Robust Traffic Signal Control for Uncertain Road Networks

Shimon Komarovsky

Robust Traffic Signal Control for Uncertain Road Networks

Research Thesis

In Partial Fulfilment of the Requirements for the Degree of
Master of Science in Transportation Sciences

Shimon Komarovsky

Submitted to the Senate of the Technion - Israel Institute of Technology
Tishrei, 5780 Haifa October, 2019

The Research Thesis Was Done Under The Supervision of Assoc. Prof. Jack Haddad.

List of Publications

Some results in this thesis have been published as an article in a conference during the course of the author's research period:

- Shimon Komarovsky and Jack Haddad, "Robust interpolating traffic signal control for uncertain road networks," in *Proceedings of the 2019 18th European Control Conference (ECC'19)*, pages 3656–3661, IEEE, 2019.

The generous financial support of the Technion and the Andrea & Alfredo Pollitzer Memorial Fellowship by Mr. and Mrs. Lepri is gratefully acknowledged.

Contents

List of Figures

List of Tables

List of Symbols and Abbreviations

$\mathbf{d}(k)$ Disturbance vector which consists of the demand flows of each link, in cycle k, [veh/sec], see equation (3.6) 17

$\mathbf{d}^{\mathrm{N}}$ Demand nominal vector, [veh/sec] 17

$\Delta\mathbf{g}(k)$ Control input vector for a network, which is green duration period, [sec], see equation (2.1) 5

$\Delta\mathbf{g}_0$ Local controller, see equation (4.20) 27

$\Delta\mathbf{g}_a$ Inner SIC linear saturated controller 33

$\Delta\mathbf{g}_b$ Outer SIC linear saturated controller 33

$\Delta\mathbf{g}_l$ An admissible vertex control, for each of the vertices of C_N, see equation (4.14) 25

$\Delta\mathbf{g}_v$ Vertex controller, see equation (4.20) 27

$\Delta\mathbf{x}(k)$ State vector of all vehicles in the whole network, [veh], see equation (2.1) 5

$\Delta\mathbf{x}_l$ Vertex of C_N, see equation (4.14) 25

$\mathbf{g}^{\mathrm{N}}$ Control input nominal vector, [sec] 17

$\mathbf{g}_N$ Control input matrix consists of N control input vectors, [sec], see equation (2.2) 6

$\Omega_{\max}$ Invariant set or maximal admissible invariant set, see equation (4.12) 23

$\tilde{T}$ Sampling or Control interval, [sec], see equation (3.4) 16

L Gain matrix for a linear state-feedback controller, [sec/veh], see equation (2.3) 6

L_a Gain matrix for the inner SIC controller, [sec/veh] 33

L_b Gain matrix for the outer SIC controller, [sec/veh] 33

L_j The total lost time of intersection j, [sec] 13

N Prediction horizon of MPC or control invariant horizon, [simulation steps], see equation (2.2) 6

N_{m} Control horizon, in MPC method, [simulation steps] 8

P_a Inner SIC ellipsoidal invariant set 34

P_b Outer SIC ellipsoidal invariant set 34

Q Weighting matrix for state quadratic terms in the quadratic objective function, see equation (2.2) 6

$q_z(k)$ average inflow to link z througout the cycle k, [veh/sec], see equation (3.4) 16

R Weighting matrix for control quadratic terms in the quadratic objective function, see equation (2.2) 6

$s_z(k)$ exit flow within link z througout the cycle k, [veh/sec], see equation (3.4) 16

T Diagonal matrix, consists of $\tilde{T}$ at its diagonal, [sec], see equation (3.5) 16

$u_z(k)$ average outflow to link z througout the cycle k, [veh/sec], see equation (3.4) 16

$x_z(k)$ number of vehicles within link z at the beginning of cycle k, [veh], see equation (3.4) 16

$x_z^{\max}$ Maximum admissible queue length for link z, [veh], see equation (3.13) 19

ARE Algebraic Riccati Equation 7

DARE Discrete Algebraic Riccati Equation 7

DP Dynamical Programming 6

IC Interpolating Control . . . 3

IIC Improved IC method . . . 27

LMI Linear Matrix Inequality . . . 33

LP Linear Programming . . . 2

LQR Linear Quadratic Regulator . . . 3

MAS Maximum Admissible invariant Set . . . 22

meIIC more efficient improved IC method . . . 30

MPC Model Predictive Control . . . 3

NLP NonLinear Programming . . . 2

QP Quadratic Programming . . . 2

RHS Right Hand Side . . . 31

RMPC Robust Model Predictive Control . . . 3

SDP Semi-Definite Problem . . . 37

SF Store-and-Forward model . . . 2

SIC Simple Interpolating Control . . . 3

Abstract

As congestion in road networks remains a severe and current issue, many traffic control strategies try to tackle this problem. Many of such strategies in the literature are designed based on various traffic flow models, which do not include parameter uncertainties in their intersection queuing dynamics. Hence, these control strategies might handle poorly conditions when assumed values for the parameters vary significantly from their real values. Moreover, many utilized state feedback control approaches to develop the traffic signal controllers lack the treatment of control and state constraints directly in the design phase.

This thesis considers robust traffic signal control for uncertain urban road networks. First, parameter uncertainties are integrated into the store-and-forward (SF) model, which is utilized in this thesis to describe the queuing dynamics for traffic signalized intersections. The SF model is unique in a sense that it simplifies traffic flow description from discrete-time modes to a continuous-time mode over a cycle, by averaging the flow throughout the whole cycle. This simplification allows the implementation of various optimization and control methods and schemes.

The SF model is first upgraded to include traffic flow uncertainties to reflect real urban traffic dynamics. Then, the uncertain SF model is utilized to design a robust feedback controller by an interpolation-based approach. In this thesis two versions of this approach are implemented: Interpolating Control (IC) and Simple Interpolating Control (SIC). This approach (i) guarantees robustness against all assumed parameter uncertainties, (ii) handles control and state constraints, and (iii) presents a computationally cheap solution.

Finally, numerical results for an isolated signalized intersection show a comparison between the developed interpolating controllers and other controllers in the literature. The results demonstrate the performance advantages from applying the robust interpolating controllers.

Chapter 1

Introduction

Mitigating congestion in urban road networks still remains as one of the current great challenges in traffic control. To address the traffic congestion, many model-based control strategies have been developed for urban traffic networks during the past few decades, see e.g. [3].

The Store-and-Forward (SF) model is one of the most prominent models in the literature [2], as several variants have been introduced, and many traffic control strategies have been designed based on these variants [2, 4–8]. The core modeling idea of the SF is introducing a model simplification that enables the mathematical description of the traffic flow process at signal light without the use of discrete variables. I.e., by converting discrete-event modes - green and red durations, to a continuous-time mode over the cycle, yet a discrete dynamic among cycles (see detailed explanation in Section 3.1). In other words, the SF model disregards the inner dynamics within a cycle of green phase and then red phase. On one hand, the latter might reduce the model accuracy, while on the other hand, it can result in a simplified model that can be utilized for control design.

This model simplification enables the application of a number of highly efficient optimization and control methods, such as linear programming (LP), quadratic programming (QP), nonlinear programming (NLP), [9, 10], and multivariable regulators with polynomial complexity, [11]. However, most traffic control strategies have been designed based on traffic flow models, which do not include parameter uncertainties in their intersection queuing dynamics, and only a few variants deal explicitly with model uncertainties [12]. The parameter uncertainties may include the saturation flow rates, turning movement rates, and others, that are difficult to estimate in reality. The model parameters are assumed to be a priori known or estimated. Hence, the resulted control strategies are unable to handle conditions when the assumed values for the parameters vary significantly from their real values,

e.g. applying variants of linear quadratic regulator (LQR) on uncertain traffic model might cause low control performances, as they cannot compensate parameter uncertainties.

Moreover, many state feedback control approaches which are utilized to develop the traffic signal controllers lack the treatment of control and state constraints directly in the design phase.

Recently, the traffic signal control problem was dealt by various advanced control strategies, such as online recalculation using model predictive control (MPC) or rolling-horizon control (RHC), see e.g. [2, 13, 14]. However, these and other upgraded strategies were based on nominal demand and model information, i.e., some of them even do not explicitly include any model mismatch. Consequently new approaches were proposed to consider uncertainties, mainly robust control methods: signal calculation method accounting daily traffic flow variability [4], robustly optimal timing methods to minimize the mean of delays per vehicles under changing traffic flow demand [5], a robust signal split method capturing origin-destination uncertainties [6], a stochastic optimization approach, including probability information of uncertain and day-to-day changing traffic demand [8], and recently a robust real-time model predictive control (RMPC) approach [7].

Robust control deals with unknown but bounded model parameters, as it optimizes an objective function and/or operates under control specifications when worst case scenarios are taken into account. Indeed, robust control can compensate the parameter uncertainties, and it can be very effective for systems with structured parameter variations [15], however, robust control usually results in a conservative design.

In this thesis, an uncertain SF model is first developed. The uncertain SF model is then utilized to design a robust feedback controller by an interpolation-based approach [16]. The theory of interpolation-based control was developed in [17] and [18], while recently was implemented for traffic perimeter control problems, e.g. [19], and was applied in lab experiments of vehicle platoon formation in [20]. The results in these previous works encourage us to study the interpolation-based control on the traffic signal control problem. The interpolation-based approach (i) guarantees robustness against all parameter uncertainties, (ii) handles control and state constraints, and (iii) presents a computationally cheap solution. Two variants of interpolation-based control methods, i.e. Interpolating Control (IC) and Simple Interpolating Control (SIC), are applied.

Finally, considering different layouts of signalized intersections, a comparison between the developed interpolating controllers and other controllers in the literature is conducted. All controllers are designed with the objective to regulate the plant state to the origin. The results demonstrate the

performance advantages from applying the robust interpolating controllers.

1.1 Research goals

The research goals of this thesis are as follows:

- Developing an uncertain urban road model based on SF approach modeling. The developed model should be uncertain linear state model, which includes parameter uncertainties, in order to be more realistic compared with the real behavior.
- Based on the developed model, introducing a robust control algorithm that incorporates control and state constraints.
- Comparing the results obtained from the developed model and algorithm with other control strategies from the literature, e.g. LQR and MPC.

1.2 Thesis outline

Following the introduction and related work, this thesis contains three main chapters. Chapter 3 begins with developing an uncertain model of road links in a network, in order to formulate a robust control problem. Once the problem is defined, the Interpolating Control method is outlined in Chapter 4. Next, the Simple Interpolating Control approach is introduced in Chapter 5, which is a new and more simpler version of the Interpolating Control.

Chapter 6 focuses on performing comparison of several methods introduced in this research, in order to gain some insights about different aspects, advantages and disadvantages of these control methods.

Chapters 7–9 are numerical results of some examples, starting in Chapter 7 with the case of 2-dimensional state space, i.e. 2 links. Then, dealing with the case of 3-dimensional state space in Chapter 8, and the case of 4-dimensional state space in Chapter 9. In the case of 2 links, a single intersection is considered, but in cases of 3 and 4 links, both single intersection system and two interconnected intersections system are considered. Finally, Chapter 10 summarizes the conclusions from numerical results from chapters 7–9 and gives some suggestion for a future work.

