confront with the data coming from the Matlab model, and a Matlab optimization function in order to find the optimal parameters that can decrease the objective function, or rather the differences between the output of analytical model and the data coming from Paramics.

4.2.1 Simulation with Paramics

In this subsection we talk about the Paramics software. It is a very powerful traffic simulation software of microscopic level and it is able to manage wide nets (up to a million knots, four million of arcs, 32.000 zones O/D and a boundless number of vehicles contemporarily on the net) with a great simplicity and elaboration speed.

While the macro-simulator, on the base of physics laws and of statistics behavior, analyzes and elaborates parameters and average measures, the micro-simulator, on the base of motion laws of the vehicle and of the driver's behavior, analyzes and elaborates instant by instant the movement of every single vehicle on the net. The dynamic micro-simulations tool is able to represent, in a very accurate way, the traffic and its instant evolution, considering the geometric aspects of the infrastructure and the real behavior of the vehicles.

Paramics [29] [30] is a suite of high performance software tools used to model the movement and behavior of individual vehicles on urban and highway road networks. The Paramics Project Suite consists of:

- **Paramics Modeller**: it provides a visualization of road networks and traffic demands using a graphical user interface (GUI). Geographic and travel data is input to the program which then simulates the lane changing, gap acceptance and car following behavior for each vehicle. The speed of the simulation is governed by the computer processing power, the size of the network and the number of vehicles on the network at any one time.

- **Paramics Processor**: it configures and runs the traffic simulation in batch mode without visualization of the network through the GUI. This dramatically increases the speed of simulation and is used to collect simulation results for the numerous test options and sensitivity tests required.

- **Paramics Analyser**: it reads output from the simulation model and provides a GUI to compare post processing simulation results to observed data and to contrast and analyze different test results.

In this thesis we will use only the Paramics Modeller tool.

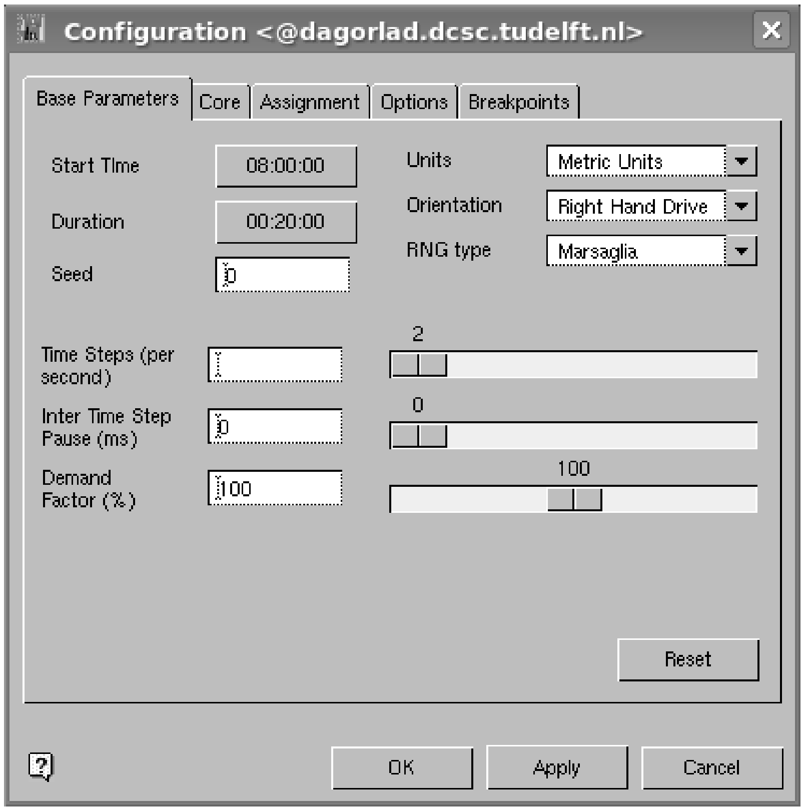
Paramics Modeller

Paramics Modeller [30], requires two main inputs. The first is the road network data; the second is the travel demand data. Road network data consists of geometric layout, junction descriptions, lane markings and turning movement information. Junction or intersection descriptions are stored in the model as "node" data where each junction is allocated a node number or name. The road network which connects between nodes, describes the geometry of the road, the lane specification and the distance. The connection between two nodes is called a "link". Travel demand data can be divided into sub-areas and it is represented by zone-to-zone movements and by an origin/destination matrix of trips. Zones within the study area are referred to as internal zones while zones outside the study area are referred to as external zones. The traffic assignment process allocates the journeys (or trips) to appropriate routes through the network. Alternative routes are calculated depending on perception of link costs, on network congestion and on network restrictions such as banned turns. To ensure that the model reflects as accurately as possible the existing road conditions, a "base year" model is usually constructed.

In the next chapter we use Modeller Paramics to analyze the case study network of Eindhoven in order to start with the calibration.

4.2.2 Simulation of Eindhoven network

The first step before to starting the simulation is to define the location and position of the "loop detectors" in the network that it was pre-built and stored. The loop detector is a very powerful tool. The output data of the simulation, in fact, can be seen where loop detectors are located. We decided to put one loop detector in each segment, in order to know segment by segment the traffic evolution. The detectors can show for each segment:


Configuration <@dagorlad.dcsc.tudelft.nl>

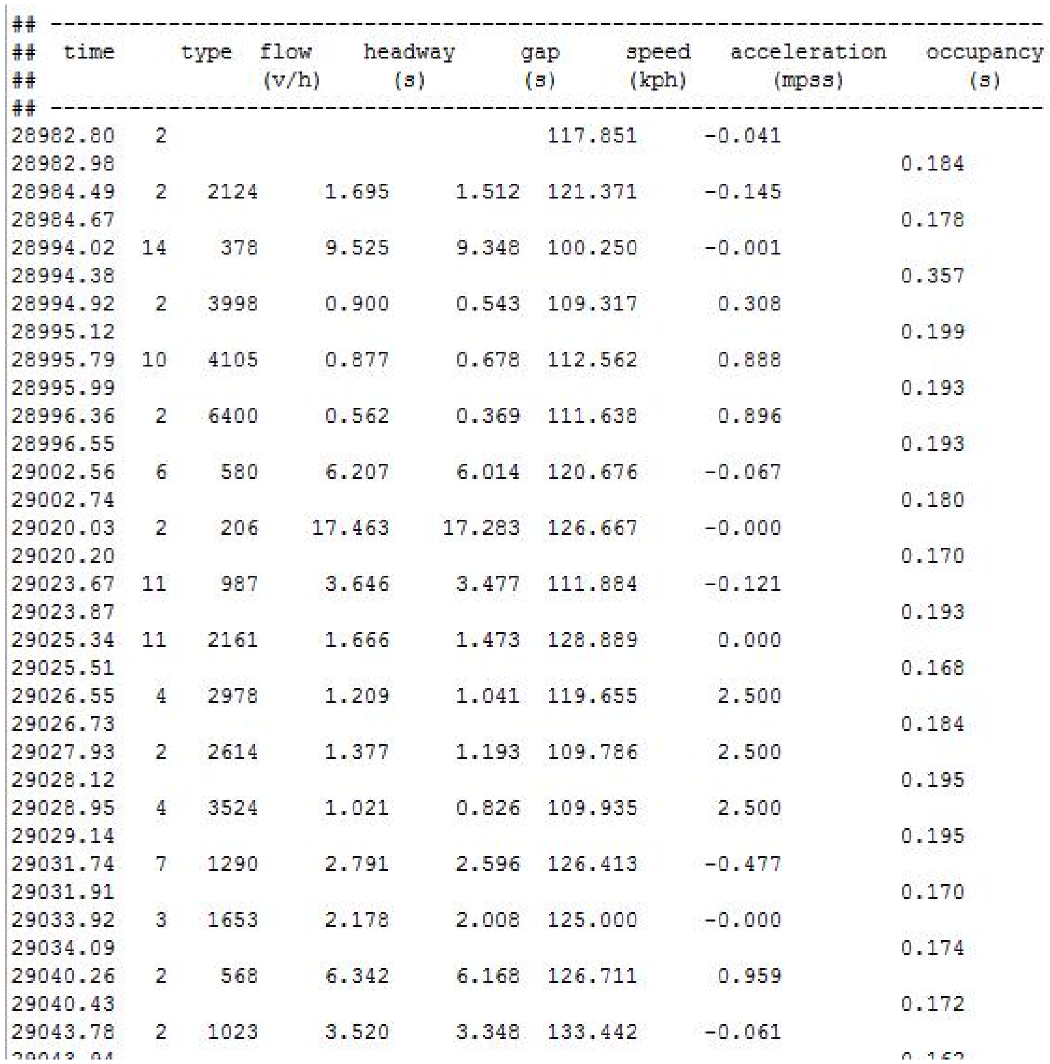
Start Time 08:00:00
Duration 00:20:00
Seed 0
Units Metric Units
Orientation Right Hand Drive
RNG type Marsaglia
Time Steps (per second) 2
Inter Time Step Pause (ms) 0
Demand Factor (%) 100

Figure 4.1: Configuration panel of Modeller Paramics tool.

- Occupancy: the time that the loop is covered by vehicles.
- Gap: the time that the loop is free.
- Headway: the time between leading edges of successive vehicles
- Flow: the instantaneous flow calculated from inverting the headway.
- Speed: it is calculated from the time difference between two rising edges
- type: the type of vehicle is identified by a number.

The second step is to configure the parameters of the simulation as the time of simulation, the duration, the units and orientation.

As the Figure (4.1) shows , we start the simulation at 8.00 a.m. with a duration of 20 minute, although we used only the first 10 minutes, we use Metric units, right-hand driving and a demand factor (use of the network) of 100%. The output data of the simulation coming from detectors give a lot of information. There is one detector for each segment, so we can have information about the state variables for each segment in each time step. We give an example in Figure (4.2) of the 7th detector (the detector place in the 7th segment):



```
## ---------------------------------------------------------------------------
##  time      type  flow    headway     gap     speed   acceleration   occupancy
##                  (v/h)     (s)       (s)     (kph)      (mpss)          (s)
## ---------------------------------------------------------------------------
28982.80   2                                 117.851     -0.041
28982.98                                                                0.184
28984.49   2   2124     1.695      1.512  121.371     -0.145
28984.67                                                                0.178
28994.02  14    378     9.525      9.348  100.250     -0.001
28994.38                                                                0.357
28994.92   2   3998     0.900      0.543  109.317      0.308
28995.12                                                                0.199
28995.79  10   4105     0.877      0.678  112.562      0.888
28995.99                                                                0.193
28996.36   2   6400     0.562      0.369  111.638      0.896
28996.55                                                                0.193
29002.56   6    580     6.207      6.014  120.676     -0.067
29002.74                                                                0.180
29020.03   2    206    17.463     17.283  126.667     -0.000
29020.20                                                                0.170
29023.67  11    987     3.646      3.477  111.884     -0.121
29023.87                                                                0.193
29025.34  11   2161     1.666      1.473  128.889      0.000
29025.51                                                                0.168
29026.55   4   2978     1.209      1.041  119.655      2.500
29026.73                                                                0.184
29027.93   2   2614     1.377      1.193  109.786      2.500
29028.12                                                                0.195
29028.95   4   3524     1.021      0.826  109.935      2.500
29029.14                                                                0.195
29031.74   7   1290     2.791      2.596  126.413     -0.477
29031.91                                                                0.170
29033.92   3   1653     2.178      2.008  125.000     -0.000
29034.09                                                                0.174
29040.26   2    568     6.342      6.168  126.711      0.959
29040.43                                                                0.172
29043.78   2   1023     3.520      3.348  133.442     -0.061
29043.94                                                                0.163
```

Figure 4.2: Detector outputs: detector in the 7th segment

Then these output data are handled to be used in a specific way.

Output data handling: data_sim function

The data coming from Paramics in the file `pointxxx`, are not ready to be used because as we can see in the Figure (4.2), it is quoted only the instant when the vehicle arrives and when it leaves the segment where the detector is located. Instead, the data we need must be reported in veichles/h , but calculated for each time step of 10 seconds. So, we calculate how many vehicles pass through the detectors in 10 seconds, then we translate this result in vehicles/h. The function, built for this objective, is called `data_sim` and it has this interface:

```
[Q_sim,RHO_sim,V_sim]=data_sim()
```

The output `Q_sim`, `RHO_sim`, `V_sim` are respectively the matrices containing the "real" traffic value of Paramics simulation of the flow, the density and the speed for all the segments and all the time steps, so they are $19x60$ dimension matrices, and they are ready to be confronted with the matrices `Q_model`, `RHO_model`, `V_model` coming from the analytical model.

The core of the function are in the following equations:

- `q_in=(3600*num_veich)/10`

  that calculates the number of vehicles that passes through the detector in 10 seconds, but reported in vehicles/h.

- `VV=sum(V)/num_veich`

  where `V` is the speed of each vehicle that passes through the detector in 10 seconds, and so `VV` is the average speed of all of them.

The vectors `q_in` and `VV` contain at the end the values of the flow and speed for each lane of each segment and they are stored respectively in the matrices `Q_in` and `VVV`. So, from these we can calculate `Q_sim`, `V_sim` and `RHO_sim`, that are the final matrices containing the flow, the speed and the density of each segment and each time steps.

The same principles are used in the functions `data_in` and `data_beta` to know the initial conditions.

4.2.3 Calibration of model parameters

In this section we describe the calibration method necessary to have a very realistic model. We describe the two main optimization function, "patternsearch" and "fmincon", justifying why it is better use the first, whose purpose is to minimize the objective function, or rather the difference between the data coming from our model, depending from the parameters to calibrate, and the "real" data coming from Paramics.

Objective function:sintax

The objective function is the function that calculates the difference between the network data coming from Paramics (flow `Q_sim`, density `RHO_sim`, speed `V_sim`), and the data coming from our model (flow `Q_model`, density `RHO_model`, speed `V_model`).

In particular we want to represent, with a Matlab function, the equation 3.16. With the matrices found it becomes:

$$\sum_{k=1}^{k_{max}}\sum_{i=1}^{N_{tot}}\left(\frac{Q\_model - Q\_sim}{Q\_average}\right)^2 + \sum_{k=1}^{k_{max}}\sum_{i=1}^{N_{tot}}\left(\frac{V\_model - V\_sim}{V\_average}\right)^2 \tag{4.1}$$

where k_{max} is the total number of simulation time steps (60), and N_{tot} is the total number of the segments of the network (19). We divided by the average to have compatible measures to add.

The interface of the objective function in Matlab is:

```
[TotError]=obj_func(x)
```

where the input is the vector `x` that contains the value of the parameters to calibrate:

```
x=[v_free,a,rho_crit,alpha,tau,eta,kappa,delta,vmin,vmax,rho_max].
```

The output instead is the total error calculated in the (4.1) and in the function obtained as:

```
TotError=Q_error+V_error
```

where `Q_error` (`V_error`) is the sum of all the elements of the matrix that represent the difference between the flow (the speed) of the network model and the flow (speed) of Paramics data for each segment and each time step (dimension:$19x60$). For this reason we recall inside the function, which load the given values for the parameters and runs the model:

- `[Q_model,RHO_model,V_model]=model_eindhoven(v_free,a,rho_crit, alpha,tau,eta,kappa,delta,vmin,vmax,rho_max)`
- `[Q_sim,RHO_sim,V_sim]=data_sim()`

4.2.4 Optimization problem

The purpose of the calibration is to find the value of the parameters in order to minimize the output of the objective function. In analytical terms, we have to find the global minimum of the objective function. But finding the global minimum is in general difficult for non-linear function, because most optimization methods are designed to find a local minimum, which may or may not be a global minimum. In the next section we talk about `fmincon`, a Matlab optimization function that uses a local gradient-based optimization method called "sequential quadratic programming" (SQP), and then in the second subsection we talk about `patternsearch` that uses another approach.