Chapter 2

Related work on traffic signal control design

Traffic signal control for road networks has a long history. In the literature, there are many works that utilize the SF model to design the traffic control inputs, e.g. green light durations, where the goal is to optimize the network performance. Various control methods and approaches have been implemented to design the control inputs [2, 4–8]. In the following, since the LQR [21, 22] and MPC [14, 23, 24] approaches are widely used in the traffic signal control literature, we focus only on LQR [21, 22] and MPC [14, 23, 24] to present and motivate our research questions.

Both LQR and MPC comprise of the same linear discrete-time model and optimal quadratic objective function. The linear discrete-time model, which describes the dynamics of the system, is given by

$$\Delta\mathbf{x}(k+1) = A\Delta\mathbf{x}(k) + B\Delta\mathbf{g}(k)\,, \tag{2.1}$$

where $\Delta\mathbf{x}(k) = \mathbf{x}(k) - \mathbf{x}^{\mathrm{N}}$, $\Delta\mathbf{g}(k) = \mathbf{g}(k) - \mathbf{g}^{\mathrm{N}}$, and $\Delta\mathbf{x}(k) \in \mathbb{R}^n, \Delta\mathbf{g}(k) \in \mathbb{R}^m$ are deviated state and control variables (deviated from nominal values), respectively, n and m are the dimensions of the state and control vector variables, respectively, and $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$. $\mathbf{x}(k) \in \mathbb{R}^n$ is the state vector, and $\mathbf{g}(k) \in \mathbb{R}^m$ is the control vector, and $(\mathbf{x}^{\mathrm{N}}, \mathbf{g}^{\mathrm{N}})$ is the equilibrium point or the nominal values of the state and control vectors, respectively.

Following the discrete dynamic model, the quadratic objective function J is defined with a finite horizon N as follows

$$J(\Delta\mathbf{x}(k)) = \min_{\mathbf{g}_N} \frac{1}{2}\left(\sum_{\tau=k}^{k+N-1} \Delta\mathbf{g}(\tau)^T R\Delta\mathbf{g}(\tau) + \sum_{\tau=k+1}^{k+N} \Delta\mathbf{x}(\tau)^T Q\Delta\mathbf{x}(\tau)\right), \tag{2.2}$$

where $\mathbf{g}_N = [\Delta\mathbf{g}(k), \Delta\mathbf{g}(k+1), \ldots, \Delta\mathbf{g}(k+N-1)]^T$, and $R \in \mathbb{R}^{m\times m}, Q \in \mathbb{R}^{n\times n}$ are weighting matrices for control and state quadratic terms, respectively. The weighting purpose is to determine the relative objective of driving the deviated state and control variables to the origin. The matrices R and Q are positive-definite and semi-positive-definite, respectively, i.e. $R \succ 0$, $Q \succeq 0$. Note that the initial state $\Delta\mathbf{x}(k)$ is a priori given, therefore, it is neglected in (2.2) in the second term. Another important note is that the formulated MPC and LQR model in (2.1) is for a nominal case, without including parameter uncertainties.

2.1 Linear Quadratic Regulator (LQR)

LQR is a state-feedback closed-form

$$\Delta\mathbf{g}(k) = L\Delta\mathbf{x}(k), \tag{2.3}$$

whose gain matrix $L \in \mathbb{R}^{m\times n}$ is calculated once offline for a fixed time window, called finite-horizon. L is calculated by solving the well-known Riccati equation. The solution of finite-horizon LQR depends on the linear dynamic matrices A, B and the objective function parameters Q, R, which do not change over the control process.

In order to obtain a feasible and stable solution, one need to solve the Riccati equation, which can be solved in different ways, e.g. with the help of Lagrange multipliers as shown in [25], [22]. Another option, which is used in here, is to utilize dynamical programming (DP) [26] with a backtracking algorithm.

For the purpose of calculating the gain matrix L, the terminal state term in (2.2) is separated as follows

$$\begin{aligned} J(\Delta\mathbf{x}(k)) = \min_{\mathbf{g}_N} \frac{1}{2} \sum_{\tau=k}^{k+N-1} \left(\Delta\mathbf{g}(\tau)^T R \Delta\mathbf{g}(\tau) + \Delta\mathbf{x}(\tau)^T Q \Delta\mathbf{x}(\tau)\right) \\ + \frac{1}{2}\Delta\mathbf{x}(k+N)^T Q \Delta\mathbf{x}(k+N) - \frac{1}{2}\Delta\mathbf{x}(k)^T Q \Delta\mathbf{x}(k) \end{aligned} \tag{2.4}$$

where $\Delta\mathbf{x}(k)^T Q \Delta\mathbf{x}(k)$ is known, given $\Delta\mathbf{x}(k)$ is known, and the terminal term $\Delta\mathbf{x}(k+N)^T Q \Delta\mathbf{x}(k+N)$ is used as a starting point for a backtracking algorithm that solves a Riccati equation and eventually yields the gain matrix. This terminal term can be weighted by Q or by any other semi-positive-definite matrix, which eventually will result different P solutions of Riccati equation.

Hence, the backtracking algorithm is run according to the following expressions for each step t, in the finite horizon $k \leq t \leq k+N-1$:

$$\Delta \mathbf{g}(t) = L_t \Delta \mathbf{x}(t)\,, \tag{2.5a}$$
$$L_t = -(B^T P_{t+1} B + R)^{-1} B^T P_{t+1} A\,, \tag{2.5b}$$
$$P_t = (A + BL_t)^T P_{t+1}(A + BL_t) + Q + L_t^T R L_t\,, \tag{2.5c}$$
$$J(\Delta \mathbf{x}(t)) = \frac{1}{2} \Delta \mathbf{x}(t)^T P_t \Delta \mathbf{x}(t)\,, \tag{2.5d}$$

where $\Delta \mathbf{g}(t)$, calculated from (2.5a), is minimizing the cost function (2.5d) from step t to $t+1$, using the gain matrix L_t (2.5b) which is dependent on $P_t \in \mathbb{R}^{n \times n}$ unknown matrix.

The matrix P_t is calculated as follows. Backward steps in (2.5c) are applied starting from the terminal condition $J(\Delta \mathbf{x}(k+N)) = \frac{1}{2} \Delta \mathbf{x}(k+N)^T P_{k+N} \Delta \mathbf{x}(k+N)$, i.e. $P_{k+N} = Q \rightarrow P_{k+N-1} \rightarrow P_{k+N-2} \rightarrow \ldots \rightarrow P_{k+1}$, such that one can find L_k using (2.5b), and thus implementing the first optimal control signal $\Delta \mathbf{g}(k) = L_k \Delta \mathbf{x}(k)$.

It means that in LQR we apply the first optimal control input $\Delta \mathbf{g}(k)$ at each iteration. However, since nothing changes by moving the horizon one step forward, except the current state $\Delta \mathbf{x}(k)$, then one can take constant control regime $L = L_k$, and solve Riccati equation only once. Therefore, in this case LQR has a fixed time window.

The discrete algebraic *dynamic* Riccati equation (DARE) as stated in (2.5c) is for *finite* horizon LQR, and substituting (2.5b) into (2.5c), one gets

$$P_{t-1} = A^T P_t A - A^T P_t B (B^T P_t B + R)^{-1} B^T P_t A + Q\,. \tag{2.6}$$

Note that the condition for definite convergence is $N \rightarrow \infty$, which also ensures that DARE converges to a unique solution P.

While for *infinite* horizon ($N \rightarrow \infty$) we substitute $P_{t-1} = P_t = P_\infty$ and solve an *algebraic* Riccati equation (ARE):

$$P_\infty = A^T P_\infty A - A^T P_\infty B (B^T P_\infty B + R)^{-1} B^T P_\infty A + Q\,, \tag{2.7}$$

where no recursion is needed, and P_∞ is applied in the control law for all steps. Note that for an infinite horizon, the terminal term in the LQR cost function (2.2) can not be reached, since $\Delta \mathbf{x}(\infty) \rightarrow 0$ in steady state conditions.

Then, substituting (2.3) into (2.1), one gets the closed-loop system

$$\Delta \mathbf{x}(k+1) = (A + BL) \Delta \mathbf{x}(k)\,, \tag{2.8}$$

which is stable [27] in case that the eigenvalues of $A_c = A + BL$ matrix are strictly inside the unit circle of the complex plane. The latter condition with infinite horizon guarantees the convergence of the state variables to their nominal values.

Finally, after calculating the gain matrix L, the LQR trajectory is calculated on-line for K steps simulation by Algorithm 1. Note that in step 3 of Algorithm 1, the control input is implemented on the plant model, which is represented by f_{plant}.

Algorithm 1 Calculation of the LQR trajectory $\Delta\mathbf{x}(k)$ for $k = 1, 2, \dots K$

Input: the matrices A, B, L, the initial state $\Delta\mathbf{x}(k)$

1: **for** $k = 1, 2, \dots K$ **do**
2: $\quad \Delta\mathbf{g}(k) \leftarrow L\Delta\mathbf{x}(k)$
3: $\quad \Delta\mathbf{x}(k+1) \leftarrow f_{\text{plant}}(\Delta\mathbf{x}(k), \Delta\mathbf{g}(k))$
4: $\quad k \leftarrow k + 1$
5: **end for**

2.2 Model Predictive Control (MPC)

Unlike LQR, the MPC is an optimization technique with a receding time window [13], which has the following controller form

$$\Delta\mathbf{g}(k) = f(\Delta\mathbf{x}(k))\,. \tag{2.9}$$

Note the difference between (2.9) and (2.3). Considering a control problem with linear model in (2.1) and a quadratic function in (2.2), the MPC controller (2.9) is actually a piecewise-affine linear state feedback controller, refer to [28]. The solution is calculated on-line, at each sample step. The resulting controlled system then:

$$\Delta\mathbf{x}(k+1) = A\Delta\mathbf{x}(k) + Bf(\Delta\mathbf{x}(k))\,. \tag{2.10}$$

In the following, the QP problem that MPC solves is formulated. The QP consists of the original system evolving in time, with prediction horizon N, and control horizon N_{m} which states that control variables exist for steps $k \le t \le k+N_{\text{m}}-1$, and after that for steps $k+N_{\text{m}} \le t \le k+N-1$ the control signals are assumed constant and equal to the last varying control signal of step $k + N_{\text{m}} - 1$. An MPC concept illustration, taken from [1], is shown in Fig. 2.1. Note that in the following, it is assumed that the prediction and control horizons are equal, i.e. $N_{\text{m}} = N$.

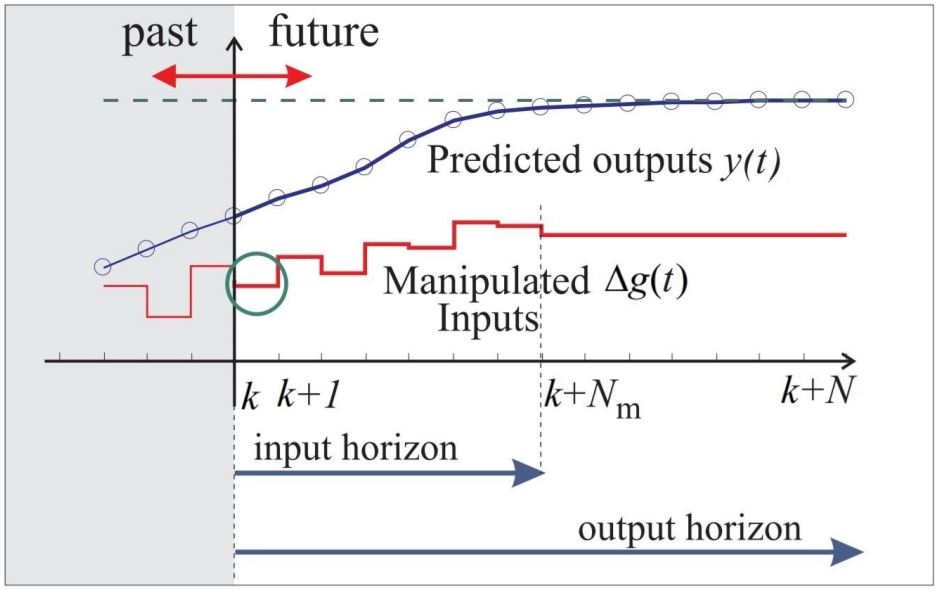


Figure 2.1: MPC concept illustration, taken from [1], where the outputs are equal to the states ($y(t) = \Delta x(t)$).

At each time step, the MPC solves an open loop control problem via QP, where each iteration depends upon the current initial state, therefore, it is a state feedback controller in a general point of view, or a closed loop type of controller, as seen in (2.9) and (2.10). Then, only the first control input $\Delta\mathbf{g}(k)$ is applied, since the next iteration performs a new calculation. This mechanism allows control designing for real systems, with disturbances and accounting for input and output noise.

Now we will see the QP derivation as it developed in [16].

The original system (2.1), (2.2), but with constraints, is re-written as:

$$J(\Delta\mathbf{x}(k)) = \min_{\mathbf{g}_N} \frac{1}{2}\left(\sum_{\tau=k}^{k+N-1} \Delta\mathbf{g}(\tau)^T R \Delta\mathbf{g}(\tau) + \sum_{\tau=k+1}^{k+N} \Delta\mathbf{x}(\tau)^T Q \Delta\mathbf{x}(\tau)\right)$$

s.t.

$$\begin{cases} \Delta\mathbf{x}(t+1) = A\Delta\mathbf{x}(t) + B\Delta\mathbf{g}(t)\,, \\ \Delta\mathbf{x}(t+1) \in \mathcal{X} = \{\Delta\mathbf{x} \in \mathbb{R}^n : F_x \Delta\mathbf{x}(t+1) \leq h_x\}, \\ \Delta\mathbf{g}(t) \in \mathcal{U} = \{\Delta\mathbf{g} \in \mathbb{R}^m : F_u \Delta\mathbf{g}(t) \leq h_u\}, \\ \forall t = k, k+1, \ldots, k+N-1\,, \end{cases} \tag{2.11}$$

where $F_x \in \mathbb{R}^{n_{h_x} \times n}, F_u \in \mathbb{R}^{n_{h_u} \times m}$ with n_{h_x}, n_{h_u} as the number of half-spaces defining state and control constraints, respectively. Note that the inequality

"$\leq$" or "$\geq$" is performed element-wisely.

One can augment state and control variables for all steps as follows

$$\Delta\mathbf{x}(t) \Rightarrow \mathbf{x}_N = \begin{bmatrix} \Delta\mathbf{x}(k+1) \\ \Delta\mathbf{x}(k+2) \\ \vdots \\ \Delta\mathbf{x}(k+N) \end{bmatrix}, \Delta\mathbf{g}(t) \Rightarrow \mathbf{g}_N = \begin{bmatrix} \Delta\mathbf{g}(k) \\ \Delta\mathbf{g}(k+1) \\ \vdots \\ \Delta\mathbf{g}(k+N-1) \end{bmatrix} \tag{2.12}$$

After augmenting all other matrices in (2.11), one gets the following formulation:

$$J(\Delta\mathbf{x}(k)) = \min_{\mathbf{g}_N} \frac{1}{2} \left(\mathbf{g}_N^T \mathbf{R}_N \mathbf{g}_N + \mathbf{x}_N^T \mathbf{Q}_N \mathbf{x}_N \right), \tag{2.13a}$$

s.t.

$$\mathbf{x}_N = \mathbf{A}_N \Delta\mathbf{x}(k) + \mathbf{B}_N \mathbf{g}_N \tag{2.13b}$$

$$F_{x,N}\mathbf{x}_N \leq \mathbf{g}_{x,N}, \tag{2.13c}$$

$$F_{u,N}\mathbf{g}_N \leq \mathbf{g}_{u,N}, \tag{2.13d}$$

with the augmented matrices

$$\mathbf{Q}_N = \begin{bmatrix} Q & 0 & \cdots & 0 \\ 0 & Q & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & Q \end{bmatrix}, \mathbf{R}_N = \begin{bmatrix} R & 0 & \cdots & 0 \\ 0 & R & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R \end{bmatrix}, \tag{2.14a}$$

$$\mathbf{A}_N = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}, \mathbf{B}_N = \begin{bmatrix} B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \cdots & B \end{bmatrix}, \tag{2.14b}$$

$$\mathbf{F}_{x,N} = \begin{bmatrix} F_x & 0 & \cdots & 0 \\ 0 & F_x & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & F_x \end{bmatrix}, \mathbf{g}_{x,N} = \begin{bmatrix} h_x \\ h_x \\ \vdots \\ h_x \end{bmatrix}, \tag{2.14c}$$

$$\mathbf{F}_{u,N} = \begin{bmatrix} F_u & 0 & \cdots & 0 \\ 0 & F_u & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & F_u \end{bmatrix}, \mathbf{g}_{u,N} = \begin{bmatrix} h_u \\ h_u \\ \vdots \\ h_u \end{bmatrix}. \tag{2.14d}$$

One can re-write the formulation in (2.13) and (2.14) to exclude $\mathbf{x}_N$ and letting it be dependent only on $\mathbf{g}_N$ and $\Delta\mathbf{x}(k)$, by substituting (2.13b)

into (2.13a) and (2.13c), i.e.

$$
\begin{aligned}
&J(\Delta\mathbf{x}(k)) = \min_{\mathbf{g}_N} \frac{1}{2}\left(\mathbf{g}_N^T\mathbf{H}_N\mathbf{g}_N + 2\Delta\mathbf{x}(k)^T\mathbf{F}_N\mathbf{g}_N + \cancel{\Delta\mathbf{x}(k)^T\mathbf{Y}_N\Delta\mathbf{x}(k)}\right),\\
&\text{s.t.}\\
&\mathbf{G}_N\mathbf{g}_N \leq \mathbf{E}_N\Delta\mathbf{x}(k) + \mathbf{S}_N\,,
\end{aligned}
\tag{2.15}
$$

where $\Delta\mathbf{x}(k)^T\mathbf{Y}_N\Delta\mathbf{x}(k)$ is a constant term, therefore, it can be neglected in the optimization, and

$$
\mathbf{G}_N = \begin{bmatrix}\mathbf{F}_{u,N}\\ \mathbf{F}_{x,N}\mathbf{B}_N\end{bmatrix}, \mathbf{E}_N = \begin{bmatrix}0\\ -\mathbf{F}_{x,N}\mathbf{A}_N\end{bmatrix}, \mathbf{S}_N = \begin{bmatrix}\mathbf{g}_{u,N}\\ \mathbf{g}_{x,N}\end{bmatrix}, \tag{2.16a}
$$

$$
\mathbf{H}_N = \mathbf{B}_N^T\mathbf{Q}_N\mathbf{B}_N + \mathbf{R}_N, \mathbf{F}_N = \mathbf{A}_N^T\mathbf{Q}_N\mathbf{B}_N, \mathbf{Y}_N = \mathbf{A}_N^T\mathbf{Q}_N\mathbf{A}_N. \tag{2.16b}
$$

Recall that (2.15) is an QP formulation of the MPC. For more information related to feasibility and stability of the MPC, the reader can refer to Appendix A.3.

Finally, the algorithm for on-line MPC computation is shown in Algorithm 2.

Algorithm 2 Calculation of the MPC trajectory $\Delta\mathbf{x}(k)$ for $k = 1, 2, \ldots K$

Input: the matrices A, B, Q, R, predictive horizon N, the initial state $\Delta\mathbf{x}(k)$, the sets $\mathcal{X}$ and $\mathcal{U}$

1: **for** $k = 1, 2, \ldots K$ **do**
2: $\quad\Delta\mathbf{g}(k) \leftarrow \text{MPC}(A, B, Q, R, N, \Delta\mathbf{x}(k), \mathcal{X}, \mathcal{U})$
3: $\quad\Delta\mathbf{x}(k+1) \leftarrow f_{\text{plant}}(\Delta\mathbf{x}(k), \Delta\mathbf{g}(k))$
4: $\quad k \leftarrow k + 1$
5: **end for**

Chapter 3

Uncertain store-and-forward model for urban road networks

In this chapter, the store-and-forward model that will be utilized for modeling urban road networks is first presented. Afterwards, the SF model is upgraded to include uncertainties. The state and control flow constraints are also described.

3.1 Main idea of the store-and-forward model

An illustration for a road link example with two intersections Ψ and Σ is shown in Fig. 3.2. A basic vehicle conservation equation, refer e.g. to [29], can be written as follows

$$x_z(t) - x_z(t_0) = \int_{t_0}^{t} (q_z(\tau) - u_z(\tau))d\tau, \tag{3.1}$$

where $q_z(t)$ [veh/s] and $u_z(t)$ [veh/s] are respectively the arrival (or the inflow) and departure (or the outflow) rates at time t for link z, and $x_z(t)$ [veh] denotes the queue of link z, i.e. the total number of vehicles in link z. This means that the change in the queue between t_0 and t is due to the integrated difference between the inflow of vehicles coming to the link and the outflow of vehicles getting out from the link.

Note that usually (3.1) is a non-linear equation, since the inflow and outflow have nonlinear dynamics which can change over each cycle and within cycles, e.g. once the light switches from green to red, the outflow drops to zero. The SF model, refer to e.g. [2], strives to simplify the dynamics of the system, by converting discrete-event modes, i.e. green and red durations (see

Fig. 3.1(a)), to a continuous-time mode over the cycle (see Fig. 3.1(b)), yet a discrete dynamic among cycles (see Fig. 3.1(c)), as follows:

$$x_z(k+1) - x_z(k) = \text{total inflow} - \text{total outflow} \quad (\text{during cycle } k) \quad (3.2)$$

where k is the cycle number, and $x_z(k)$ is the queue of link z at the beginning of cycle k. In other words, the SF model disregards the inner dynamics within a cycle of green phase and then red phase. On one hand, the latter might be a disadvantage of the model, while on the other hand, this results in a simplified model that can be utilized for control design.

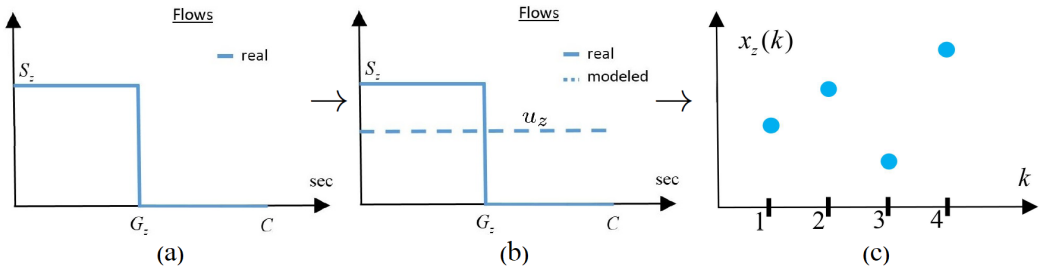


Figure 3.1: SF converts discrete-event modes (green and red durations) to a continuous-time mode over the cycle.