Optimization tool: fmincon

The optimization function `fmincon` find minimum of constrained nonlinear multivariable function. It is a gradient-based optimization method.

The `fmincon` function, implemented in the MATLAB Optimization Toolbox, attempts to find a constrained minimum of a scalar function of several variables respecting linear and non linear constrains. This function allows to find the minimum through an iterative algorithm that progressively leads to a convergence value.

The `fmincon` function is a gradient-based optimization method, or rather it uses numerical algorithm that try to find the optimum following the gradient direction. But traditional gradient-based techniques work well for well-defined problems and sometimes is not possible to reach the convergence and find a solution. These cases of non convergence can be due by different causes, by a not completely defined problem and by the unlucky choice of start point. In fact, for instance, the function `fmincon` could have come upon local minimum that cannot consent to the algorithm to reach the convergence and find global minimum. In these cases we can try to find another start point, but this could be not solving the problem. In the Figure 4.3 we give an example to better understand.

In the general function in the Figure (4.3) if the starting point of `fmincon` is point A, it can find the absolute minimum, but if it starts from B, it can reach only a relative minimum point.

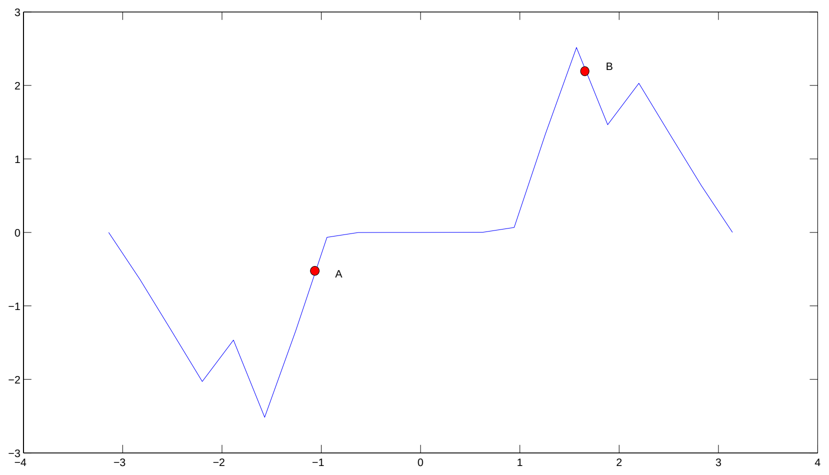


Figure 4.3: if the starting point of fmincon is point A, it can find the absolute minimum, but if it starts from B, it can reach only a local minimum point

Fmincon uses two kinds of optimization method:

- Large-scale method: By default `fmincon` will choose the large-scale algorithm. This algorithm is a subspace trust region method and is based on the interior-reflective Newton method method described in [31], [32]. Each iteration involves the approximate solution of a large linear system using the method of preconditioned conjugate gradients (PCG)
- Medium-scale method: `fmincon` uses a Sequential Quadratic Programming (SQP) method. In this method, a Quadratic Programming (QP) subproblem is solved at each iteration [33].

We give the interface of `fmincon` as used in our program:

```
[x,z,exitflag,output]=fmincon(@(x)obj_func(x),x0,[],[],[],[],lb,
ub,[],options)
```

The inputs are:

- `[x1]=fmincon(@(x)obj_func(x),x0]`: The function `fmincon` start at `x0` (starting point) and finds a minimum `x1` (the output) to the function `obj_func` that is our objective function. `obj_func` accepts input `x0` and returns a scalar function value `f(x)` evaluated at `x`, that in each iteration change, to become at the end `x1` when the optimum is reached. `x0` may be a scalar, vector, or matrix, but in our case is the following vector:

```
x=[v_free,a,rho_crit,alpha,tau,eta,kappa,delta,vmin,
vmax,rho_max]
```

where `x0` are the numerical values associates to `x`.

- `[x1]=fmincon(@(x)obj_func(x),x0,[],[],[],[])`: The four empty matrices should be respectively:

  A: matrix for inequality constraints

  B: vector for inequality constraints

  Aeq: matrix for equality constraints

  Beq: B vector for equality constraints

  A, B, Aeq, Beq should minimize `obj_func` if it was subject to the linear equalities or inequalities constraints $Aeq * x = Beq$ as well as $A * x <= B$. In our case we set A=[], B=[], Aeq=[], Beq=[] because no inequalities or equalities exist.

- `[x1]=fmincon(@(x)obj_func(x),x0,[],[],[],[],lb,ub)`: Defines a set of lower and upper bounds on the design variables `x`, so that a solution is found in the range $lb <= x <= ub$. We define `lb` and `ub` as follows:
  `lb=[100,0,10,0,2,30,20,0,0,130,40]`
  `ub=[130,2,50,1,25,80,70,1,10,160,70]`

- `[x1]=fmincon(@(x)obj_func(x),x0,[],[],[],[],lb,ub,[])`:
  The last empty matrix is NONLCON, that is a nonlinear constraint function. In our case there is no constrain function.

- `[x1]=fmincon(@(x)obj_func(x),x0,[],[],[],[],lb,ub,[],options)`: It minimizes with the default optimization parameters replaced by values in the structure `options`, an argument created with the `optimset` function. We set `optimset` as follows:

  ```
  options=optimset('Display','Iter','DiffMinChange',0.1,
  'DiffMaxChange',1,'MaxFunEvals',30000,'MaxIter',200)
  ```

  Where:

  – Display: Level of display. We set to Iter, in order to display all the iteration of the optimization.

- DiffMaxChange: Maximum change in variables for finite-difference gradients. It is use only in medium-scale method. We set to 0.1.
- DiffMinChange: Minimum change in variables for finite-difference gradients. It is use only in medium-scale method. We set to 1. The so large interval is due to horizontal part in cost function.
- MaxFunEvals - Maximum number of function evaluations allowed. We set it at 30000 to permit a good evaluation of the minimum problem.
- MaxIter - Maximum number of iterations allowed. We set it to 200.

The output are:

- `[x1,z]=fmincon(@(x)obj_func(x),x0)` returns `z`, the value of

  the objective function `obj_func` at the solution `x1`.

- `[x1,z,exitflag]=fmincon(@(x)obj_func(x),x0)` returns an `exitflag` that describes the exit condition of `fmincon`. Possible values of `exitflag` and the corresponding exit conditions are listed below:

  Both medium- and large-scale:

  - 1 First order optimality conditions satisfied to the specified tolerance.
  - 0 Maximum number of function evaluations or iterations reached.
  - -1 Optimization terminated by the output function. Large-scale only:
  - 2 Change in `x` less than the specified tolerance.
  - 3 Change in the objective function value less than the specified tolerance. Medium-scale only:
  - 4 Magnitude of search direction smaller than the specified tolerance and constraint violation less than options.TolCon.
  - 5 Magnitude of directional derivative less than the specified tolerance and constraint violation less than options.TolCon.
  - -2 No feasible point found.

- `[x1,z,exitflag,output]=fmincon(@(x)obj_func(x),x0)` returns a structure `output` like this:

  `iterations`: the number of iterations.

  `funcCount`: number of function evaluations.

  `lssteplength`: The medium scale algorithm returns here the final line search step-length.

`cgiterations`:the large scale algorithm returns here the number of `cg` iterations, or rather the number of conjugate gradient iterations take by the current (optimization) iteration.

`stepsize`: returns the norm of the final step.

`algorithm`: returns the algorithm used.

`firstorderopt`: the first-order optimality. Returns the current violation of the optimality condition.

`message`: returns the exit message.

We will show all the output and result of the function `fmincon` in the chapter Results.

Optimization tool: patternsearch

The optimization function `patternsearch` uses a different approach than `fmincon`. In fact it is defined as a "direct search" method. This is a method for solving optimization problems that does not require any information about the gradient of the objective function. The direct search algorithm searches a set of points around the current point, looking for one where the value of the objective function is lower than the value at the current point. Direct search methods can solve a variety of optimization problems that are not well suited for standard optimization algorithms, including problems in which the objective function is discontinuous, non-differentiable, stochastic, or highly nonlinear, or have different local minimum.

A pattern search algorithm computes a sequence of points that get closer to the optimal point.

- At each step, the algorithm searches a set of points, called a mesh, around the current point, the point computed at the previous step of the algorithm (at first the starting point then the point computed at the next step of the algorithm).
- The algorithm forms the mesh by adding the current point to a scalar multiple of a fixed set of vectors called a pattern.
- If the algorithm finds a point in the mesh that improves the objective function at the current point, the new point becomes the current point at the next step of the algorithm.

At each step, the pattern search algorithm searches a set of points, called a mesh, for a point that improves the objective function. The algorithm forms the mesh by:

1. Multiplying the pattern vectors by a scalar, called the mesh size.

2. Adding the resulting vectors to the current point (the point with the best objective function value found at the previous step).

For example, if the current point is $[1.63.4]$ and the mesh size is 4, the algorithm multiplies the pattern vectors by 4 and adds them to the current point to obtain the following mesh:

$[1.63.4] + 4 * [10] = [5.63.4]$
$[1.63.4] + 4 * [01] = [1.67.4]$
$[1.63.4] + 4 * [-10] = [-2.43.4]$
$[1.63.4] + 4 * [0 - 1] = [1.6 - 0.6]$

The pattern vector that produces a mesh point is called its direction. At each step, the algorithm polls the points in the current mesh by computing their objective function values. By default the algorithm stops polling the mesh points as soon as it finds a point whose objective function value is less than that of the current point. The poll is then called successful and that point becomes the current point at the next iteration.

After a successful poll, the algorithm multiplies the current mesh size by 2, the default value of Mesh Expansion factor. Because the initial mesh size is 1, at the second iteration the mesh size is 2. If the algorithm fails to find a point that improves the objective function, the poll is called unsuccessful and the current point stays the same at the next iteration. After an unsuccessful poll, the algorithm multiplies the current mesh size by 0.5, the default value of Mesh Contraction factor. The algorithm then polls with a smaller mesh size.

We give the interface of `patternsearch` as used in our program:

```
[x1,z,exitflag,output]=patternsearch(@(x)obj_func(x),x0,[],
[],[],[],lb,ub,[],options)
```

As we can see, the syntax is equal to `fmincon`. The input have the same meaning and so the same values, but the output have some important differences.

- `[x1,z]=patternsearch(@(x)obj_func(x),x0)` returns z, the value of the objective function `obj_func` at the solution `x1`.

- `[x,z,exitflag]=patternsearch(@(x)obj_func(x),x0)` returns `exitflag` that describes the exit condition of `patternsearch`. Possible values of `exitflag` and the corresponding exit conditions are:

  – 1 Magnitude of mesh size is less than specified tolerance and constraint violation less than options.TolCon.

  - 2 Change in `x` less than the specified tolerance and constraint violation less than options.TolCon.
  - 3 Change in `z` less than the specified tolerance and constraint violation less than options.TolCon.
  - 4 Magnitude of step smaller than machine precision and constraint violation less than options.TolCon.
  - 0 Maximum number of function evaluations or iterations reached.
  - -1 Optimization terminated by the output or plot function.
  - -2 No feasible point found.

- `[x,z,exitflag,output]=patternsearch(@(x)obj_func(x),x0)` returns a structure `output` with the following information:

  function: Objective function

  problemtype: Type of problem (Unconstrained, Bound constrained or linear constrained)

  pollmethod: Polling technique

  searchmethod: Search technique used

  iterations: Total iterations

  funccount: Total function evaluation

  meshsize: Mesh size at `x`

  maxconstraint: Maximum constraint violation

  message: PATTERNSEARCH termination message

We show all the outputs of `patternsearch` in the next chapter, "Results".

Minimization of the objective function

In this subsection we show the differences between the two optimization function, `fmincon` and `patternsearch`, and we explain why we prefer the second above the first. For this purpose we show some graphics that represent the state of the objective function while only one parameter vary and the other stay constant at its average value between the upper bound and the lower bound defined in `fmincon` or `patternsearch`.

- lower bound: `lb=[100,0,10,0,2,30,20,0,0,130,40]`
- upper bound: `ub=[130,2,50,1,25,80,70,1,10,160,70]`

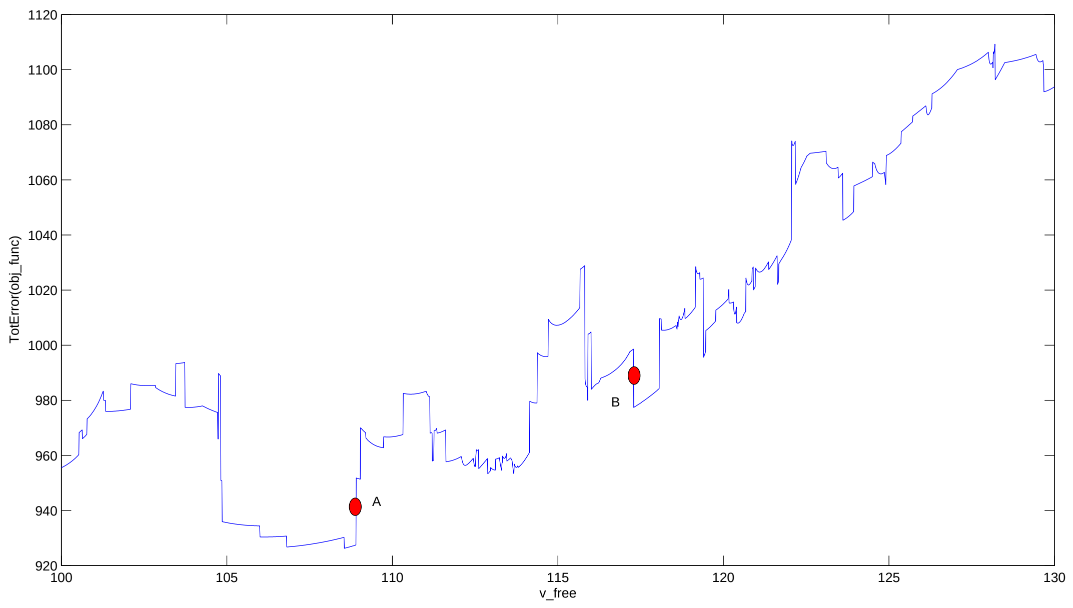


Figure 4.4: v_free

- $x\_average = [115, 1, 30, 0.5, 13.5, 55, 45, 0.5, 5, 145, 55]$

In the Figure (4.4) we represent the state of the objective function (see 4.1) during the variation of `v_free` while the other parameters (`a`,`rho_crit`,`alpha`,`tau`,`eta`,`kappa`, `delta`,`vmin`,`vmax`,`rho_max`) stay constant at the average value. We can see that if the starting point is A, `fmincon` can find the global minimum following the gradient direction, but if it is B, the algorithm finds only a local minimum. So `fmincon` is not useful in this case and it is better to use `patternsearch`, which, as we already explained, does not use the gradient.

In the Figures (4.5) and (4.6) we represent the state of the objective function during the variation respectively of `a` and `rho_crit` while the other parameters stay constant at the average value. Here we can see that any starting point for `fmincon` is right to find the global minimum.

In the Figure (4.7) we represent the state of the objective function varying `rho_max` while the other parameters (`v_free`,`a`,`rho_crit`,`alpha`,`tau`,`eta`,`kappa`,`delta`,`vmin`, `vmax`) stay constant at the average value. Here we can see that if the starting point is A, there is no problem, but if the starting point is B, here we have no gradient, but each possible point is already the minimum, so we have no problem.

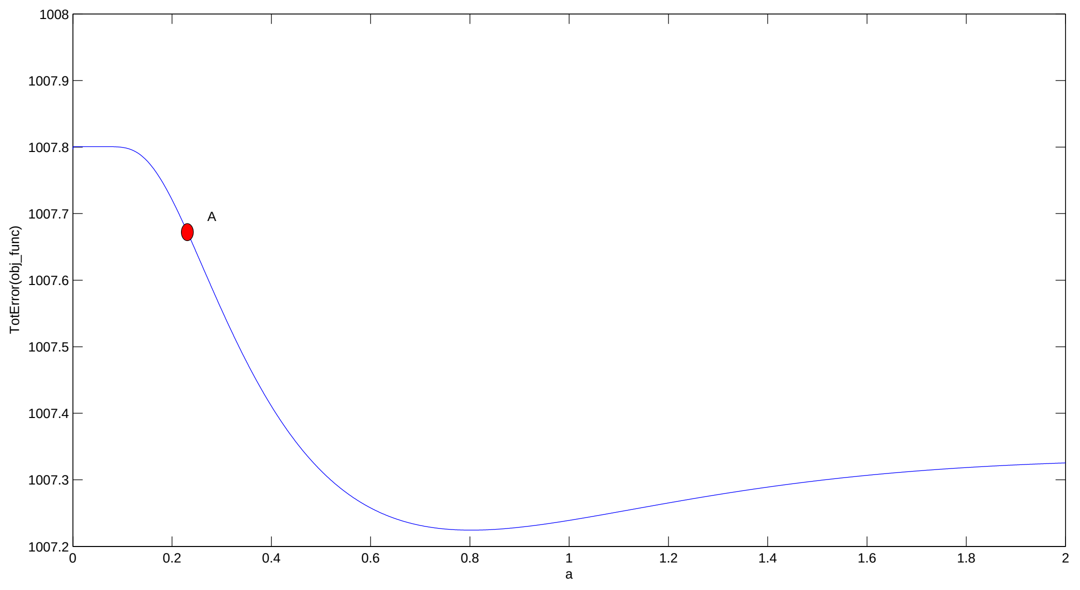


Figure 4.5: a

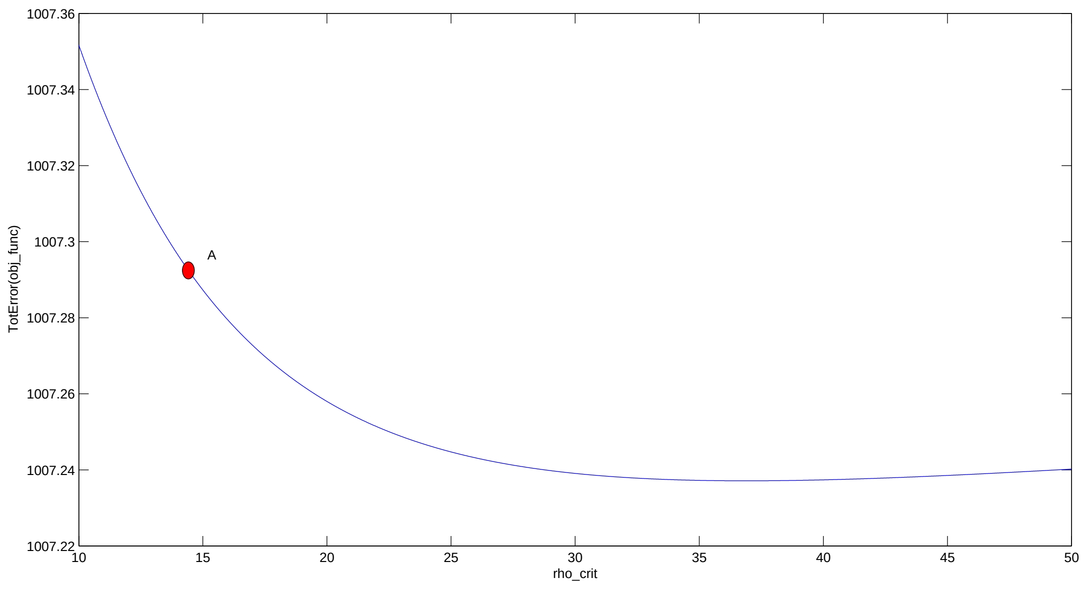


Figure 4.6: rho_crit

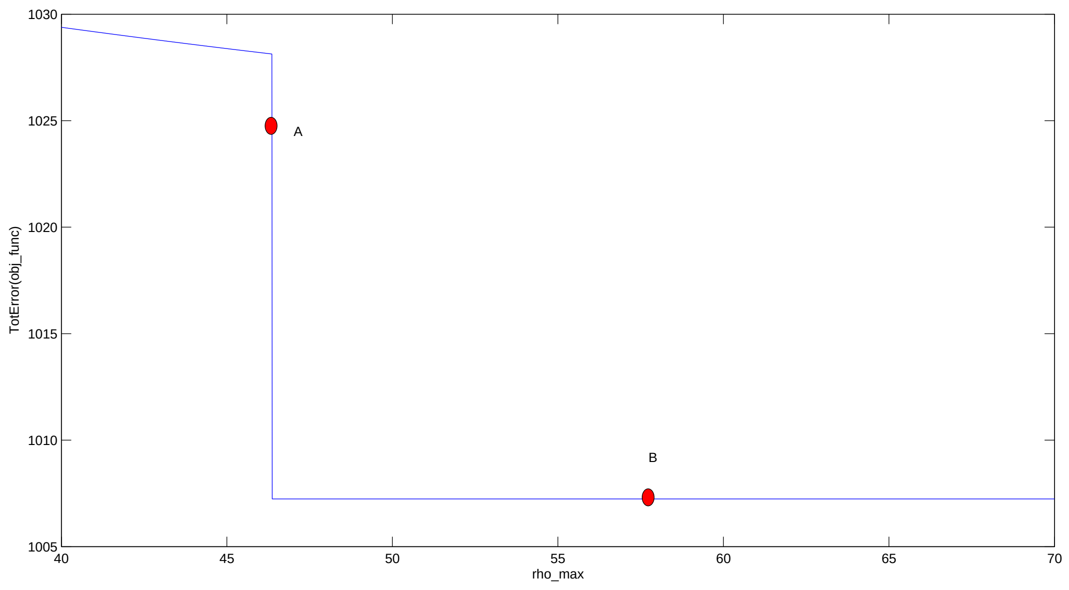


Figure 4.7: rho_max

The same problem is shown in Figure (4.8). Wherever we take point A, we have not gradient, and despite `fmincon` works only with the gradient of the function, here we have no problem, because the point is already the minimum.. In the other Figures (4.9), (4.10),(4.11), (4.11), (4.12), (4.13), (4.14), we show the same problem. The function `fmincon` can find the global minimum only if the choice of the starting point is right. So, for all these cases, is better to use the optimization function `patternsearch`.

At the end we can say, considering the state of the objective function in different cases, that is better to use the optimization function `patternsearch` over `fmincon`.

In the next chapter we will show the results of the optimization with both methods, and we will see that with `fmincon` we are not able to reach the convergence and so the optimal parameters. Instead, with `patternsearch` this is possible.