3.2 Store-and-forward dynamic equations

The store-and-forward dynamic equations are briefly introduced here following the description in [2, 21].

Let us consider an urban network which comprises several streets that cross at signalized intersections, $j \in J$, and links (approaches) $z \in Z$, see Fig. 3.2. For each signalized intersection j, I_j and O_j denote the sets of incoming and outgoing links, respectively. It is assumed that the offset, the cycle time C_j, and the total lost time L_j of intersection j are fixed; for simplicity, $C_j = C$ is assumed for all intersections $j \in J$. Furthermore, the signal control of intersection j is based on a fixed number of stages that belong to the set F_j, while v_z denotes the set of stages where link z has a right-of-way. Finally, the saturation flows S_z, $z \in I_j$, and the turning movement rates $t_{z,w}$, $z \in I_j$, $w \in O_j$, are assumed known and fixed.

Fig. 3.3 compares between *link outflow and queue* dynamics of the real process and the SF model over a cycle C for two conditions: under-saturated and over-saturated. Under-saturated condition is when vehicles exit the link in a maximal saturation flow during time period which is less than the given

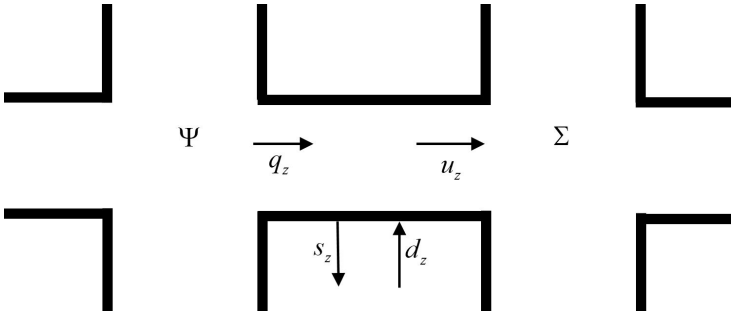


Figure 3.2: A schematic road link (based on [2]).

green duration period, which can be due to small queues in the link, for example. Over-saturated condition is when all vehicles exit a link in a maximal constant saturation flow, during the prescribed green period.

In Fig. 3.3(a), real outflows that would have been obtained at a stop-line of an approach are shown schematically for undersaturated and oversaturated conditions. The link outflow is a discrete-event process with two periods, i.e. green light $[0, G_z)$ and red light $[G_z, C]$. The outflow is assumed to be equal to the maximum flow value, i.e. the saturation flow S_z, during the whole or part of the green light for oversaturated or undersaturated condition, respectively, while during the red light the outflow is equal to zero. The corresponding saw-like queueing profiles are shown in Fig. 3.3(b), due to both the outflow and the inflow. The queue length decreases during the green light, while it increases during the red light.

Note that the dashed line in Fig. 3.3(b) is merely for illustrating linear behavior throughout the cycle, but it is not the actual model. The real model is discrete as shown in Fig. 3.1(c).

However, to simplify the modeling complexity, instead of modeling two levels of value for the outflow during a cycle, the SF model describes the average outflow [veh/s] over the cycle period (over the green light and also the red light). The averaged outflow for link z at cycle k, i.e. $u_z(k)$, is schematically shown in dashed line in Fig. 3.3(a), and calculated as follows:

$$u_z(k) = S_z G_z(k)/C\,, \tag{3.3}$$

where $G_z(k)$ [s] is the effective green duration of link z at cycle k, calculated as $G_z(k) = \sum_{i \in v_z} g_{j,i}(k)$, where $g_{j,i}(k)$ is the green duration of stage i at intersection j at cycle k. Since the average outflow is constant over the cycle, the corresponding queue length dynamic becomes linear, as shown in dashed line in Fig. 3.3(b).

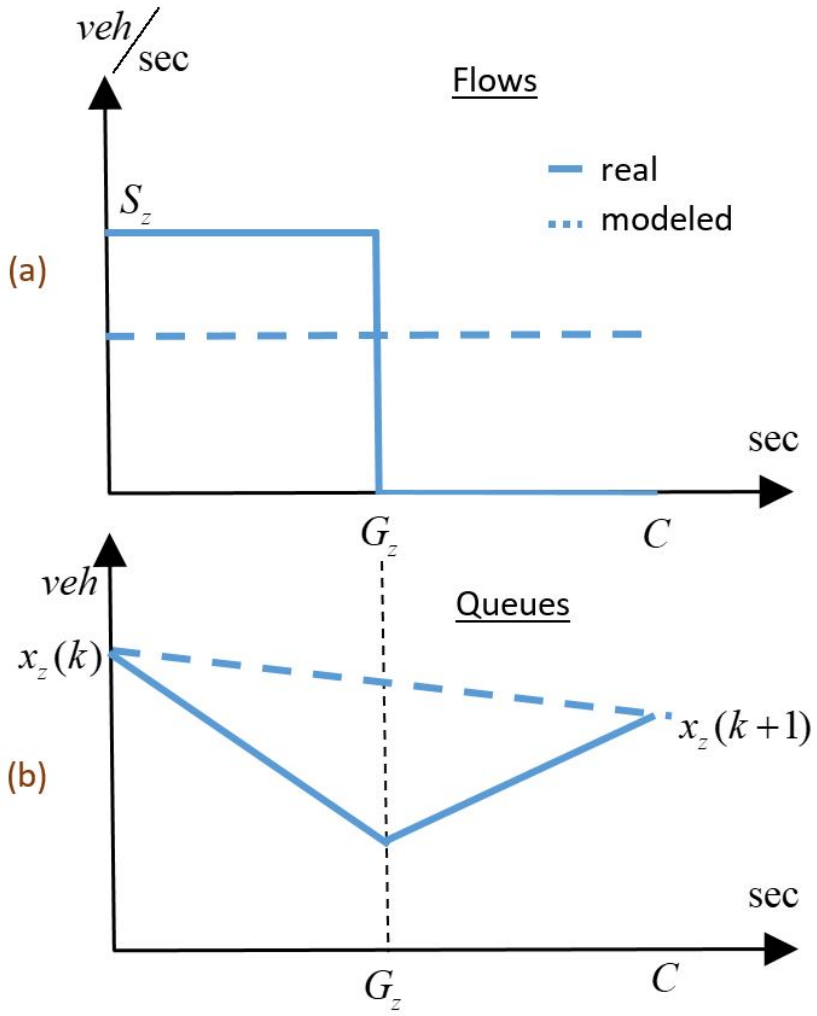


Figure 3.3: Real vs SF model comparison, in under- or over- saturated cases.

Hence, for each link (approach) $z \in Z$, the vehicle conservation equation (3.2) can be written as, see also Fig. 3.2,

$$x_z(k+1) = x_z(k) + \tilde{T}[q_z(k) - s_z(k) + d_z(k) - u_z(k)], \tag{3.4}$$

where the total inflow is $q_z(k) + d_z(k)$ and the total outflow is $s_z(k) + u_z(k)$.

$x_z(k)$ [veh] is the number of vehicles (or sometimes is referred as queue in the following) within link z at the beginning of cycle k, $d_z(k)$ [veh/s] and $s_z(k)$ [veh/s] are the traffic flow demand and the exit flow, respectively, and $q_z(k)$ and $u_z(k)$ are respectively the average inflow and outflow of link z over the period $[k\tilde{T}, (k+1)\tilde{T}]$, where $\tilde{T}$ is the control interval (which is defined to be equal to the cycle time duration C as in (3.3)) and k is the cycle time index $k = 1, 2, \dots K$. Following [21], the inflow to the link z is given by $q_z(k) = \sum_{w \in I_\Psi} t_{w,z} u_w(k)$, where $t_{w,z}$ with $w \in I_\Psi$ are the turning rates towards link z from the links that enter intersection Ψ. The exit flow $s_z(k)$ is determined by $s_z(k) = t_{z,0} q_z(k)$, where the exit rates $t_{z,0}$ are assumed to be known and constant.

It should be noted that the inflow $q_z(k)$ and the outflow $u_z(k)$ are under traffic signal control, while $d_z(k)$ and $s_z(k)$ are respectively the uncontrolled inflow and outflow for which vehicles enter or exit the link. E.g., due to lane changing, parking, entering/exiting to non-controlled links.

Now, substituting the above equations for inflow and exit flows and (3.3) into (3.4) for all z, then replacing the variable from $G_z(k)$ to $g_{j,i}(k)$, and finally organizing all resulting equations in one single vector-based equation leads to the following linear state-space form

$$\mathbf{x}(k+1) = A\mathbf{x}(k) + B\mathbf{g}(k) + T\mathbf{d}(k)\,, \tag{3.5}$$

where the state vector $\mathbf{x}(k) \in \mathbb{R}^n$ consists the number of vehicles x_z for all links, i.e. $\mathbf{x}(k) = \mathbf{Vec}_{j=1}^{n} x_j(k)$[1], the control vector $\mathbf{g}(k) \in \mathbb{R}^m$ consists of the green times g_j for all stages, i.e. $\mathbf{g}(k) = \mathbf{Vec}_{j=1}^{m} g_j(k)$, and the disturbance vector $\mathbf{d}(k) \in \mathbb{R}^n$ consists of the demand flows d_z of each link z, i.e. $\mathbf{d}(k) = \mathbf{Vec}_{j=1}^{n} d_j(k)$. Also $T \in \mathbb{R}^{n \times n}$ is a diagonal matrix, consists of $\tilde{T}$ at its diagonal. The matrix B is a constant matrix of appropriate dimensions containing the network characteristics, i.e. topology, saturation flows, and turning rates.

Note that the model (3.5) can be rewritten as a linear model with deviation state variables around an equilibrium point $(\mathbf{x}^{\mathrm{N}}, \mathbf{g}^{\mathrm{N}}, \mathbf{d}^{\mathrm{N}})$. It would be more reasonable to regulate the dynamic system around an equilibrium

[1]The symbol $\mathbf{Vec}_{j=1}^{m} z_j$ is defined as $\mathbf{Vec}_{j=1}^{m} z_j = [z_1, \cdots, z_m]'$.

point, which is not equal to zero, but corresponds to nominal queue lengths and green durations.

Therefore, the corresponding linear model for deviation variables is of the form

$$\Delta\mathbf{x}(k+1) = A\Delta\mathbf{x}(k) + B\Delta\mathbf{g}(k) + T\Delta\mathbf{d}(k)\,, \tag{3.6}$$

where $\Delta\mathbf{x}(k) = \mathbf{x}(k) - \mathbf{x}^{\mathrm{N}}$, $\Delta\mathbf{g}(k) = \mathbf{g}(k) - \mathbf{g}^{\mathrm{N}}$, $\Delta\mathbf{d} = \mathbf{d}(k) - \mathbf{d}^{\mathrm{N}}$. See Section A.1 for more details. $\mathbf{x}^{\mathrm{N}}$ is free to choose[2], however, $\mathbf{g}^{\mathrm{N}}$ and $\mathbf{d}^{\mathrm{N}}$ are functionally dependent on each other. Substituting nominal values in the conservation formula (3.5), i.e. $\mathbf{g}(k) = \mathbf{g}^{\mathrm{N}}$, $\mathbf{d}(k) = \mathbf{d}^{\mathrm{N}}$, $\mathbf{x}(k+1) = \mathbf{x}(k) = \mathbf{x}^{\mathrm{N}}$, and $A = I$ according to (3.5), one gets

$$B\mathbf{g}^{\mathrm{N}} + T\mathbf{d}^{\mathrm{N}} = 0\,. \tag{3.7}$$

Note that it is usually assumed that the deviation in demand is equal to zero for control design, i.e. $\Delta\mathbf{d}(k) = 0$, or in other words the demand stays constant and equal to $\mathbf{d}^{\mathrm{N}}$. Hence, eventually (3.6) becomes the Nominal model

$$\Delta\mathbf{x}(k+1) = A\Delta\mathbf{x}(k) + B\Delta\mathbf{g}(k). \tag{3.8}$$

However, if we consider disturbances in demand that deviate from $\mathbf{d}^{\mathrm{N}}$, then $\Delta\mathbf{d}(k) \neq 0$ and (3.6) holds.