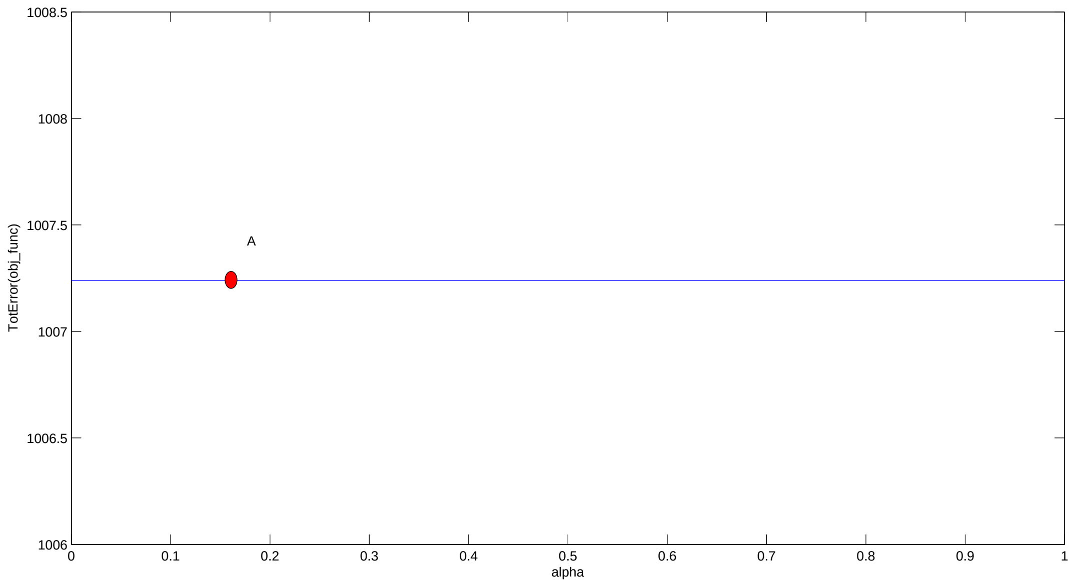


Figure 4.8: alpha

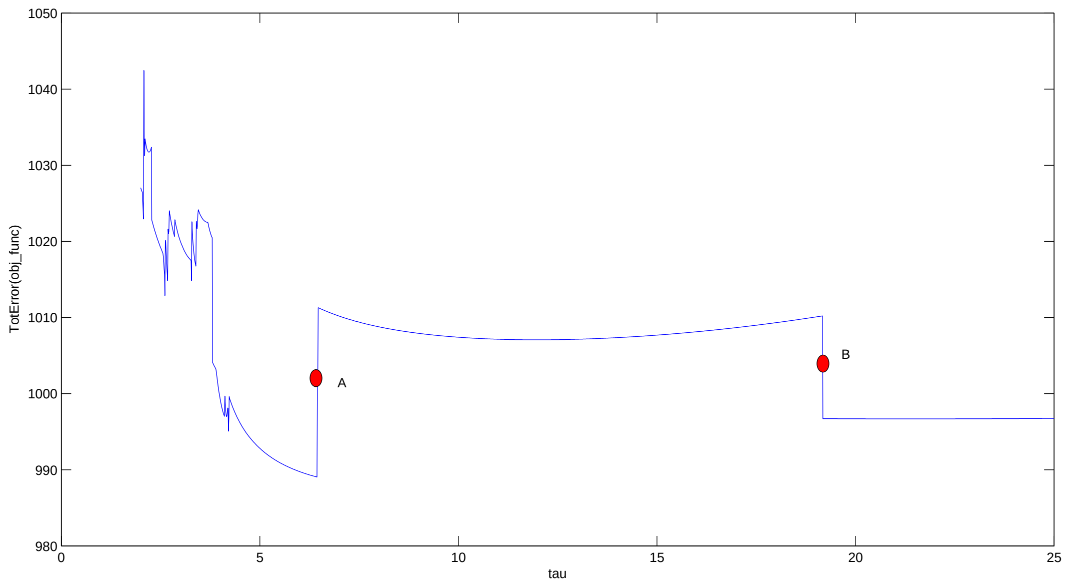


Figure 4.9: tau

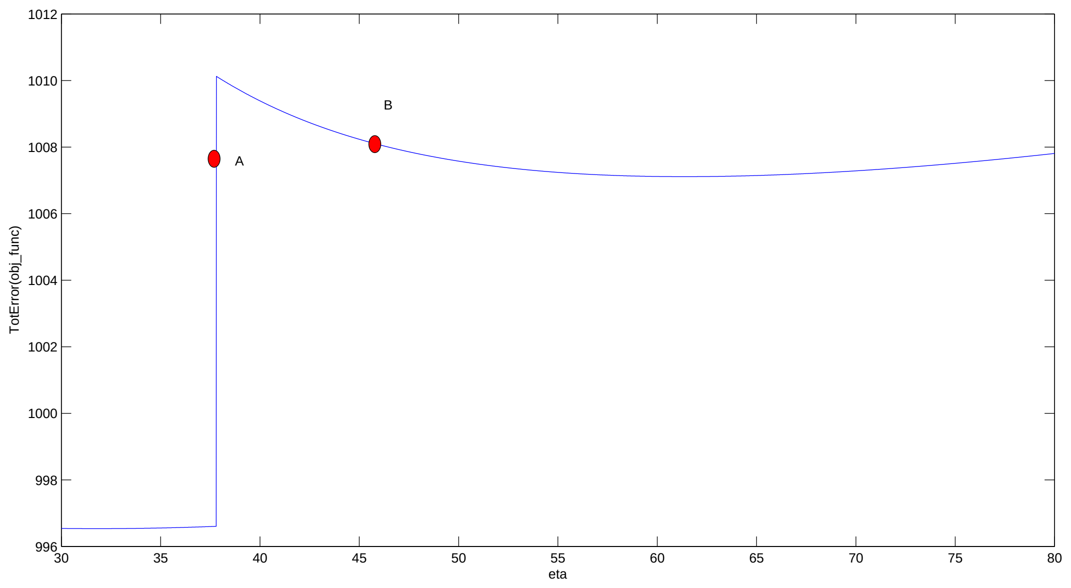


Figure 4.10: eta

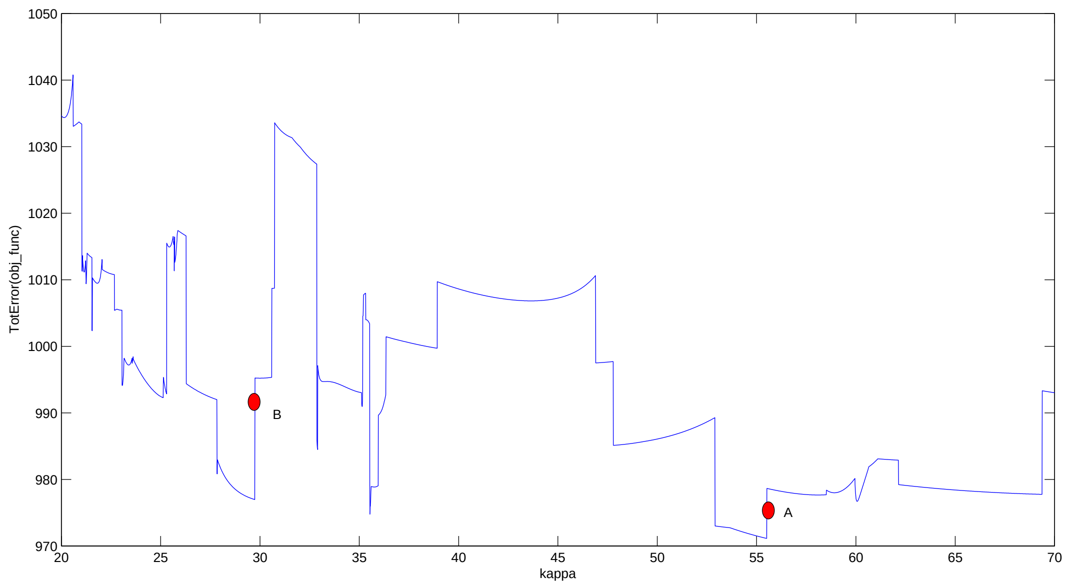


Figure 4.11: kappa

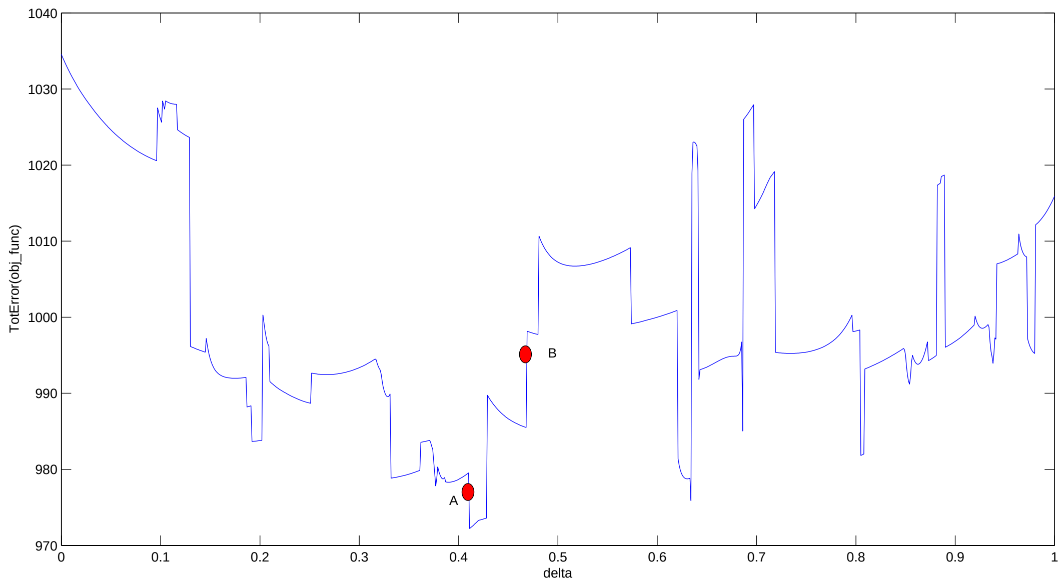


Figure 4.12: delta

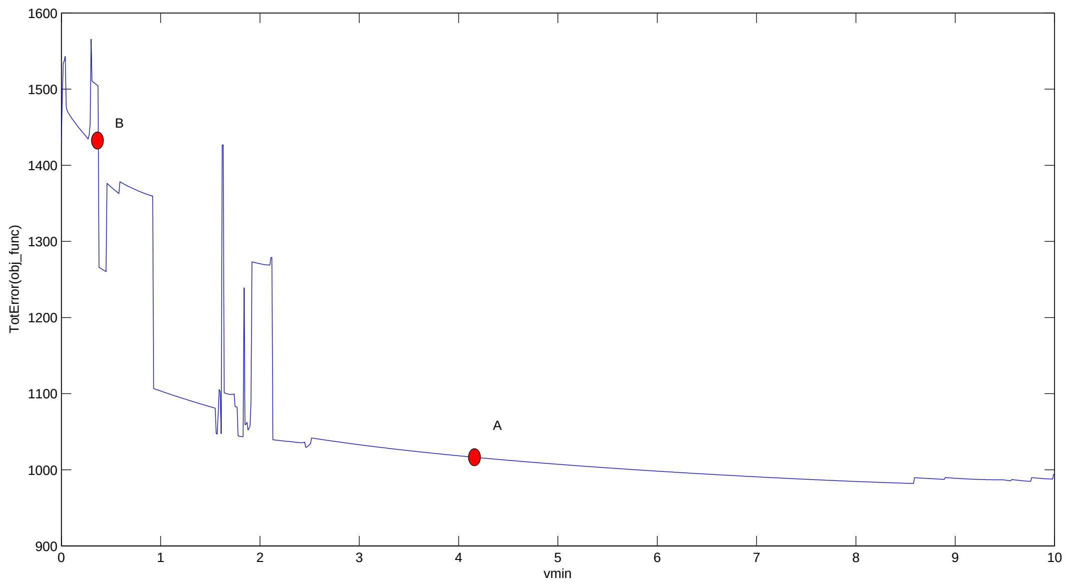


Figure 4.13: vmin

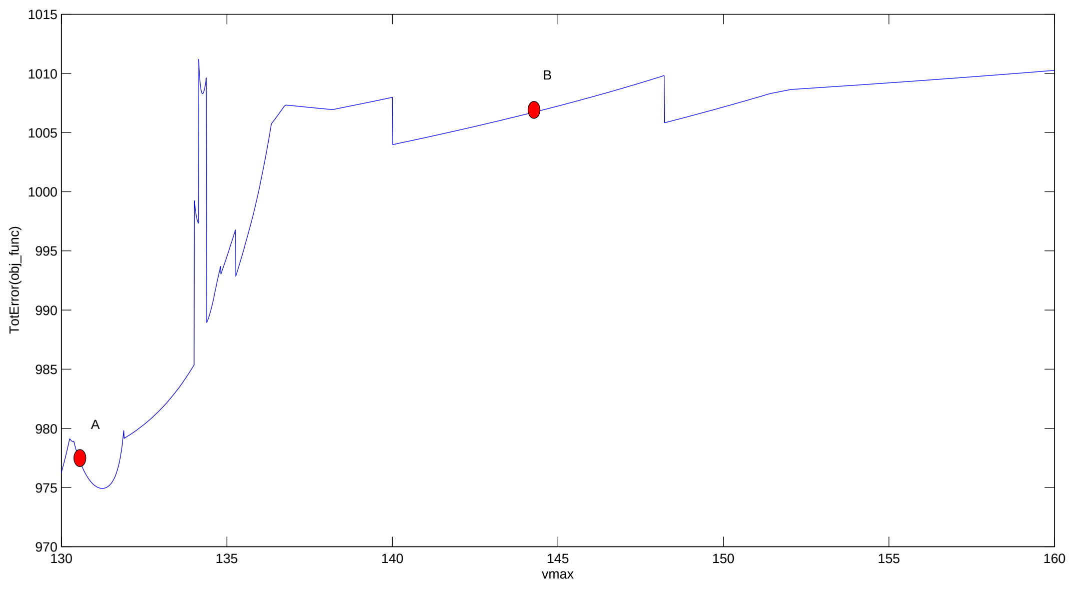


Figure 4.14: vmax

Chapter 5

Results

In this chapter we will show the results of the optimization with both methods, `fmincon` and `patternsearch`, and we will highlight to the faults of `fmincon` and the success of `patternsearch` in our case of study. Next, we will show the results of the calibrated model.

5.1 Optimization results with "fmincon" function

We described in the subsection of Chapter 4, Minimization of objective function, the difficulties that `fmincon` has to find the global minimum in the objective function if it is not differentiable or has many local minimum. During the calibration process all the parameters vary between upper and lower bound and so we can imagine that the state of objective function can present these characteristics. We show the outputs of the `fmincon` optimization with the following starting point:

```
x0=[105,1,35,0.5,13.5,55,45,0.5,5,145,55];
```

```
>> fminconn
Warning: Large-scale (trust region) method does not currently solve
this type of problem,
 using medium-scale (line search) instead.
> In fmincon at 317
  In fminconn at 20

                                Max     Line search  Directional  First-order
 Iter F-count        f(x)   constraint   steplength   derivative   optimality Procedure
    0     12      947.789         -0.5
    1     38      947.788         -0.5    -6.1e-005        -73.4         42.3
    2     53      927.096      -0.4709        0.125          152          361
    3     68      926.466      -0.4961         0.125     1.1e+003          331
    4     94      926.465      -0.4961    -6.1e-005         -334          236
    5    120      926.465       -0.496    -6.1e-005         -623          375  Hessian modified twice
    6    137      926.449      -0.4882        0.0313         -189          486  Hessian modified twice
    7    163      926.448      -0.4882    -6.1e-005         -162          130
    8    189      926.447      -0.4883    -6.1e-005         -162          156  Hessian modified twice
    9    215      926.446      -0.4883    -6.1e-005         -161          158  Hessian modified twice
   10    241      926.445      -0.4883    -6.1e-005         -161          159  Hessian modified twice
   11    267      926.444      -0.4884    -6.1e-005         -161          161  Hessian modified twice
   12    293      926.442      -0.4884    -6.1e-005         -160          146  Hessian modified twice
   13    319      926.441      -0.4884    -6.1e-005         -159          151  Hessian modified twice
   14    345       926.44      -0.4884    -6.1e-005         -161          147  Hessian modified twice
   15    371      926.438      -0.4885    -6.1e-005         -160          148  Hessian modified twice
   16    397      926.437      -0.4885    -6.1e-005         -160          160  Hessian modified twice
   17    423      926.436      -0.4885    -6.1e-005         -161          151  Hessian modified twice
   18    449      926.435      -0.4886    -6.1e-005         -163          161  Hessian modified twice
   19    475      926.434      -0.4886    -6.1e-005         -165          143  Hessian modified twice
   20    501      926.433      -0.4886    -6.1e-005         -163          161  Hessian modified twice
  [...]


  180   4661      926.278      -0.4934    -6.1e-005         -167          155  Hessian modified twice
  181   4687      926.277      -0.4935    -6.1e-005         -167          155  Hessian modified twice
  182   4713      926.276      -0.4935    -6.1e-005         -167          155  Hessian modified twice
  183   4739      926.275      -0.4935    -6.1e-005         -167          155  Hessian modified twice
  184   4765      926.274      -0.4935    -6.1e-005         -167          155  Hessian modified twice
  185   4791      926.274      -0.4936    -6.1e-005         -167          155  Hessian modified twice
  186   4817      926.273      -0.4936    -6.1e-005         -167          155  Hessian modified twice
  187   4843      926.272      -0.4936    -6.1e-005         -167          155  Hessian modified twice
  188   4869      926.271      -0.4937    -6.1e-005         -167          155  Hessian modified twice
  189   4895       926.27      -0.4937    -6.1e-005         -167          155  Hessian modified twice
  190   4921      926.269      -0.4937    -6.1e-005         -167          154  Hessian modified twice
  191   4947      926.268      -0.4938    -6.1e-005         -168          154  Hessian modified twice
  192   4973      926.267      -0.4938    -6.1e-005         -168          154  Hessian modified twice
  193   4999      926.266      -0.4938    -6.1e-005         -168          154  Hessian modified twice
  194   5025      926.266      -0.4938    -6.1e-005         -168          154  Hessian modified twice
  195   5051      926.265      -0.4939    -6.1e-005         -168          154  Hessian modified twice
  196   5077      926.264      -0.4939    -6.1e-005         -168          154  Hessian modified twice
  197   5103      926.263      -0.4939    -6.1e-005         -168          154  Hessian modified twice
  198   5129      926.262       -0.494    -6.1e-005         -168          154  Hessian modified twice
  199   5155      926.261       -0.494    -6.1e-005         -168          154  Hessian modified twice
  200   5181       926.26       -0.494    -6.1e-005         -168          154  Hessian modified twice
Maximum number of iterations exceeded;
 increase OPTIONS.MaxIter.

x1 =

  Columns 1 through 9

  108.8146    1.0142   35.7455    0.5000   13.3370   55.0223   42.5363    0.4940    5.9557

  Columns 10 through 11

  144.9955   55.0000


z =

  926.2611


exitflag =
     0


output =

       iterations: 200
        funcCount: 5181
     lssteplength: -6.1035e-005
         stepsize: 9.2490e-004
        algorithm: 'medium-scale: SQP, Quasi-Newton, line-search'
    firstorderopt: 154.0397
          message: [1x65 char]
```

We can see in the "iteration outputs" that medium-scale method is used. The outputs in the list are:

- Iter: number of iteration.
- F-count: Number of evaluation for each iteration.
- f(x): value of the objective function calculated with the parameters to calibrate that change in each iteration.
- Max constraint: whether the solution violates the constraints
- Line search steplength: Using the medium-scale method, we have to define in the option the maximal and the minimum step-length. During calibration the procedure can choose the best step-length for the optimization.
- Directional derivative: It shows the derivative direction.
- Procedure: The procedure used to find the global minimum.

We can see in the final output that `exitflag` is equal to zero. It means that despite the maximum number of iterations is reached there is not convergence and so the optimization function is not able to find the optimum. We can try to increase the number of iterations, but the result will be the same if we continue to use the same starting point, because as we can see in the results in the column `f(x)`, or in the Figure (5.1), the value of the objective function saturates and does not change, and it means that the function had reached a local minimum.

To prove this, we try, as shown in the following list of iterations, to start with another optimization session where we use the results of the first optimization run as initial value for the second run:

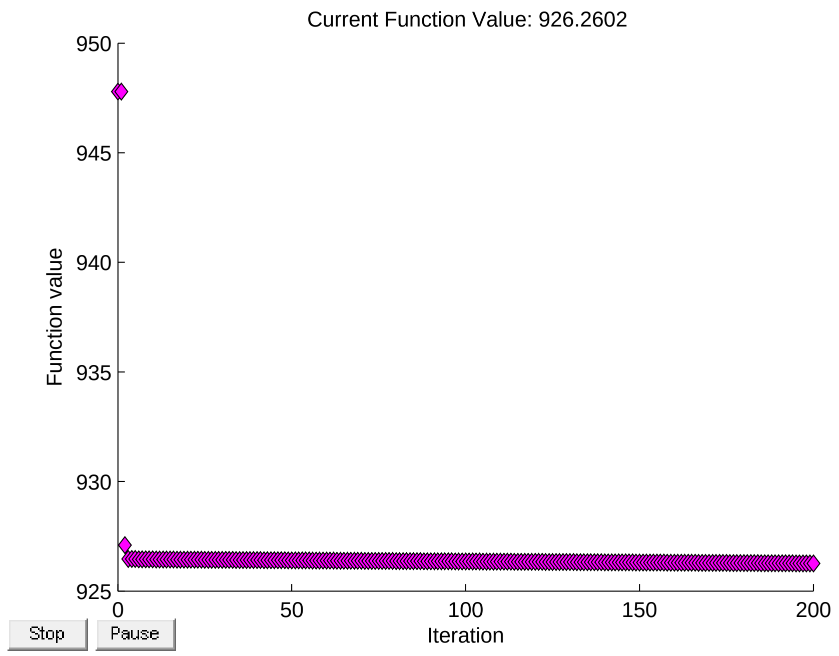


Figure 5.1: Outputs of the fmincon optimization with the following starting point x0=[105,1,35,0.5,13.5,55,45,0.5,5,145,55]

```
>> fminconn
Warning: Large-scale (trust region) method does not currently solve
this type of problem,
 using medium-scale (line search) instead.
> In fmincon at 317
  In fminconn at 21

                                  Max     Line search  Directional  First-order
 Iter F-count        f(x)   constraint   steplength   derivative   optimality Procedure
    0     12      947.789         -0.5
    1     38      947.788         -0.5    -6.1e-005        -73.4         42.3
    2     53      927.096      -0.4709        0.125          152          361
    3     68      926.466      -0.4961        0.125     1.1e+003          331
    4     94      926.465      -0.4961    -6.1e-005         -334          236
    5    120      926.465       -0.496    -6.1e-005         -623          375  Hessian modified twice
    6    137      926.449      -0.4882       0.0313         -189          486  Hessian modified twice
    7    163      926.448      -0.4882    -6.1e-005         -162          130
    8    189      926.447      -0.4883    -6.1e-005         -162          156  Hessian modified twice
    9    215      926.446      -0.4883    -6.1e-005         -161          158  Hessian modified twice
   10    241      926.445      -0.4883    -6.1e-005         -161          159  Hessian modified twice
   11    267      926.444      -0.4884    -6.1e-005         -161          161  Hessian modified twice
   12    293      926.442      -0.4884    -6.1e-005         -160          146  Hessian modified twice
   13    319      926.441      -0.4884    -6.1e-005         -159          151  Hessian modified twice
   14    345       926.44      -0.4884    -6.1e-005         -161          147  Hessian modified twice
   15    371      926.438      -0.4885    -6.1e-005         -160          148  Hessian modified twice
   16    397      926.437      -0.4885    -6.1e-005         -160          160  Hessian modified twice
   17    423      926.436      -0.4885    -6.1e-005         -161          151  Hessian modified twice
   18    449      926.435      -0.4886    -6.1e-005         -163          161  Hessian modified twice
   19    475      926.434      -0.4886    -6.1e-005         -165          143  Hessian modified twice
   20    501      926.433      -0.4886    -6.1e-005         -163          161  Hessian modified twice
  [...]


  180   4661      926.278      -0.4934    -6.1e-005         -167          155  Hessian modified twice
  181   4687      926.277      -0.4935    -6.1e-005         -167          155  Hessian modified twice
  182   4713      926.276      -0.4935    -6.1e-005         -167          155  Hessian modified twice
  183   4739      926.275      -0.4935    -6.1e-005         -167          155  Hessian modified twice
  184   4765      926.274      -0.4935    -6.1e-005         -167          155  Hessian modified twice
  185   4791      926.274      -0.4936    -6.1e-005         -167          155  Hessian modified twice
  186   4817      926.273      -0.4936    -6.1e-005         -167          155  Hessian modified twice
  187   4843      926.272      -0.4936    -6.1e-005         -167          155  Hessian modified twice
  188   4869      926.271      -0.4937    -6.1e-005         -167          155  Hessian modified twice
  189   4895       926.27      -0.4937    -6.1e-005         -167          155  Hessian modified twice
  190   4921      926.269      -0.4937    -6.1e-005         -167          154  Hessian modified twice
  191   4947      926.268      -0.4938    -6.1e-005         -168          154  Hessian modified twice
  192   4973      926.267      -0.4938    -6.1e-005         -168          154  Hessian modified twice
  193   4999      926.266      -0.4938    -6.1e-005         -168          154  Hessian modified twice
  194   5025      926.266      -0.4938    -6.1e-005         -168          154  Hessian modified twice
  195   5051      926.265      -0.4939    -6.1e-005         -168          154  Hessian modified twice
  196   5077      926.264      -0.4939    -6.1e-005         -168          154  Hessian modified twice
  197   5103      926.263      -0.4939    -6.1e-005         -168          154  Hessian modified twice
  198   5129      926.262       -0.494    -6.1e-005         -168          154  Hessian modified twice
  199   5155      926.261       -0.494    -6.1e-005         -168          154  Hessian modified twice
  200   5181       926.26       -0.494    -6.1e-005         -168          154  Hessian modified twice
Maximum number of iterations exceeded;
 increase OPTIONS.MaxIter.

x1 =

  Columns 1 through 9

  108.8146    1.0142   35.7455    0.5000   13.3370   55.0223   42.5363    0.4940    5.9557

  Columns 10 through 11

  144.9955   55.0000


z =

  926.2611


exitflag =

     0


output =

       iterations: 200
        funcCount: 5181
     lssteplength: -6.1035e-005
         stepsize: 9.2490e-004
        algorithm: 'medium-scale: SQP, Quasi-Newton, line-search'
    firstorderopt: 154.0397
          message: [1x65 char]

Warning: Large-scale (trust region) method does not currently solve
this type of problem,
 using medium-scale (line search) instead.
```

```
> In fmincon at 317
  In fminconn at 25

                                Max     Line search  Directional  First-order
 Iter F-count        f(x)   constraint   steplength   derivative   optimality Procedure
    0     12      926.261       -0.494
    1     38      926.261       -0.494    -6.1e-005         -174          161
    2     64       926.26      -0.4941    -6.1e-005         -179          174  Hessian modified twice
    3     90       926.26      -0.4941    -6.1e-005         -179          174  Hessian modified twice
    4    116       926.26      -0.4941    -6.1e-005         -177          176  Hessian modified twice
    5    142       926.26      -0.4941    -6.1e-005         -179          175  Hessian modified twice
    6    168       926.26      -0.4942    -6.1e-005         -177          175  Hessian modified twice
    7    194      926.259      -0.4942    -6.1e-005         -178          172  Hessian modified twice
    8    220      926.259      -0.4942    -6.1e-005         -176          174  Hessian modified twice
    9    246      926.259      -0.4943    -6.1e-005         -178          171  Hessian modified twice
   10    272      926.259      -0.4943    -6.1e-005         -176          174  Hessian modified twice
   11    298      926.258      -0.4943    -6.1e-005         -178          170  Hessian modified twice
   12    324      926.258      -0.4944    -6.1e-005         -178          175  Hessian modified twice
   13    350      926.258      -0.4944    -6.1e-005         -176          173  Hessian modified twice
   14    376      926.258      -0.4944    -6.1e-005         -178          170  Hessian modified twice
   15    402      926.257      -0.4944    -6.1e-005         -176          174  Hessian modified twice
   16    428      926.257      -0.4945    -6.1e-005         -178          169  Hessian modified twice
   17    454      926.257      -0.4945    -6.1e-005         -176          174  Hessian modified twice
   18    480      926.256      -0.4945    -6.1e-005         -178          169  Hessian modified twice
   19    506      926.256      -0.4946    -6.1e-005         -176          173  Hessian modified twice
   20    532      926.256      -0.4946    -6.1e-005         -178          169  Hessian modified twice
  [...]


  180   4692      926.166      -0.4994    -6.1e-005         -192          219  Hessian modified twice
  181   4718      926.165      -0.4994    -6.1e-005         -192          219  Hessian modified twice
  182   4744      926.165      -0.4995    -6.1e-005         -192          219  Hessian modified twice
  183   4770      926.164      -0.4995    -6.1e-005         -192          219  Hessian modified twice
  184   4796      926.164      -0.4995    -6.1e-005         -192          219  Hessian modified twice
  185   4822      926.163      -0.4995    -6.1e-005         -193          219  Hessian modified twice
  186   4848      926.162      -0.4996    -6.1e-005         -193          219  Hessian modified twice
  187   4874      926.162      -0.4996    -6.1e-005         -193          219  Hessian modified twice
  188   4900      926.161      -0.4996    -6.1e-005         -193          219  Hessian modified twice
  189   4926      926.161      -0.4997    -6.1e-005         -193          219  Hessian modified twice
  190   4952       926.16      -0.4997    -6.1e-005         -193          219  Hessian modified twice
  191   4978       926.16      -0.4997    -6.1e-005         -193          219  Hessian modified twice
  192   5004      926.159      -0.4998    -6.1e-005         -193          219  Hessian modified twice
  193   5030      926.159      -0.4998    -6.1e-005         -194          219  Hessian modified twice
  194   5056      926.158      -0.4998    -6.1e-005         -194          218  Hessian modified twice
  195   5082      926.157      -0.4999    -6.1e-005         -194          218  Hessian modified twice
  196   5108      926.157      -0.4999    -6.1e-005         -194          218  Hessian modified twice
  197   5134      926.156      -0.4999    -6.1e-005         -194          218  Hessian modified twice
  198   5160      926.156      -0.4999    -6.1e-005         -194          218  Hessian modified twice
  199   5186      926.155         -0.5    -6.1e-005         -194          218  Hessian modified twice
  200   5212      926.155         -0.5    -6.1e-005         -194          218  Hessian modified twice
Maximum number of iterations exceeded;
 increase OPTIONS.MaxIter.

xf =

  Columns 1 through 9

  108.8311    1.0131   35.7497    0.5000   13.2456   55.2160   42.5714    0.5000    5.9557

  Columns 10 through 11

  145.1354   55.0000
```

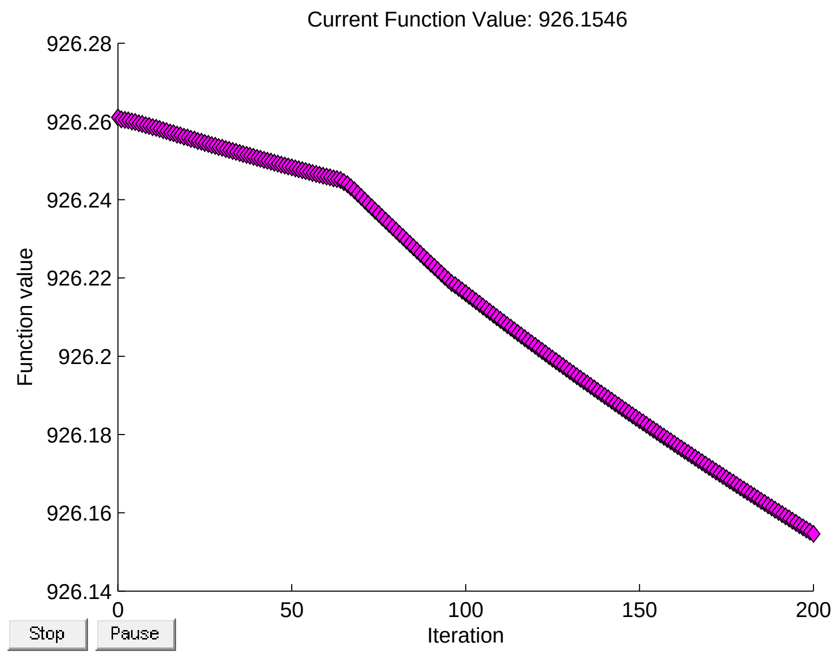


Figure 5.2: Another fmincon optimization session where the results of the first optimization run is the initial value for the second run.

```
z =

  926.1552


exitflag =

     0


output =

       iterations: 200
        funcCount: 5212
     lssteplength: -6.1035e-005
         stepsize: 0.0020
        algorithm: 'medium-scale: SQP, Quasi-Newton, line-search'
    firstorderopt: 218.3449
          message: [1x65 char]
```

As we can see, `exitflag` is still zero. The reason is shown in Figure (5.2), where it looks that the function values is still decreasing, but it is almost horizontal because the variation is very small.

We also try to change radically the starting point, for example:

```
x0=[110,1,30,0.5,15,50,50,0.3,5,145,60];
```

The iteration outputs and the final output are:

```
>> fminconn
Warning: Large-scale (trust region) method does not currently solve
this type of problem,
 using medium-scale (line search) instead.
> In fmincon at 317
  In fminconn at 20

                                Max     Line search  Directional  First-order
 Iter F-count        f(x)   constraint   steplength   derivative   optimality Procedure
    0      12      950.614         -0.3
    1      26      933.442       -0.475         0.25        -54.8          372
    2      40      928.524      -0.3938         0.25        -32.7         41.1
    3      55      926.766      -0.3445        0.125          147          440
    4      71      926.764      -0.3494       0.0625        -30.8          367
    5      97      926.764      -0.3494    -6.1e-005         -131          367
    6     123      926.764      -0.3494    -6.1e-005         -176          367
    7     149      926.763      -0.3493    -6.1e-005         -247          325
    8     175      926.762      -0.3493    -6.1e-005         -244          222  Hessian modified twice
    9     201      926.761      -0.3493    -6.1e-005         -242          230  Hessian modified twice
   10     227       926.76      -0.3492    -6.1e-005         -242          241  Hessian modified twice
   11     253       926.76      -0.3492    -6.1e-005         -242          192  Hessian modified twice
   12     279      926.759      -0.3491    -6.1e-005         -242          174  Hessian modified twice
   13     305      926.758      -0.3491    -6.1e-005         -242          213  Hessian modified twice
   14     331      926.757      -0.3491    -6.1e-005         -242          235  Hessian modified twice
   15     357      926.756       -0.349    -6.1e-005         -242          281  Hessian modified twice
   16     383      926.755       -0.349    -6.1e-005         -242          212  Hessian modified twice
   17     409      926.754      -0.3489    -6.1e-005         -242          157  Hessian modified twice
   18     435      926.753      -0.3489    -6.1e-005         -242          166  Hessian modified twice
   19     461      926.753      -0.3489    -6.1e-005         -242          190  Hessian modified twice
   20     487      926.752      -0.3488    -6.1e-005         -242          209  Hessian modified twice
  [...]


  180    4647      926.681      -0.3425    -6.1e-005         -252          242  Hessian modified twice
  181    4673      926.681      -0.3424    -6.1e-005         -252          242  Hessian modified twice
  182    4699      926.681      -0.3424    -6.1e-005         -252          243  Hessian modified twice
  183    4725      926.681      -0.3423    -6.1e-005         -252          243  Hessian modified twice
  184    4751      926.681      -0.3423    -6.1e-005         -252          244  Hessian modified twice
  185    4777       926.68      -0.3423    -6.1e-005         -252          244  Hessian modified twice
  186    4803       926.68      -0.3422    -6.1e-005         -252          244  Hessian modified twice
  187    4829       926.68      -0.3422    -6.1e-005         -252          245  Hessian modified twice
  188    4855       926.68      -0.3421    -6.1e-005         -252          245  Hessian modified twice
  189    4881       926.68      -0.3421    -6.1e-005         -252          246  Hessian modified twice
  190    4907       926.68      -0.3421    -6.1e-005         -252          246  Hessian modified twice
  191    4933       926.68       -0.342    -6.1e-005         -252          246  Hessian modified twice
  192    4959      926.679       -0.342    -6.1e-005         -252          246  Hessian modified twice
  193    4985      926.679      -0.3419    -6.1e-005         -252          247  Hessian modified twice
  194    5011      926.679      -0.3419    -6.1e-005         -191          362  Hessian modified twice
  195    5037      926.679      -0.3419    -6.1e-005         -202          362  Hessian modified twice
  196    5063      926.679      -0.3418    -6.1e-005         -201          362  Hessian modified twice
  197    5089      926.679      -0.3418    -6.1e-005         -204          362  Hessian modified twice
  198    5115      926.679      -0.3418    -6.1e-005         -209          362  Hessian modified twice
  199    5136      926.679      -0.3429      0.00195         -213          363  Hessian modified twice
```

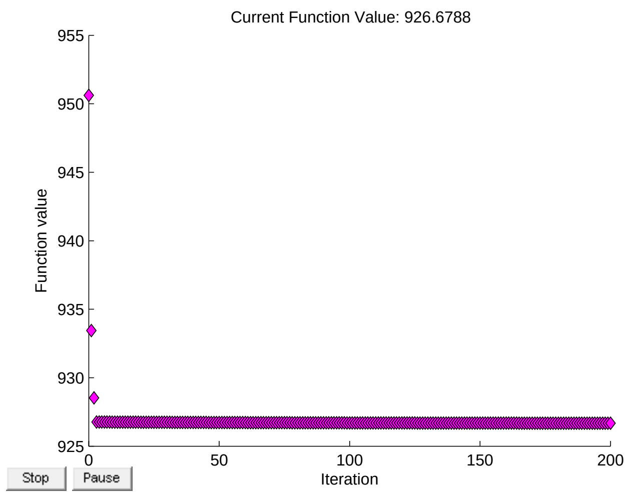


Figure 5.3: Optimizaton output with fmincon: another starting point x0=[110,1,30,0.5,15,50,50,0.3,5,145,60]

```
  200   5162      926.679      -0.3428    -6.1e-005         -254          206  Hessian modified twice
Maximum number of iterations exceeded;
 increase OPTIONS.MaxIter.

x1 =

  Columns 1 through 9

  107.6959    1.5371   30.4044    0.5000   14.8801   50.1110   48.9659    0.6571    5.0000

  Columns 10 through 11

  144.8931   60.0000


z =

  926.6789


exitflag =

     0


output =

       iterations: 200
        funcCount: 5162
     lssteplength: -6.1035e-005
         stepsize: 0.0017
        algorithm: 'medium-scale: SQP, Quasi-Newton, line-search'
    firstorderopt: 206.2510
          message: [1x65 char]
```

As we can see in the final output exitflag is zero and as we can see in the Figure (5.3) we made the same evaluation error. So we can conclude that this method is not good to our case study, because our problem is evidently not well define for `fmincon`, and that `patternsearch` optimization method is more appropriate.