3.3 Modeling uncertainty in the SF model

Urban road networks in reality include many uncertainties. Therefore, in relation to our problem formulation, some of the parameters that characterize the network should be assumed as uncertain parameters. The SF model utilizes the following parameters for each link in a network:

- Saturation flows vector S_z [veh/h], which influences the outflow $u_z(t)$ [veh/h],
- Turning rates (matrix) $t_{w,z}$ [veh/h] towards link z from other links w, which influence the inflow $q_z(t)$ [veh/h],
- Exit rates vector $t_{z,0}$ [veh/h], which influences the exit flows $s_z(k)$ [veh/h].

[2] One can choose $\mathbf{x}^{\mathrm{N}}$ as an average queue length per cycle in a steady-state condition, see [30].

- Demand flows vector d_z [veh/h], which should be equal to the nominal demands d^{N}, which defines the nominal green durations g^{N}, see (3.7).

In previous works, e.g. [2], [4], all above parameters are assumed ideally to be fixed and known, but in reality they are uncertain parameters that might be unknown and/or changing over time.

The parameters are within the group: S_z, $t_{w,z}$ and $t_{z,0}$. The uncertainties in the group are embedded in the B matrix.

3.3.1 Modeling uncertainty in S_z, $t_{w,z}$, and $t_{z,0}$

The parameter uncertainties S_z, $t_{w,z}$, and $t_{z,0}$ would affect only the B matrix, and eventually the control performances as they are not designed to compensate uncertainties.

The parameter uncertainties are integrated in the SF model by introducing uncertainty sets. It is assumed that the saturation flow S_z, the turning rate $t_{w,z}$, and the exit rate $t_{z,0}$ of link $z \in Z$, belong respectively to interval sets

$$\theta_z = \{S_z \,|\, S_{z,\min} \leq S_z(k) \leq S_{z,\max}\} \,, \tag{3.9}$$

$$\pi_z = \{t_{w,z} \,|\, t_{w,z,\min} \leq t_{w,z}(k) \leq t_{w,z,\max}\} \,, \tag{3.10}$$

$$\gamma_z = \{t_{z,0} \,|\, t_{z,0,\min} \leq t_{z,0}(k) \leq t_{z,0,\max}\} \,. \tag{3.11}$$

Substituting (3.3), $q_z(k) = \sum_{w \in I_M} t_{w,z} u_w(k)$, and $s_z(k) = t_{z,0} q_z(k)$ in (3.4) for all links, and including the uncertainty sets (3.9)–(3.11), one gets the SF model in the form of a discrete-time uncertain linear model as follows

$$\Delta \mathbf{x}(k+1) = A \Delta \mathbf{x}(k) + B(\boldsymbol{\beta}) \Delta \mathbf{g}(k) + T \Delta \mathbf{d}(k) \,. \tag{3.12}$$

The uncertain matrix $B(\boldsymbol{\beta})$ depends on the uncertain saturation flows, the turning and exit rates parameters, i.e. $\boldsymbol{\beta} = [\mathbf{S}, \mathbf{t_w}, \mathbf{t_0}]' \in \Sigma$, where $\mathbf{S} = \mathbf{Vec}_{j=1}^{J} S_j$, $\mathbf{t_w} = \mathbf{Vec}_{j=1}^{J} t_{w,j}$, $\mathbf{t_0} = \mathbf{Vec}_{j=1}^{J} t_{j,0}$.

Recall the nominal matrix B is shown in Appendix A.1.

3.4 Traffic flow model constraints

State and control constraints are imposed on each link and individual isolated intersection in the network, [2].

State constraints

Queue lengths are subject to lower and upper bound constraints, i.e.

$$0 \leq x_z \leq x_z^{\max} \qquad \forall z \in Z, \tag{3.13}$$

where $x_z^{\max}$ [veh] is the maximum admissible queue length.

Control constraints

The sum of the green durations and the total lost time must be equal to (or less than) the cycle time, i.e.

$$\sum_{i \in F_j} g_{j,i} + L_j = (\text{or} \leq) C \qquad \forall j \in J, \tag{3.14}$$

must hold at each intersection j, where $g_{j,i}$ is the green time of stage i at intersection j.

It is worth mentioning that the equality is more restrictive compared with the inequality, since it constrains the total sum of the control inputs to be constant throughout all cycles, while the inequality allows this sum to change from one cycle to another, implying a time varying cycle time.

Another control constraints are given as follows

$$g_{j,i}^{\min} \leq g_{j,i} \leq g_{j,i}^{\max}, \tag{3.15}$$

which are the upper and lower bound constraints, i.e. the green light durations are in some minimum-maximum ranges.

Following the linear model for deviation variables, see (3.6), the state constraints (3.13) and the control constraints (3.15) can be re-written respectively as

$$-x_z^{\mathrm{N}} < \Delta x_z < x_z^{\max} - x_z^{\mathrm{N}}\,, \tag{3.16}$$

$$g_{j,i}^{\min} - g_{j,i}^{\mathrm{N}} \leq \Delta g_{j,i} \leq g_{j,i}^{\max} - g_{j,i}^{\mathrm{N}}\,, \tag{3.17}$$

and if the sum of green duration constraints equation (3.14) is subtracted from the same equation (3.14) applied to nominal values $\sum_{i \in F_j} g_{j,i}^{\mathrm{N}} + L_j = C$, then one gets

$$\sum_{i \in F_j} \Delta g_{j,i} = (\text{or} \leq) 0\,. \tag{3.18}$$

For the purpose of simple representation of these constraints in the following chapters, the vector notation is utilized as follows:

$$\mathbf{x}^{\max} = \mathbf{Vec}_{z=1}^{n} x_z^{\max}\,, \quad \mathbf{g}^{\max} = \mathbf{Vec}_{j=1}^{m} g_j^{\max}\,, \quad \mathbf{g}^{\min} = \mathbf{Vec}_{j=1}^{m} g_j^{\min}\,, \tag{3.19}$$

$$\mathbf{x}^{\mathrm{N}} = \mathbf{Vec}_{z=1}^{n} x_z^{\mathrm{N}}\,, \quad \mathbf{g}^{\mathrm{N}} = \mathbf{Vec}_{j=1}^{m} g_j^{N}\,, \quad \mathbf{d}^{\mathrm{N}} = \mathbf{Vec}_{z=1}^{n} d_z^{\mathrm{N}}\,. \tag{3.20}$$

Chapter 4

Interpolation-based control for traffic signals

Interpolating Control (IC) [17] is a regulating control method, which can handle model uncertainties and allow inclusion of state and control constraints. It also guarantees recursive feasibility and asymptotic stability, for all feasible initial conditions. Recently, a new simple version of IC, using saturated control inputs and ellipsoidal feasible sets, called Simple Interpolating Control (SIC) was presented in [31]. Unlike IC, SIC does not need to solve an LP at each time step, which is less computationally expensive than IC.

In this thesis, IC and SIC methods will be implemented to design traffic signal control. Both methods can handle uncertain and/or time-varying linear discrete time systems, i.e. they can be utilized to design robust traffic signal controllers.

Both interpolation-based design methods can be seen as alternatives to optimization-based control schemes such as Model Predictive Control. The design methods are suitable for problems which finding the optimal solution requires calculations, where IC and SIC can provide straightforward suboptimal solutions. In the following sections, a summary of the methods proposed in [16,17,31–33] is presented.

4.1 Maximal admissible and controlled invariant sets

Let us consider the following uncertain and/or time-varying linear discrete-time system:

$$\Delta\mathbf{x}(k+1) = A(k)\Delta\mathbf{x}(k) + B(k)\Delta\mathbf{g}(k) + T\Delta\mathbf{d}(k)\,, \tag{4.1}$$

where $\Delta\mathbf{x}(k) \in \mathbb{R}^n$ is the state and $\Delta\mathbf{g}(k) \in \mathbb{R}^m$ is the control input, and $\mathbf{d}(k) \in \mathbb{R}^n$ is the disturbance. The matrices $A(k) \in \mathbb{R}^{n\times n}$ and $B(k) \in \mathbb{R}^{n\times m}$ are given with polytopic uncertainty as follows

$$A(k) = \sum_{i=1}^{s} \alpha_i(k) A_i, \qquad B(k) = \sum_{i=1}^{s} \alpha_i(k) B_i, \tag{4.2a}$$

$$\sum_{i=1}^{s} \alpha_i(k) = 1,\ \alpha_i(k) \geq 0,\ \forall i = 1, \ldots, s, \tag{4.2b}$$

where $\alpha_i(k)$ is unknown and time-varying. The state, the control, and the disturbance are subject to the following polytopic constraints:

$$\Delta\mathbf{x}(k) \in \mathcal{X} = \{\Delta\mathbf{x} \in \mathbb{R}^n : F_x \Delta\mathbf{x}(k) \leq h_x\},\ h_x > 0, \tag{4.3a}$$

$$\Delta\mathbf{g}(k) \in \mathcal{U} = \{\Delta\mathbf{g} \in \mathbb{R}^m : F_u \Delta\mathbf{g}(k) \leq h_u\},\ h_u > 0, \tag{4.3b}$$

$$\Delta\mathbf{d}(k) \in \mathcal{W} = \{\Delta\mathbf{d} \in \mathbb{R}^n : F_d \Delta\mathbf{d}(k) \leq h_d\},\ h_d > 0, \tag{4.3c}$$

where $F_x \in \mathbb{R}^{n_{h_x}\times n}, F_u \in \mathbb{R}^{n_{h_u}\times m}, F_d \in \mathbb{R}^{n_{h_d}\times n}$ with $n_{h_x}, n_{h_u}, n_{h_d}$ as the number of half-spaces defining state, control, and disturbance constraints, respectively.

In our traffic signal problem, we represent state and control constraints (3.16)–(3.18) in the form (4.3):

$$F_x = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ \hdashline -1 & 0 & \cdots & 0 \\ 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & -1 \end{bmatrix}, h_x = \begin{bmatrix} \\ \mathbf{x}^{\max} - \mathbf{x}^{\mathrm{N}} \\ \hdashline \\ -\mathbf{x}^{\mathrm{N}} \\ \\ \end{bmatrix}, \tag{4.4a}$$

$$F_u = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ \hdashline -1 & 0 & \cdots & 0 \\ 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & -1 \\ \hdashline 1 & 1 & \cdots & 1 \end{bmatrix}, h_u = \begin{bmatrix} \\ \mathbf{g}^{\max} - \mathbf{g}^{\mathrm{N}} \\ \hdashline \\ \mathbf{g}^{\mathrm{N}} - \mathbf{g}^{\min} \\ \hdashline 0 \end{bmatrix}. \tag{4.4b}$$

Let us denote nx as the total number of states (total links in all intersections), nu as the total number of control inputs (total phases in all intersections), and $njuncs$ as the number of intersections. Then, the number of state constraints is $nfx = 2nx$, and the number of control constraints is $nfu = 2nu + njuncs$.

Remark. *The form given in (4.4b) considers one signalized intersection, as the last row in F_u is only for one intersection. If more intersections are considered, then the number of rows is equal to the number of intersections. For each intersection there will be ones in columns' elements which correspond to the control phases in $\Delta\mathbf{g}$, the other elements of the row will be zeros.*

Assuming a robustly stabilizing control law, such as

$$\Delta\mathbf{g}_0(k) = L\Delta\mathbf{x}(k)\,, \tag{4.5}$$

where $L \in \mathbb{R}^{m\times n}$, one can combine control and state constraints into one polytopic set:

$$\mathcal{X} = \{\Delta\mathbf{x}(k) \in \mathbb{R}^n : F_c\Delta\mathbf{x}(k) \leq h_c\}\,, \tag{4.6}$$

where $F_c = \begin{bmatrix} F_x \\ F_uL \end{bmatrix} \in \mathbb{R}^{(n_{h_x}+n_{h_u})\times n}$ and $h_c = \begin{bmatrix} h_x \\ h_u \end{bmatrix} \in \mathbb{R}^{(n_{h_x}+n_{h_u})\times 1}$.

Applying the controller (4.5) to the system dynamics (4.1) with the robust uncertainty (4.2), the closed loop dynamics become

$$\Delta\mathbf{x}(k+1) = A_c(k)\Delta\mathbf{x}(k) + T\Delta\mathbf{d}(k)\,, \tag{4.7}$$

where

$$A_c(k) = A(k) + B(k)L\,, \quad A_c(k) \in \mathbb{R}^{n\times n}\,. \tag{4.8}$$

Another form for (4.8) is called convex hull, representing convex combination of some given matrices:

$$A_c(k) = A(k) + B(k)L = \mathrm{Conv}\{A_{ci}\}\,, \tag{4.9}$$

with

$$A_{ci} = A_i + B_iL\,\forall i = 1,\ldots,s\,. \tag{4.10}$$

While imposing input and state constraints with L as some given gain matrix, one can calculate $\Omega_{\max}$, a *maximal admissible invariant set* (MAS) for which any starting point inside the set will converge to the origin according to the control law (4.5), without violating the state and control constraints.

Algorithm 3 Calculation of the maximal invariant set $\Omega_{\max}$

Input: the matrix A, the sets $\mathcal{X}$ and $\mathcal{W}$

1: $\Omega_0 \leftarrow \mathcal{X},\ \Omega_1 \leftarrow \emptyset,\ k \leftarrow 0$
2: **while** $\Omega_{k+1} \neq \Omega_k$ **do**
3: $\quad \Omega_{k+1} \leftarrow \Omega_k \cap \mathrm{Pre}(\Omega_k)$
4: $\quad k \leftarrow k+1$
5: **end while**
6: $\Omega_{\max} \leftarrow \Omega_k$

This calculation is shown in Algorithm 3, taken from [16]. This means that we start from the set of state constraints $\mathcal{X}$, see step 1 of Algorithm 3, and then apply (4.6) constraints on next step $k+1$ in step 3 of Algorithm 3

$$\mathrm{Pre}(\Omega_k) = \left\{ \begin{array}{l} \Delta\mathbf{x}(k) \in \mathbb{R}^n :\ F_c A_{ci} \Delta\mathbf{x}(k) \leq h_c - T \max\limits_{\Delta\mathbf{d}(k)\in\mathcal{W}} \{F_c \Delta\mathbf{d}(k)\} \\ \\ \forall i = 1, \ldots, s\,, \end{array} \right\} \tag{4.11}$$

according to the system dynamics (4.7). Then, one gets a new invariant set which satisfies both current step and next step state constraints. This is done by calculating the intersection of these two sets, as shown in step 3 of Algorithm 3.

In the next iteration, the same rule of intersecting current and next constraints sets is applied, where at each time the current set $\{F_c, h_c\}$ is updated. The iteration loop continues until the maximal set is determined, see step 6 of Algorithm 3, i.e.

$$\Omega_{\max} = \{\Delta\mathbf{x}(k) \in \mathbb{R}^n :\ F_0 \Delta\mathbf{x}(k) \leq h_0\}\,, \tag{4.12}$$

where no change occurs when going from the current set to the next one, see step 2.

Afterwards, one can calculate C_N, a *controlled invariant set*, using Algorithm 4, taken from [16], for some specific size of a receding window, i.e. a set of states for which there exists an admissible control sequence such that states starting inside the set, will be steered into $\Omega_{\max}$ in no more than N steps.

Similar to the calculation of the MAS, we start from some starting set, this time from $\Omega_{\max}$, and also apply next step constraints. Only this time we want to find the set which gives the maximum possible control set $\Delta\mathbf{g}(k)$. Hence, in the next step, the set is determined as shown in step 3 of Algorithm 4

Algorithm 4 Calculation of the N-step controlled set $C_N(\Omega)$

Input: the matrices A, B, the sets $\mathcal{X}$, $\mathcal{U}$, $\mathcal{W}$ and the target set Ω

1: $C_0 \leftarrow \Omega,\ k \leftarrow 0$
2: **while** $k < N$ **do**
3: $\quad C_{k+1} \leftarrow \mathrm{Pre}(C_k) \cap \mathcal{X}$
4: $\quad k \leftarrow k + 1$
5: **end while**

with

$$\mathrm{Pre}(C_k) = \left\{ \begin{array}{l} \Delta\mathbf{x}(k) \in \mathbb{R}^n : \exists \Delta\mathbf{g}(k) \in \mathcal{U} \\ \text{s.t.} \quad F_k\big(A_i\Delta\mathbf{x}(k) + B_i\Delta\mathbf{g}(k)\big) \leq h_k - T \max\limits_{\Delta\mathbf{d}(k)\in\mathcal{W}} \{F_k\Delta\mathbf{d}(k)\} \\ \forall i = 1, \ldots, s\,, \end{array} \right\} \tag{4.13}$$

according to the uncertain linear system dynamics (4.1).

Note that calculating $\mathrm{Pre}(C_k)$, in step 3 of Algorithm 4, requires projection of $(\Delta\mathbf{x}(k), \Delta\mathbf{g}(k))$ space to $\Delta\mathbf{x}(k)$ state space only. And after that, calculation of next C_{k+1} is the intersection of next steps' space and the original constraints, since we are interested in the largest set possible to steer from C_k to C_{k+1} using some feasible control $\Delta\mathbf{g}(k) \in \mathcal{U}$.

After N predefined steps we retrieve the final controlled invariant set $C_N = \{\Delta\mathbf{x}(k) \in \mathbb{R}^n : \ F_1\Delta\mathbf{x} \leq h_1\}$. If we are interested in the maximal controlled invariant set $C_{\max}$, then we change the while loop in step 2 to $C_{k+1} \neq C_k$, similarly to Algorithm 3.

Obviously $\Omega_{\max} \subset C_N$. The $\Omega_{\max}$ is the inner set and designed for performance (4.5), while C_N is the outer set and designed to enlarge the admissible set.

4.2 Vertex Control

This control uses the outer set we calculated in Algorithm 4, to regulate any initial state in this set to the origin, see [34]. It uses the admissible control signal $\Delta\mathbf{g}(k) \in \mathcal{U}$ of the 1^{st} step of C_N for that. We solve the following LP problem [35], which finds the most contractive value λ and an admissible

control $\Delta\mathbf{g}_l$ for each vertex $\Delta\mathbf{x}_l$ of C_N's vertices set $\{\Delta\mathbf{x}_l\}$:

$$\min_{\Delta\mathbf{g}_l,\lambda}\{\lambda\} \quad \text{s.t.} \begin{cases} F_1(A_i\Delta\mathbf{x}_l + B_i\Delta\mathbf{g}_l) \leq \lambda h_1 - T \max\limits_{\Delta\mathbf{d}(k)\in\mathcal{W}}\{F_1\Delta\mathbf{d}(k)\}\,, \\ F_u\Delta\mathbf{g}_l \leq h_u\,, \\ \lambda \geq 0\,, \\ \forall i = 1,\ldots,s. \end{cases} \tag{4.14}$$

This LP problem can be solved once, for given $\{\Delta\mathbf{x}_l\}$ set of C_N vertices, obtaining the most contractive control signal $\{\Delta\mathbf{g}_l\}$ for the vertex LP problem.

Once we have a control signal $\{\Delta\mathbf{g}_l\}$, we solve vertex LP which finds the strongest control that brings the current state to the origin, by finding the minimal linear convex combination of the current state from the $\{\Delta\mathbf{x}_l\}$ set:

$$\min_{\gamma_l}\left(\sum_{l=1}^{p}\gamma_l\right) \quad \text{s.t.} \begin{cases} \sum\limits_{l=1}^{p}\gamma_l\Delta\mathbf{x}_l = \Delta\mathbf{x}(k)\,, \\ \sum\limits_{l=1}^{p}\gamma_l \leq 1\,, \\ \gamma_l \geq 0\ \forall l = 1,\ldots,p\,. \end{cases} \tag{4.15}$$

Note that the reason that the sum of coefficients of the combination might be smaller than 1, is due to informal including of the origin as an additional vertex, to take some portion of the total $\sum\limits_{l=1}^{p}\gamma_l$.

Now the control law is the same convex combination only of control signal $\{\Delta\mathbf{g}_l\}$ instead the state $\{\Delta\mathbf{x}_l\}$, i.e.

$$\Delta\mathbf{g}(k) = \sum_{l=1}^{p}\gamma_l\Delta\mathbf{g}_l\,. \tag{4.16}$$

Next step is to calculate the state at the next time step, $\Delta\mathbf{x}(k+1) = f_{\text{plant}}(\Delta\mathbf{x}(k), \Delta\mathbf{g}(k))$, using the control we found. Next iterations can evolve by solving the LP in (4.15).

Another option is to apply "Minkowski norm minimization controller" [35]. I.e. solving only the 1st LP in (4.14) each iteration, where one solves for the current state instead of for each of C_N's vertices, i.e. $\Delta\mathbf{x}_l = \Delta\mathbf{x}(k)$, and obtaining next state $\Delta\mathbf{x}(k+1) = f_{\text{plant}}(\Delta\mathbf{x}(k), \Delta\mathbf{g}(k))$, by using the control signal $\Delta\mathbf{g}_l$ retrieved from this LP. This LP replaces (4.15), but needs solving an additional LP to guarantee uniqueness. On the other hand, it is more aggressive than the vertex method, since it finds most contractive control each step, while the vertex control uses only the most contractive control for the first step, on the border of C_N, and gets weaker control as it goes farther from it.