5.2 Optimization results with "patternsearch" function

In this subsection we talk about the success of the `patternsearch` optimization method and we show and discuss about its results. The inputs are found among a set of points into the upper and lower bound values:

`x0=[115,1,30,0.5,13.5,55,45,0.5,5,145,70];`
The iteration outputs and final output are:

```
>> pattern_search


Iter     f-count          f(x)      MeshSize     Method
    0        1        1001.06             1
    1        2        983.101             2     Successful Poll
    2        4        981.909             4     Successful Poll
    3        6        980.232             8     Successful Poll
    4        8        978.931            16     Successful Poll
    5       11        941.926            32     Successful Poll
    6       12        941.926            16     Refine Mesh
    7       19        941.926             8     Refine Mesh
    8       21        941.814            16     Successful Poll
    9       26        937.219            32     Successful Poll
   10       29        937.219            16     Refine Mesh
   11       35        937.219             8     Refine Mesh
   12       39        936.745            16     Successful Poll
   13       46        936.745             8     Refine Mesh
   14       51        927.917            16     Successful Poll
   15       59        927.917             8     Refine Mesh
   16       64        927.504            16     Successful Poll
   17       72        927.504             8     Refine Mesh
   18       79        927.345            16     Successful Poll
   19       86        927.345             8     Refine Mesh
   20       96        927.345             4     Refine Mesh
  [...]


  151     1373        898.229    3.052e-005     Refine Mesh
  152     1376        898.229    6.104e-005     Successful Poll
  153     1392        898.229    3.052e-005     Refine Mesh
  154     1408        898.229    1.526e-005     Refine Mesh
  155     1424        898.229    7.629e-006     Refine Mesh
  156     1440        898.229    3.815e-006     Refine Mesh
  157     1443        898.229    7.629e-006     Successful Poll
  158     1459        898.229    3.815e-006     Refine Mesh
  159     1475        898.229    1.907e-006     Refine Mesh
  160     1478        898.229    3.815e-006     Successful Poll
  161     1494        898.229    1.907e-006     Refine Mesh
  162     1510        898.229    9.537e-007     Refine Mesh
Optimization terminated: change in the function value less than
options.TolFun.

x1 =

  101.0000    2.0000   50.0000    0.5000   11.5507   30.0000   20.0000    0.5161   10.0000

  155.9335   70.0000


z =

  898.2287


exitflag =

     3


output =

         function: @(x)obj_func(x)
      problemtype: 'boundconstraints'
       pollmethod: 'gpspositivebasis2n'
     searchmethod: []
       iterations: 162
        funccount: 1510
         meshsize: 9.5367e-007
    maxconstraint: 0
          message: 'Optimization terminated: change in the function value less than
                    options.TolFun.'
```

The iteration outputs are:

- Iter: number of iteration.
- f-count: number of function evaluation for each iteration.
- f(x): value of the objective function calculated with the input estimated for each iteration.
- MeshSize: Give the dimension of the mesh for each iteration.
- Method: For each iteration tell the state of the `patternsearch` algorithm.

`patternsearch` has found a solution. As we can see in the final output, the optimization problem is bound constrained, because we have define the lower and upper bound of the parameters to optimize. The final output shows that `exitflag=3`, or rather, that the change

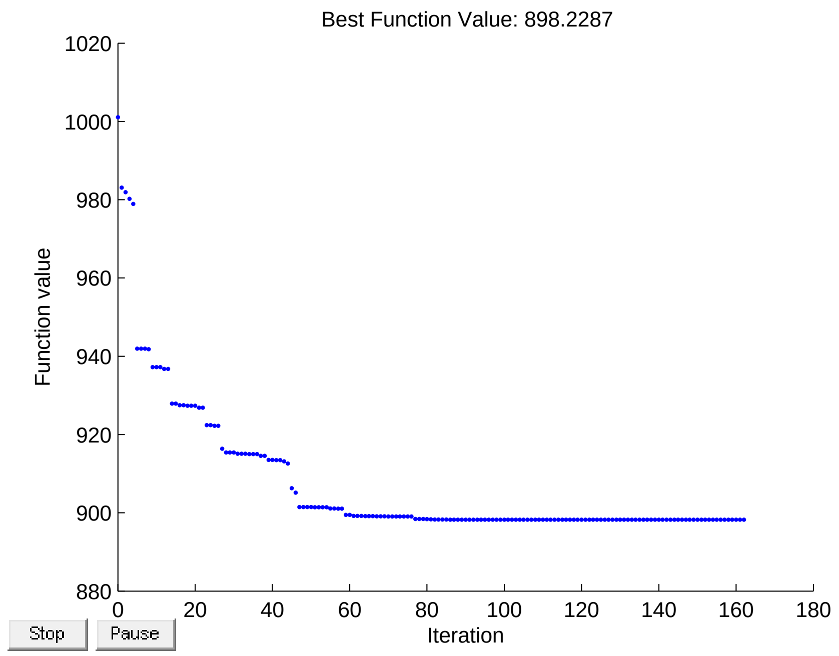


Figure 5.4: State of objective function during the optimization with patternsearch

in z is less than specified tolerance and the constraint violation less than specified ones. Increasing these values would allow the solution to converge more tightly on the 'Ideal' profile, but it would require longer solution times and moreover is not necessary, because as we can see in the Figure (5.4), the value of the objective function stays almost constant for a lot of iterations.

So we can consider as optimum the following solution:

```
x1=[101,2,50,0.5,11.5507,30,20,0.5161,10,155.9335,70]
```

or rather, the optimum values that calibrate our model are:

1. `v_free=101`
2. `a=2`
3. `rho_crit=50`
4. `alpha=0.5`
5. `tau=11.5507`
6. `eta=30`

7. `kappa=20`

8. `delta=0.5161`

9. `vmin=10`

10. `vmax=155.9335`

11. `rho_max=70`

The final value of the objective function after the optimization, or rather, the error between the "real data" coming from Paramics and the data coming from the model calculated with the optimum parameters, is:

`z=898.2287`

If we consider that this error is evaluated for 19 segment and 60 time steps, the average error for each segment in one time step is:

$$average\_error = \frac{898.2287}{19\times60\times2} \simeq 0.39$$

which seems good. In the next Figures (5.5), (5.6), (5.7) (flow), (5.8), (5.9), (5.10) (density), (5.11),(5.12), (5.13) (speed) we show the differences between the "real" outflow, density and speed, coming from Paramics, and the data coming from our model after the calibration for some segments, precisely the 1st, the 9th and the 19th.

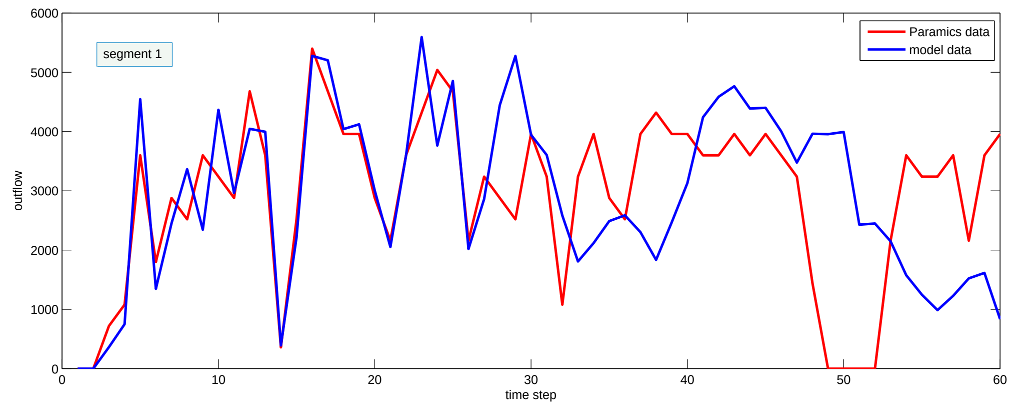


Figure 5.5: Comparison between Paramics and model_eindhoven flow data for the 1st segment

The Figure (5.5) shows that the results of comparison are almost good because both the values and the pattern of Paramics flow and model data correspond, with some difficulties in the last part of the period, because at the end are added all the computational error of the

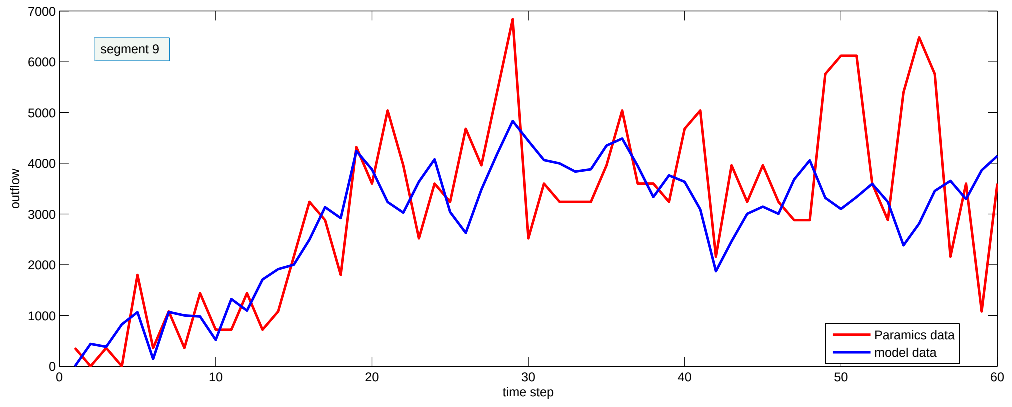


Figure 5.6: Comparison between Paramics and model_eindhoven flow data for the 9th segment

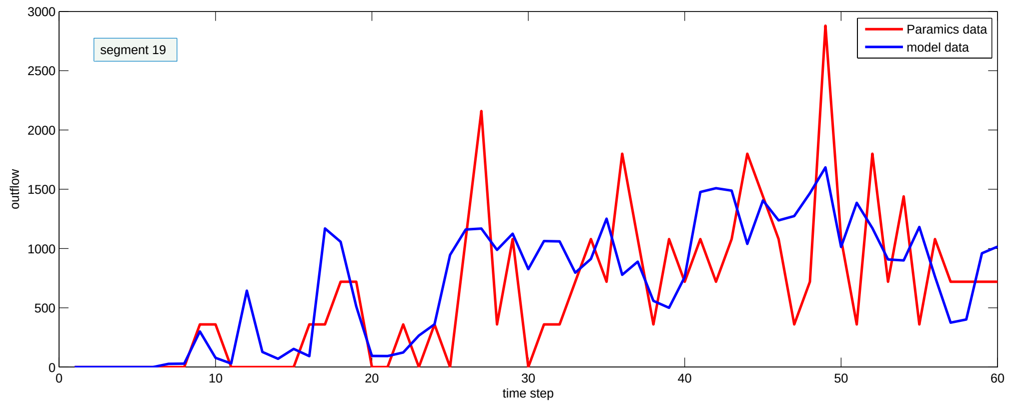


Figure 5.7: Comparison between Paramics and model_eindhoven flow data for the 19th segment

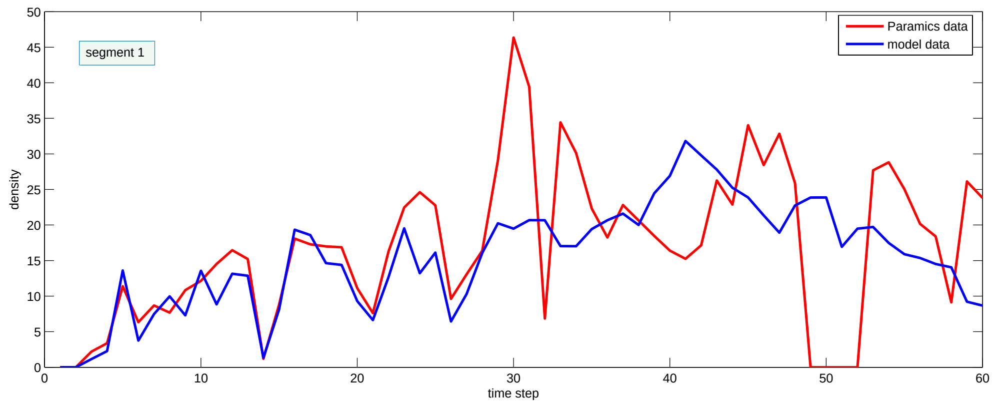


Figure 5.8: Comparison between Paramics and model_eindhoven density data for the 1st segment

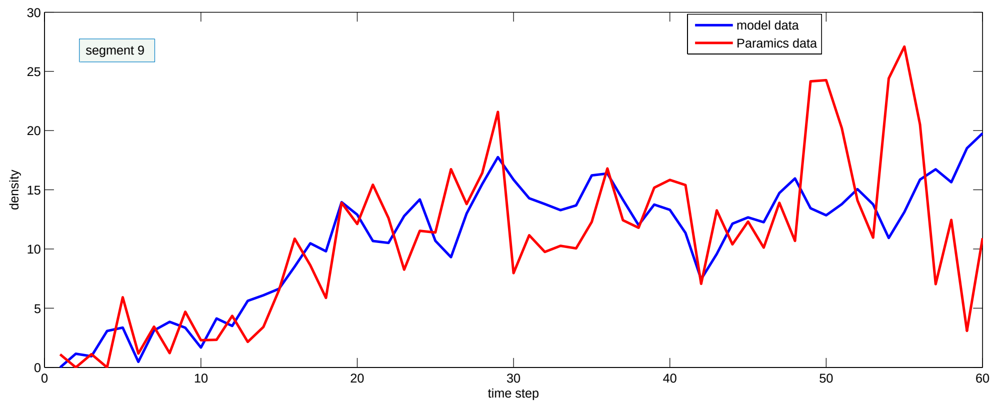


Figure 5.9: Comparison between Paramics and model_eindhoven density data for the 9th segment