4.3 Interpolating Control (IC) method

The IC method is similar to the Vertex Control, refer to Section 4.2, with one difference only: the convex combination now is not from the vertex set and the origin, but instead from some state inside the outer set $\Delta\mathbf{x}_v(k) \in C_N/\Omega_{\max}$ and some state inside the inner set $\Delta\mathbf{x}_0 \in \Omega_{\max}$. In other words, the vertex control is the outer control of IC. In IC method unlike in vertex control, we solve only the LP in (4.15), while for obtaining the convex combination of the vertex set and $\Delta\mathbf{x}_0$ we solve the following nonlinear problem:

$$\min_{c(k),\Delta\mathbf{x}_v,\Delta\mathbf{x}_0} c(k) \quad \text{s.t.} \begin{cases} F_1\Delta\mathbf{x}_v \leq h_1\,, \\ F_0\Delta\mathbf{x}_0 \leq h_0\,, \\ c(k)\Delta\mathbf{x}_v(k) + (1-c(k))\Delta\mathbf{x}_0(k) = \Delta\mathbf{x}(k)\,, \\ 0 \leq c(k) \leq 1\,, \end{cases} \tag{4.17}$$

which can be rewritten as the following LP, by applying $r_v(k) = c(k)\Delta\mathbf{x}_v(k)$,

$$\min_{c(k),r_v(k)} c(k) \quad \text{s.t.} \begin{cases} F_1 r_v(k) \leq c(k)h_1\,, \\ F_0(\Delta\mathbf{x}(k) - r_v(k)) \leq (1-c(k))h_0\,, \\ 0 \leq c(k) \leq 1\,. \end{cases} \tag{4.18}$$

Using the solution, we retrieve the convex combination of the current state:

$$\Delta\mathbf{x}(k) = \underbrace{c(k)\Delta\mathbf{x}_v(k)}_{r_v(k)} + (1-c(k))\Delta\mathbf{x}_0(k)\,. \tag{4.19}$$

$\Delta\mathbf{g}_v(k)$ can be computed (i) by first solving the vertex control LP, which finds the convex combination of $\Delta\mathbf{x}_v(k)$ by C_N's vertices (4.15), and then by using calculated combination, one can get $\Delta\mathbf{g}_v(k)$; or (ii) by finding the most contractive control signal using the LP problem (4.14), where we replace all vertices $\Delta\mathbf{x}_l$ with the current $\Delta\mathbf{x}_v(k)$, and the output signal $\Delta\mathbf{g}_l$ is interpreted as the $\Delta\mathbf{g}_v(k)$ we look for, following the "Minkowski norm minimization controller". It should be noted that $\Delta\mathbf{x}_0(k)$ performs according to (4.5).

The outer control $\Delta\mathbf{g}_v(k)$ is fully exploited only on the border of C_N. Note that the time to regulate the plant state to the origin would be much longer for the outer controller compared to the inner controller, i.e. a more aggressive local controller $\Delta\mathbf{g}_0(k)$ operating near the origin in $\Omega_{\max}$. Therefore, it is preferable that the control for an initial state in the outer region should be eventually switched to the inner controller. The outer and inner controllers are interpolated by applying a convex combination as follows,

$$\Delta\mathbf{g}(k) = c(k)\Delta\mathbf{g}_v(k) + (1-c(k))\Delta\mathbf{g}_0(k)\,, \tag{4.20}$$

where $0 \leq c(k) \leq 1$ is the interpolating coefficient, which is minimized in order to have the state as close as possible to the $\Omega_{\max}$, implying a better control performance. The interpolating coefficient is calculated by solving an LP. The inner controller $\Delta\mathbf{g}_0$ is assumed to be chosen a priori, e.g. (4.5), and the outer controller can be found by solving an LP each sample step. Hence, at each time step, two LPs should be solved. For a detailed information, the reader can refer to [17].

Finally, the algorithm to calculate the IC trajectory $\Delta\mathbf{x}(k)$ using Minkowski-norm method, see Section 4.2, is Algorithm 5.

Algorithm 5 Calculation of the IC trajectory $\Delta\mathbf{x}(k)$ using Minkowski-norm method, for $k = 1, 2, \dots K$

Input: the matrices A, B, L, the sets $\Omega_{\max}, C_N, \mathcal{U}, \mathcal{W}$, and the initial state $\Delta\mathbf{x}(k)$.

for $k = 1, 2, \dots K$ **do**
  $c(k), r_v(k) \leftarrow \text{LP}(4.18)(\Omega_{\max}, C_N, \Delta\mathbf{x}(k))$
  $\Delta\mathbf{x}_l \leftarrow r_v(k)/c(k)$
  $\Delta\mathbf{g}_l \leftarrow \text{LP}(4.14)(A, B, C_N, \mathcal{U}, \mathcal{W}, \Delta\mathbf{x}_l)$
  $\Delta\mathbf{g}_v(k) \leftarrow \Delta\mathbf{g}_l$
  $\Delta\mathbf{g}_0(k) \leftarrow L(\Delta\mathbf{x}(k) - r_v)/(1 - c(k))$
  $\Delta\mathbf{g}(k) \leftarrow c(k)\Delta\mathbf{g}_v(k) + (1 - c(k))\Delta\mathbf{g}_0(k)$
  $\Delta\mathbf{x}(k+1) \leftarrow f_{\text{plant}}(\Delta\mathbf{x}(k), \Delta\mathbf{g}(k))$
  $k \leftarrow k + 1$
end for

4.4 Improved IC

In this special method [18] of solving IC problem, we avoid the need to calculate the projection in Algorithm 4. Projection is a heavy computational process, which gets heavier as the number of states and control inputs increase. Sometimes it can not even yield any solution, as happened in some of the numerical runs in Chapter 8. Improved IC (IIC) can bypass the projection process, where the inner and outer invariant sets were calculated using the constraints. Instead, IIC imposes constraints explicitly in the main LP problem of IC (4.18). In other words, instead of $F_1 r_v(k) \leq c(k)h_1$, we use N

steps constraints which comprise the C_N control invariant set:

$$\min_{c(k),r_v(k),\mathbf{V}_N} \{c(k)\} \quad \text{s.t.} \begin{cases} F_N \begin{bmatrix} r_v(k) \\ \mathbf{V}_N \end{bmatrix} \leq c(k) h_N \,, \\ F_0(\Delta\mathbf{x}(k) - r_v(k)) \leq (1 - c(k)) h_0 \,, \\ 0 \leq c(k) \leq 1 \,, \end{cases} \tag{4.21}$$

where for a given current state $\Delta\mathbf{x}(k)$, the C_N's constraints are:

$$C_N = \begin{cases} A_i^N \Delta\mathbf{x}(k) + A_i^{N-1} B_i \Delta\mathbf{g}(k) + A_i^{N-1} T \Delta\mathbf{d}(k) + A_i^{N-2} B_i \Delta\mathbf{g}(k+1) \\ + A_i^{N-2} T \Delta\mathbf{d}(k+1) + \ldots + B_i \Delta\mathbf{g}(k+N-1) + T \Delta\mathbf{d}(k+N-1) \in \Omega_{\max} \\ A_i^{N-1} \Delta\mathbf{x}(k) + A_i^{N-2} B_i \Delta\mathbf{g}(k) + A_i^{N-2} T \Delta\mathbf{d}(k) + \ldots \\ + B_i \Delta\mathbf{g}(k+N-2) + T \Delta\mathbf{d}(k+N-2) \in \mathcal{X} \\ A_i^{N-2} \Delta\mathbf{x}(k) + A_i^{N-3} B_i \Delta\mathbf{g}(k) + A_i^{N-3} T \Delta\mathbf{d}(k) + \ldots \\ + B_i \Delta\mathbf{g}(k+N-3) + T \Delta\mathbf{d}(k+N-3) \in \mathcal{X} \\ \vdots \\ A_i \Delta\mathbf{x}(k) + B_i \Delta\mathbf{g}(k) + T \Delta\mathbf{d}(k) \in \mathcal{X} \\ \Delta\mathbf{g}(k) \in \mathcal{U}, \ \Delta\mathbf{g}(k+1) \in \mathcal{U}, \ \ldots, \ \Delta\mathbf{g}(k+N-1) \in \mathcal{U} \\ \Delta\mathbf{d}(k) \in \mathcal{W}, \ \Delta\mathbf{d}(k+1) \in \mathcal{W}, \ \ldots, \ \Delta\mathbf{d}(k+N-1) \in \mathcal{W} \\ \forall i = 1, 2, \ldots, s \end{cases} \tag{4.22}$$

represented in matrix form:

$$F_N = \left[\begin{array}{c|cccc} F_0A_i^N & F_0A_i^{N-1}B_i & F_0A_i^{N-2}B_i & \cdots & F_0B_i \\ F_xA_i^{N-1} & F_xA_i^{N-2}B_i & \cdots & F_xB_i & 0 \\ F_xA_i^{N-2} & F_xA_i^{N-3}B_i & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ F_xA_i & F_xB_i & 0 & \cdots & 0 \\ \hline 0 & F_u & 0 & \cdots & 0 \\ 0 & 0 & F_u & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & F_u \end{array}\right], \tag{4.23a}$$

$$h_N = \left[\begin{array}{c} h_0 - \max\limits_{\Delta\mathbf{d}(k)\in\mathcal{W}} \{F_0T\left(A_i^{N-1}\Delta\mathbf{d}(k) - A_i^{N-2}\Delta\mathbf{d}(k+1)\cdots - \Delta\mathbf{d}(k+N-1)\right)\} \\ h_x - \max\limits_{\Delta\mathbf{d}(k)\in\mathcal{W}} \{F_xT\left(A_i^{N-2}\Delta\mathbf{d}(k) - A_i^{N-3}\Delta\mathbf{d}(k+1)\cdots - \Delta\mathbf{d}(k+N-2)\right)\} \\ h_x - \max\limits_{\Delta\mathbf{d}(k)\in\mathcal{W}} \{F_xT\left(A_i^{N-3}\Delta\mathbf{d}(k) - A_i^{N-4}\Delta\mathbf{d}(k+1)\cdots - \Delta\mathbf{d}(k+N-3)\right)\} \\ \vdots \\ h_x - \max\limits_{\Delta\mathbf{d}(k)\in\mathcal{W}} \{F_xA_iT\Delta\mathbf{d}(k)\} \\ \hline h_u \\ h_u \\ \vdots \\ h_u \end{array}\right], \tag{4.23b}$$

$$\mathbf{U}_N = \begin{bmatrix} \Delta\mathbf{g}(k) \\ \Delta\mathbf{g}(k+1) \\ \vdots \\ \Delta\mathbf{g}(k+N-1) \end{bmatrix}, \quad \mathbf{V}_N = c(k)\mathbf{U}_N\,, \quad r_v(k) = c(k)\Delta\mathbf{x}_v(k) \tag{4.23c}$$

Finally we use only the first step control of C_N representing the outer control $\Delta\mathbf{g}_v(k)$ in (4.20):

$$\begin{aligned} \Delta\mathbf{g}(k) = c(k)\Delta\mathbf{g}_v(k) + (1-c(k))\Delta\mathbf{g}_0(k) = \\ c(k)\mathbf{U}_N(k) + (1-c(k))L\Delta\mathbf{x}_0(k) = \\ \mathbf{V}_N(k) + L(\Delta\mathbf{x}(k) - r_v(k)) \end{aligned} \tag{4.24}$$

The big advantage of the IIC method is that it can solve control problem for a system with large amount of states and control inputs, while the IC method

is usually limited up to about 4 state-vector, because of C_N's calculation computation burden, see Chapter 8.

Algorithm 6 calculates on-line the IIC trajectory $\Delta\mathbf{x}(k)$, for $k = 1, 2, \dots K$.

Algorithm 6 Calculation of the IIC trajectory $\Delta\mathbf{x}(k)$, for $k = 1, 2, \dots K$

Input: the matrices $A, B, T, L,\ N$, the sets $\Omega_{\max}, \mathcal{X}, \mathcal{U}, \mathcal{W}$, and the initial state $\Delta\mathbf{x}(k)$.