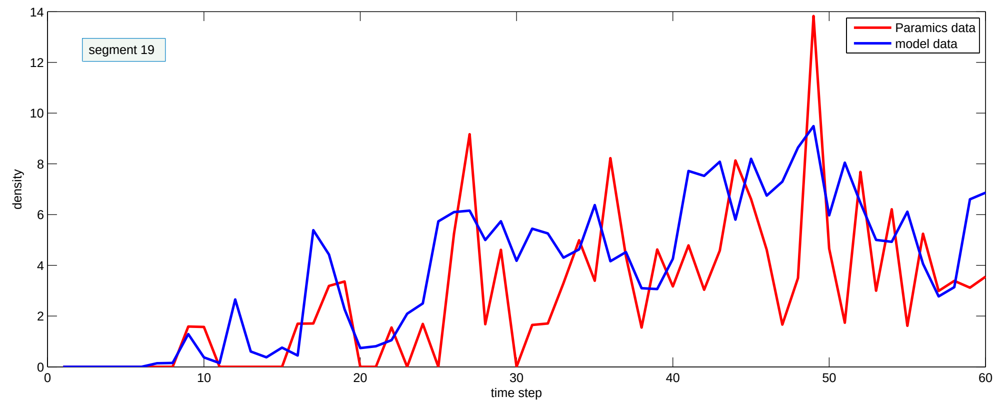


Figure 5.10: Comparison between Paramics and model_eindhoven density data for the 19th segment

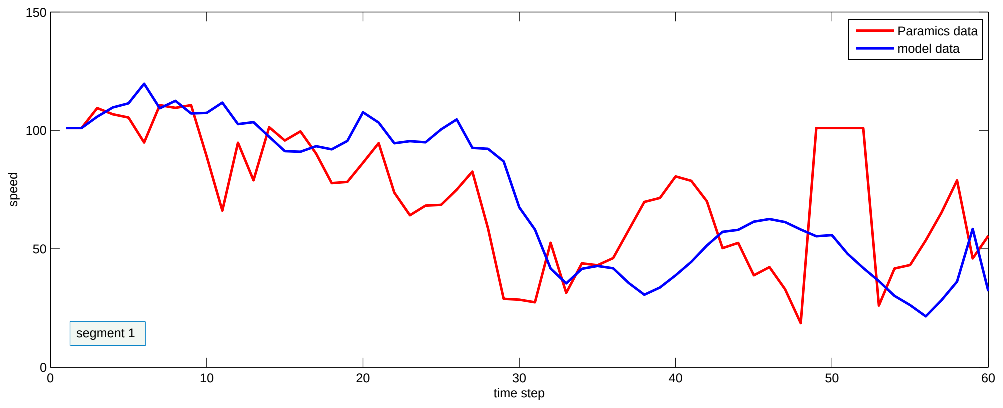


Figure 5.11: Comparison between Paramics and model_eindhoven speed data for the 1st segment

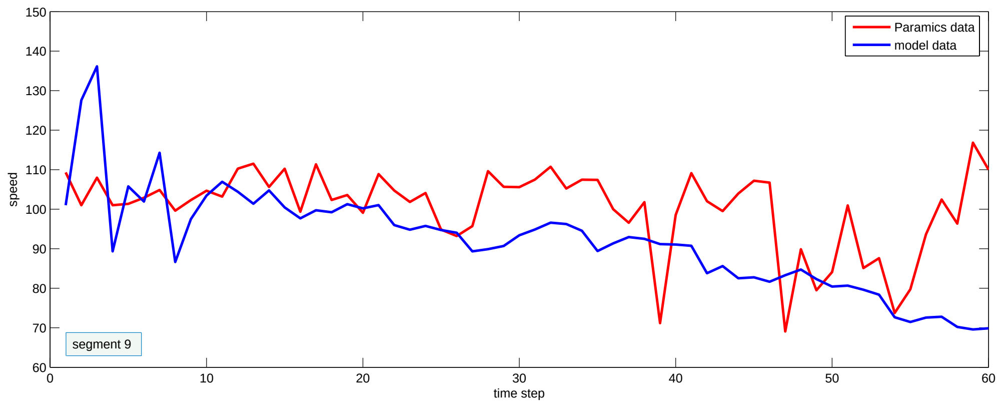


Figure 5.12: Comparison between Paramics and model_eindhoven speed data for the 9th segment

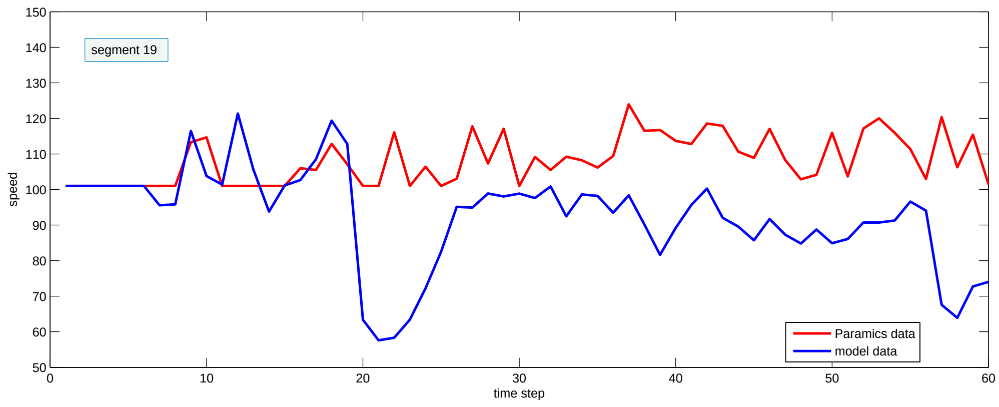


Figure 5.13: Comparison between Paramics and model_eindhoven speed data for the 19th segment

model. In the Figure (5.6) we can see that the result are good. The model data follows the real data in a good way. In the Figure (5.7) we can see a good convergence between Paramics and model outputs. The Figure (5.8) shows that there is a little problem where the Paramics data have a rapid increase of the values, but in average it is almost good. In the Figure (5.9) and (5.10) we can see a little no convergence but the results is good. The Figures (5.11) and (5.12) show that the results of speeds comparison are good, but the Figure (5.13) shows some problems.

5.3 Validation of the calibrated model

In this section we will validate the results obtained with the calibration.

The validation of results is a very important process because it shows if the model, with the new characteristic parameters obtained with the calibration, is ready to represent the real system. The purpose of the validation is, in fact, to check if the model has a good behavior with data coming from other Paramics simulations different from that used for the calibration. The configuration of the new Paramics simulation is the same, but Paramics gives for each simulation different outputs.

The value of the objective function, with the new data is:

```
z=835.8260
```

The average error for each segment for all the state variables in one time step is:

$average\_error = \frac{835.8260}{19\times60} \simeq 0.72$

that is similar to the result obtained for the calibration.

These results become more significant if we find the average error for each state variable.

So, the error for the average outflow is :

$outflow\_average\_error = \frac{697.0321}{19\times60} \simeq 0.60$

the error for the average speed is:

$speed\_average\_error = \frac{138.7939}{19\times60} \simeq 0.12$

and the error for the average density is:

$density\_average\_error = \frac{627.1939}{19\times 60} \simeq 0.55$

These results are not good, in particular for the outflow and the density. Nevertheless, the results shown in the Figures (5.14), (5.15), (5.16) (flow), (5.17), (5.18), (5.19) (density), (5.20),(5.21), (5.22) (speed) seem quite good.

This is due to 16th segment that introduces a big error on the evaluation of the results. In fact the 16th segment is too short and it is difficult for the METANET model gives a good implementation of it. The METANET model in fact describes average values and so is better for longer segments.

The 16th segment introduces errors for the next segments too.

If we consider the error for the average outflow calculated only for the firsts 15th segments, we obtain:

$outflow\_average\_error \simeq 0.28$

for the average density:

$outflow\_average\_error \simeq 0.32$

and, finally, for the average speed:

$outflow\_average\_error \simeq 0.11$

that are quite good only for the speed and good for the density and outflow.

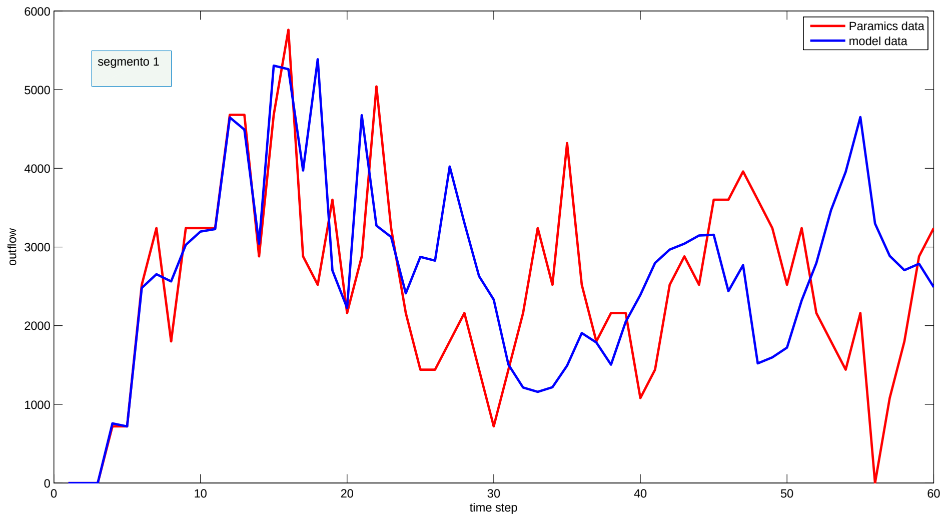


Figure 5.14: Comparison between Paramics and model_eindhoven flow data for the 1st segment

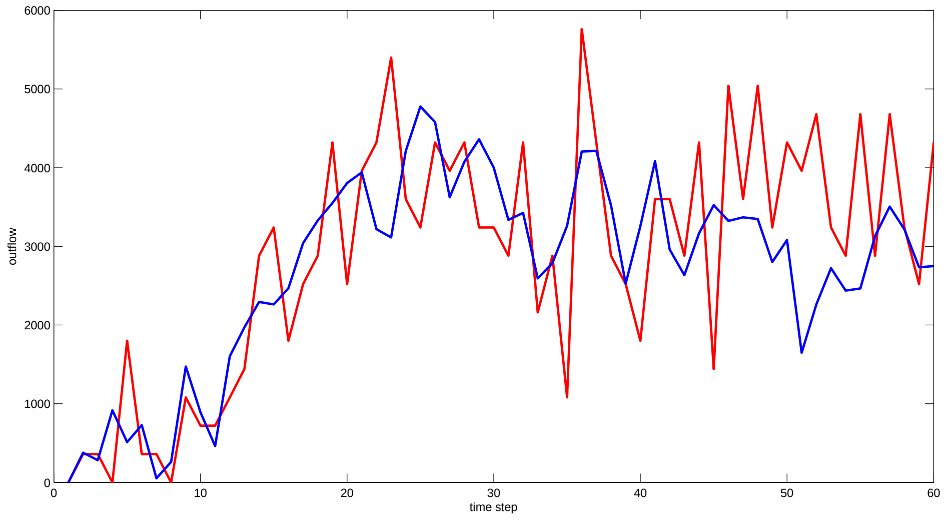


Figure 5.15: Comparison between Paramics and model_eindhoven flow data for the 9th segment

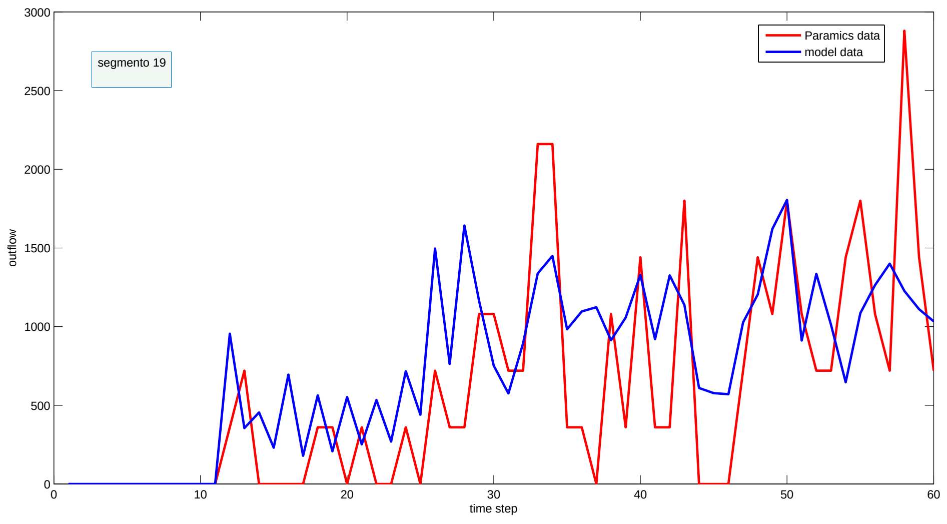


Figure 5.16: Comparison between Paramics and model_eindhoven flow data for the 19th segment

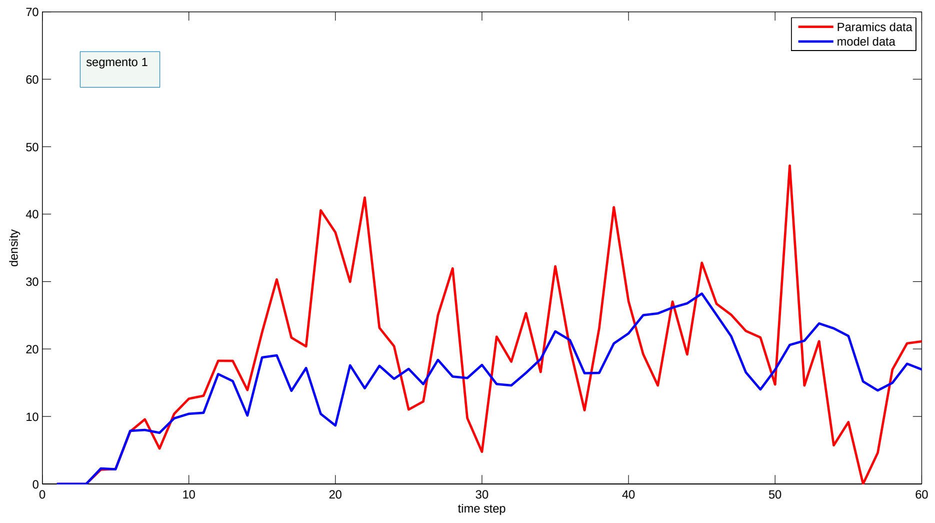


Figure 5.17: Comparison between Paramics and model_eindhoven density data for the 1st segment

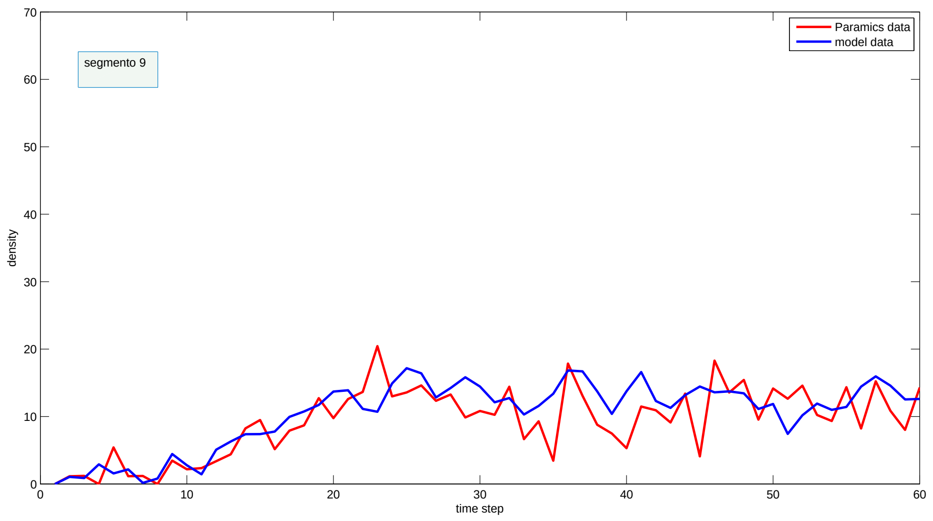


Figure 5.18: Comparison between Paramics and model_eindhoven density data for the 9th segment

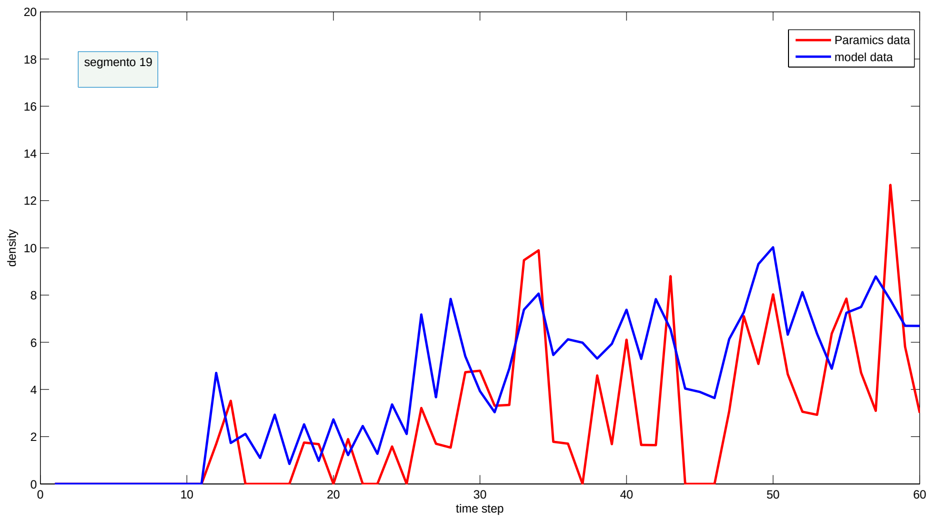


Figure 5.19: Comparison between Paramics and model_eindhoven density data for the 19th segment

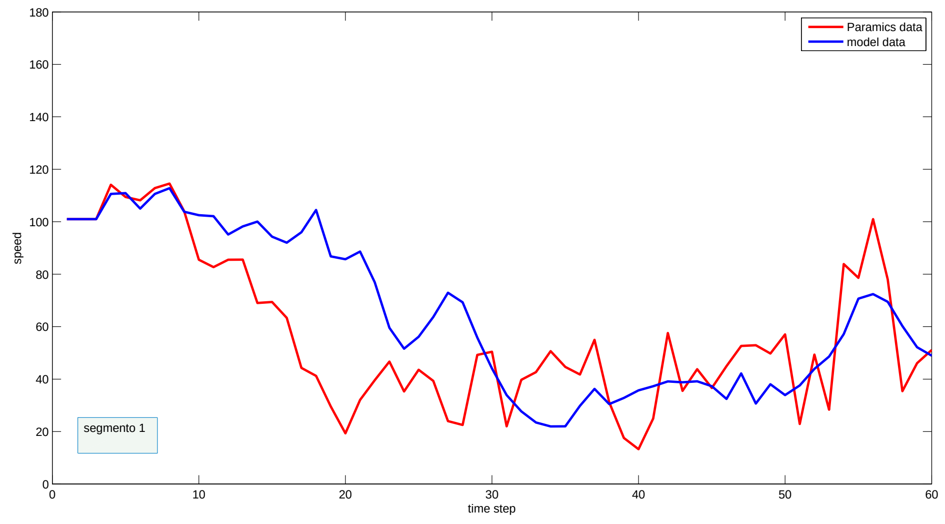


Figure 5.20: Comparison between Paramics and model_eindhoven speed data for the 1st segment

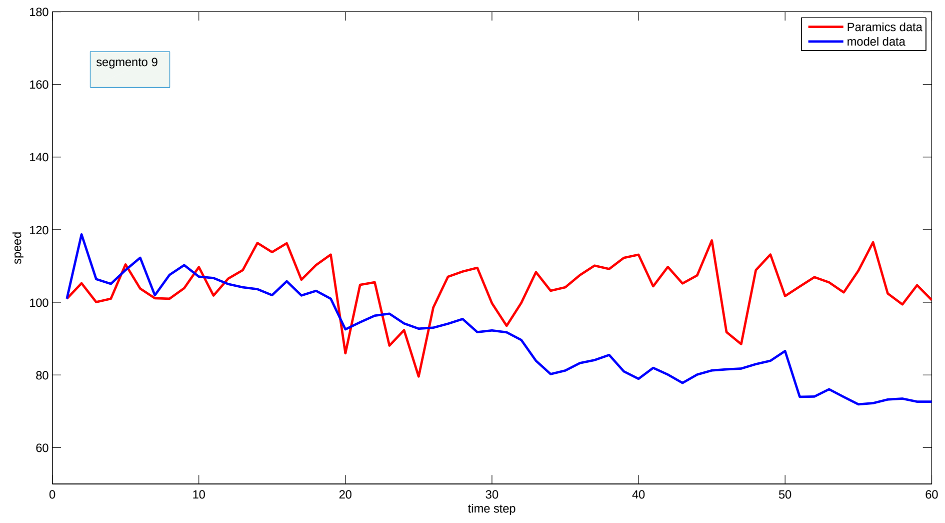


Figure 5.21: Comparison between Paramics and model_eindhoven speed data for the 9th segment

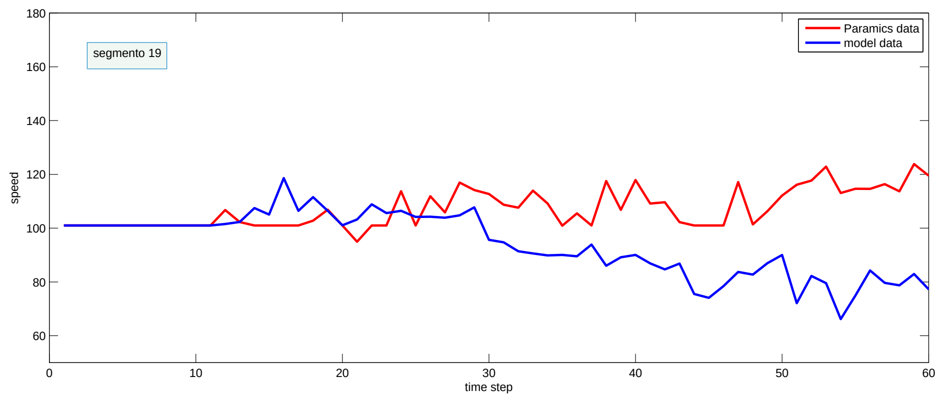


Figure 5.22: Comparison between Paramics and model_eindhoven speed data for the 19th segment

Chapter 6

Conclusion and future research

6.1 Conclusion

The case study of this thesis concerns the freeway network of the Dutch city of Eindhoven. This network, in fact, every day, is subjected to a lot of traffic flow problems, and a control policy is tightly necessary. We decided to represent this network using the model-based macroscopic traffic flow model METANET because it provides a good trade off between simulation speed and accuracy. This model needs to be calibrated in order to asses the accuracy. For this purpose we gave a MATLAB implementation of the network and we calibrated its parameters with the MATLAB optimization function *patternsearch* in order to reach a very realistic model. Then, during the validation, we compared the outputs (flow, density, speed) of a new Paramics simulation with the calibrated METANET model, to check if the model is good in other conditions too. We found a good results but with some exceptions. We will talk how to improve the efficiency in the next section "Future research". The calibrated and validated model can be used by the MPC (Model Predictive Control) in order to find the optimal control signal used by the controller to coordinate the traffic control measures.

6.2 Future Research

6.2.1 Future short term work

The differences between real and the simulated data can be decrease adopting the following rules:

- Increasing the precision of the optimization method.
- Decreasing the duration of the time step, in order to have more accurate values of the parameters.
- To find the parameter values valid for each segment, and not only for the total network.

These rules increase a lot the computational effort, but are necessary to reach a good control strategy.

6.2.2 Future long term work

In the long term the calibrated model can be used, as already said, by the MPC in order to solve the problem of coordination of speed limits, ramp metering and route guidance. The goal of the controller is, in fact, to find the control signals that results in an optimal process traffic behavior [2].

This control method can include not only freeway networks but also urban roads. They are closely connected: congestion on the freeway often causes spill back of urban queues, slowing down the urban traffic and viceversa. As a consequence, control measures taken in one of the areas can have significant influence on the other area [34].

Bibliography

[1] A. Kotsialos and M. Papageorgiou. Freeway ramp metering: An overview. *IEEE Transactions on Intelligent Transportation Systems*, 3(4):271–280, 2002.

[2] A. Hegyi. *Model Predictive Control for Integrating Traffic Control Measures*. PhD thesis, Delft University of Technology, Delft, The Netherlands, February 2004. TRAIL Thesis Series T2004/2.

[3] A. Alessandri, A. Di Febbraro, A. Ferrara, and E. Punta. Freeway ramp metering: An overview. *Control Engineering Practice*, 6(3):771–780, 1998.

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

[5] C. Chien, C. Zhang, and Y. Ioannou. Tutorial overview of model predictive control. *Automatica*, 33(7):1273–1285, 1997.

[6] H. Lenz, R. Sollacher, and R. Lang. Standing waves and the influence of speed limits. In *European Control Conference*, pages 1228–1232, Porto, Portugal, May 2001.

[7] B.H. Heutinck, M.van den Berg, J.Hellendoorn, and L.H. Immers. Dynamic route guidance during maintenance works, a case study.

[8] A. Kotsialos, M. Papageorgiou, and A. Messmer. Integrated optimal control of motorway traffic networks. In $18th$ *American Control Conference*, pages 2183–2187, June 1999.

[9] M. Papageorgiou. Dynamic modeling, assignment, and route guidance in traffic networks. *Transportation Research Part B*, 24B(6):471–495, 1990.

[10] M.Papageorgiou and A. Messmer. Dynamic network traffic assignment and route guidance via feedback regulation. *Transportation Research Record*, 1306(2):49–58, 1991.

[11] M. Kraan, N. van der Zijpp, B. Tutert, T. Vonk, and D. van Megen. Evaluating network-wide effects of variable message signs in the netherlands. *Transportation Research Record*, (1689):60–67, 1999.

[12] A. Kotsialos and M. Papageorgiou. Motorway network traffic control systems. *European Journal of Operational Research*, 152(2):321–3334, 2004.

[13] M. Papageorgiou. Some remarks on macroscopic traffic flow modelling. *Transportation Research Part A*, 32(5):323–398, Sept 1998.

[14] Bellemans, B. De Schutter, , and B. De Moor. Anticipative model predictive control for ramp metering in freeway networks. In *American Control Conference*, pages 4077–4082, Denver,Colorado, June 2003.

[15] A. Hegyi, B. De Schutter, and H. Hellendoorn. Model predictive control for optimal coordination of ramp metering and variable speed limits. *Transportation Research Part C*, 13(3):185–209, june 2005.

[16] A. Hegyi, B. De Schutter, and J. Hellendoorn. Optimal coordination of variable speed limits to suppress shock waves. In *The 7th TRAIL Congress - TRAILblazing into the Future*, pages 197–220, Rotterdam, The Netherlands, November 2002.

[17] M.Cremer and M. Papageorgiou. Parameter identification for a traffic flow model. *Brief Paper Automatica*, 17(6):837–843, Nov 1981.

[18] D. Ngoduy and S.P. Hoogendoorn. An automated calibration procedure for macroscopic traffic flow models. In *The10th Symposium on Control in Transportation Systems 1*, page 168, mm, 1980.

[19] J. Cuena, J. Hernandez, and M. Molina. Knowledge-based models for adaptive traffic management systems. *Transportation Research Part C*, 3(5):311–337, 1995.

[20] Adel W. Sadek, Brian L. Smith, and Micheal J. Demetsky. A prototype case-based reasoning system for real-time freeway traffic routing. *Transportation Research Part C*, 9:353–380, 2001.

[21] A. Messmer and M. Papageorgiou. Motorway network control via nonlinear optimization. *International Transactions in Operational Research*, 2(2):187–203, April 1995.

[22] J.M. Morin, B. Baradel, and J.F. Gabard. Motorway network control: Aid-to-decision approach. In *Deliverable 16 of Drive project Christiane, Part B*, page 120, mm, 1992.

[23] G. Ambrosino, M. Boero, and M. Mastretta anf G. Bielli. *Expert Systems Approach to Road Traffic Control Control. In M. Papageorgiou (Ed.) Concise Encyclopedia of Traffic and Transportation Systems, pp. 124-130:*. Oxford: Pergamon, Englewood Cliffs, New Jersey, 1991.

[24] F. Maghrebi and F. Boillot. Neural networks application in the urban traffic control field. In *1st Meeting of the EURO Working Group on Urban Traffic and Transportation*, page 120, Landshut, Germany, 1992.

[25] J. Azema. Anwendung des neuronalen netzes auf die verkehrssteuerung. In *34 at Lehrstuhl f/Jr Steuerungs- und Regelungstechnik*, page 234, Technical University of Munich, Germany, May 1992.

[26] James B.Rawlings. Tutorial overview of model predictive control. *Control Systems Magazine, IEEE*, 20(3):38–52, june 2000.

[27] M. van den Berg, A. Hegyi B. De Schutter, and J. Hellendoorn. Model predictive control for mixed urban and freeway networks. In *The 83rd Annual Meeting of the Transportation Research Board*, page 19, Washington DC, January 2004.

[28] B.H. Heutinck, M. van der Berg, J. Hellendoorn, and L.H Immers. Dynamic route guidance during maintenance works, a case study.

[29] Quadstone. Paramicsv5: Modellerreferencemanual.

[30] Quadstone. Paramicsv5: Modelleruserguide.

[31] T.F. Coleman and Y. Li. An interior, trust region approach for nonlinear minimization subject to bounds. *SIAM Journal on Optimization*, 6(5):418–445, 1996.

[32] T.F. Coleman and Y. Li. On the convergence of reflective newton methods for large-scale nonlinear minimization subject to bounds. *Mathematical Programming*, 67(2):189–224, 1994.

[33] D. Goldfarb. A family of variable metric updates derived by variational means. *Mathematics of Computing*, 24(5):23–26, 1970.

[34] M. van der Berg, J. Hellendoorn, and B. De Schutter. Integrated model predictive control for mixed urban and freeway networks. *Trail Research School, Delft*, 3(5), November 2004.