1: $F_N, h_N \leftarrow$ formulation(4.23)$(A, B, N, \mathcal{X}, \mathcal{U}, \mathcal{W}, T, \Omega_{\max})$
2: **for** $k = 1, 2, \dots K$ **do**
3: $\quad c(k), r_v(k), \mathbf{V}_N \leftarrow \text{LP}(4.21)(\Omega_{\max}, F_N, h_N, \Delta\mathbf{x}(k))$
4: $\quad \Delta\mathbf{g}(k) \leftarrow \mathbf{V}_N(k) + L(\Delta\mathbf{x}(k) - r_v(k))$
5: $\quad \Delta\mathbf{x}(k+1) \leftarrow f_{\text{plant}}(\Delta\mathbf{x}(k), \Delta\mathbf{g}(k))$
6: $\quad k \leftarrow k + 1$
7: **end for**

4.4.1 More efficient Improved IC

On the other hand for large numbers of steps N and robust matrices s (or number of uncertainty vertices), we get a large LP problem. Hence, a more efficient improved IC (meIIC) was proposed in [18], where we use only 1 step for representing the outer control, but this time it is not developed from the MAS of our high-gain controller L, but of some low-gain controller, which obviously has larger MAS region. Therefore, the interpolation is now performed between C_1 control invariant set of some low-gain control $\Delta\mathbf{g}_{\text{Low}}(k) = L_{\text{Low}}\Delta\mathbf{x}(k)$ with the larger MAS $\Omega_{\max,\text{Low}}$, and the inner MAS original high-gain controller (4.5). The outer region changes from C_N (4.22) to C_1:

$$C_1 = \begin{cases} A_i\Delta\mathbf{x}(k) + B_i\Delta\mathbf{g}(k) + T\Delta\mathbf{d}(k) \in \Omega_{\max,\text{Low}} \\ \Delta\mathbf{g}(k) \in \mathcal{U} \\ \forall i = 1, 2, \dots, s \end{cases} \tag{4.25}$$

The more efficient improved IC method has a slower convergence compared to the original improved IC, since it uses only 1st control near the border, and has lesser interpolating coefficient with each iteration step. While the original improved IC applies most contractive sequence of control for N steps.

Algorithm 7 calculates the meIIC trajectory $\Delta\mathbf{x}(k)$.

Algorithm 7 Calculation of the meIIC trajectory $\Delta\mathbf{x}(k)$, for $k = 1, 2, \dots K$

Input: the matrices A, B, T, L, the sets $\Omega_{\max}, \Omega_{\max,\text{Low}}, \mathcal{U}, \mathcal{W}$, and the initial state $\Delta\mathbf{x}(k)$.

1: $F_1, h_1 \leftarrow$ formulation(4.23)$(A, B, \mathcal{U}, \mathcal{W}, T, \Omega_{\max,\text{Low}})$
2: **for** $k = 1, 2, \dots K$ **do**
3: $\quad c(k), r_v(k), \mathbf{V}_1 \leftarrow \text{LP}(4.21)(\Omega_{\max}, F_1, h_1, \Delta\mathbf{x}(k))$
4: $\quad \Delta\mathbf{g}(k) \leftarrow \mathbf{V}_1 + L(\Delta\mathbf{x}(k) - r_v(k))$
5: $\quad \Delta\mathbf{x}(k+1) \leftarrow f_{\text{plant}}(\Delta\mathbf{x}(k), \Delta\mathbf{g}(k))$
6: $\quad k \leftarrow k + 1$
7: **end for**

Remark. *Note that disturbance in demand affects only the invariant set calculations in IC or the equivalent matrix in IIC, as it was shown in* (4.11) *and* (4.13)*, respectively. But control laws remain unchanged.*

Also, the inequality sum constraint (3.18) *is problematic for the uncertain linear model which includes disturbance* (4.1)*, since it results with negative* h_u *values, see* (4.3)*, which contradict including the origin in the invariant sets.*

In simple words, if one looks at the last row in the constraints' matrices (4.4b)*, which represent the inequality sum constraint,* h_u *violates the condition of positive* $h_u > 0$ *as stated in* (4.3)*, for the reason of including origin in the invariant sets. For that reason, in numerical runs - some small constant value* ($\epsilon > 0$) *was added to replace 0 and keep* h_u *positive. But nevertheless, when appling* (4.11) *or* (4.13) *methods for calculating invariant sets, if the disturbance term* $T \max_{\Delta\mathbf{d}(k)\in\mathcal{W}} \{F_k \Delta\mathbf{d}(k)\}$ *is positive then the RHS (Right Hand Side) becomes negative, which ruins invariant sets calculation. Hence, IC methods that include explicit invariant sets calculations should be adapted for our specific signal traffic control problem.*

Chapter 5

Simple Interpolating Control (SIC)

The simple interpolating control (SIC) method was recently introduced in [31]. In this chapter, a brief description is given, however, for a more detailed information the reader can refer to [31]. It is a simplified version of IC, which uses saturated control inputs and ellipsoidal feasible sets. Unlike IC, SIC does not need to solve an LP at each time step, which is less computationally expensive than IC. On the other hand, unlike the model used by IC model, see (4.1), SIC considers an uncertain model without disturbance, i.e.

$$\Delta\mathbf{x}(k+1) = A(k)\Delta\mathbf{x}(k) + B(k)\Delta\mathbf{g}(k)\,. \tag{5.1}$$

We start by introducing ellipsoidal invariant sets, then describing saturated linear controllers, and finally presenting the SIC control law.

5.1 Ellipsoidal invariant sets

Ellipsoidal invariant sets are actually ellipsoidal type of Quadratic Lyapunov's function level sets [36].Recall the polyhedral sets, which have been presented in Chapter 4, are piecewise linear level sets. Both types of level set can represent the region of attraction, while the ellipsoidal one is contained in the polyhedral, as shown e.g. in Fig. 5.1, since polyhedral is a more fine convex set representing more accurately the domain of attraction of a linear system in our case. Fig. 5.1 shows for a specific numerical problem, the largest region of attraction using polyhedral MAS (in blue), and a smaller region using a saturated controller (in green) with a general ellipsoid $\tilde{P}$ that can have different inner axes.

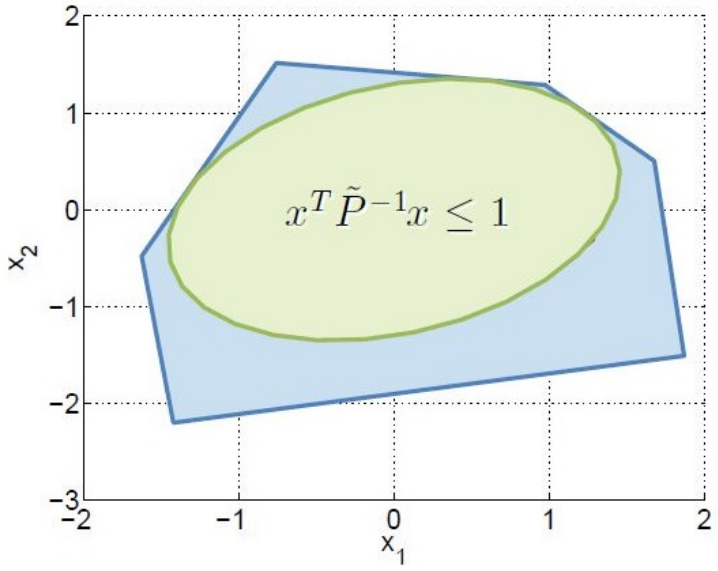


Figure 5.1: Ellipsoidal verse Polyhedral domains of attraction.

In SIC method, two robustly contractive ellipsoids replace the polyhedrals in IC, associated to linear saturated control laws.

The outer controller, $\Delta \mathbf{g}_b(k) = \text{sat}(L_b \Delta \mathbf{x}(k))$, maximizes the invariant set for possible states, where its gain matrix $L_b \in \mathbb{R}^{m \times n}$ is calculated by solving a formulated LMI problem, refer to [16]. While the inner controller, $\Delta \mathbf{g}_a(k) = \text{sat}(L_a \Delta \mathbf{x}(k))$, is responsible for a high performance, where its gain matrix $L_a \in \mathbb{R}^{m \times n}$ is calculated in advance from solving the Riccati equation of an LQR problem. The inner and outer ellipsoidal invariant sets are schematically shown in Fig. 5.2.

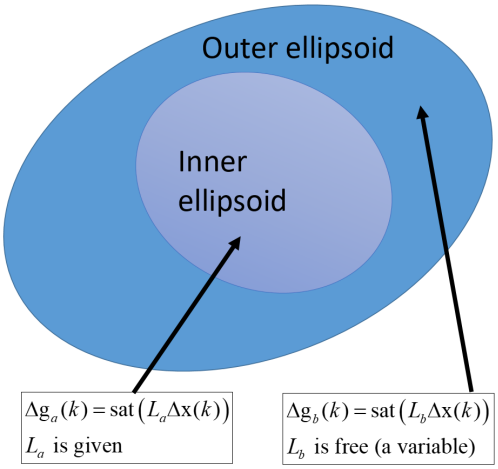


Figure 5.2: Inner and outer ellipsoidal invariant sets.

The ellipsoids are determined by positive definite matrices $P_a \in \mathbb{R}^{n\times n}$ and $P_b \in \mathbb{R}^{n\times n}$, as $\mathcal{E}(P_{el}) = \{\Delta\mathbf{x}(k) : \Delta\mathbf{x}(k)^T P_{el}^{-1} \Delta\mathbf{x}(k) \leq 1\}$, where $el = a, b$. The state constraints are first converted from their general form to a symmetrical one:

$$F_j^T \Delta\mathbf{x}(k) \leq g_j \quad \Rightarrow \quad \frac{F_j^T}{g_j} \Delta\mathbf{x}(k) \leq 1 \quad \Rightarrow \quad F_x \Delta\mathbf{x}(k) \leq 1 \quad \Rightarrow$$
$$|\tilde{F}_x \Delta\mathbf{x}(k)| \leq 1\,, \qquad \forall j = 1, \ldots, n_c\,, \tag{5.2}$$

where n_c is the number of state constraints. The same symmetry property is also requested from the control constraints, therefore, the state and control constraints are assumed to be symmetric around the origin and represented by polytopes of the form:

$$\Delta\mathbf{x}(k) \in \mathcal{X} = \{\Delta\mathbf{x} \in \mathbb{R}^n : |\tilde{F}_x \Delta\mathbf{x}(k)| \leq 1\}, \tag{5.3a}$$
$$\Delta\mathbf{g}(k) \in \mathcal{U} = \{\Delta\mathbf{g} \in \mathbb{R}^m : |\Delta\mathbf{g}(k)| \leq \Delta\mathbf{g}_{\max}\}, \tag{5.3b}$$

where $\tilde{F}_x \in \mathbb{R}^{n_c \times n}$, and in our traffic signal problem we represent the state and control constraints from Section 3.4 in the form (5.3) with

$$\tilde{F}_x = \begin{bmatrix} 1 \Big/ \Delta\tilde{x}_1 & 0 & \cdots & 0 \\ 0 & 1 \Big/ \Delta\tilde{x}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \Big/ \Delta\tilde{x}_n \end{bmatrix}, \tag{5.4}$$

$$\Delta\mathbf{g}_{\max} = \min(g_{j,i}^{\mathrm{N}} - g_{j,i,\min}, g_{j,i,\max} - g_{j,i}^{\mathrm{N}}) \in \mathbb{R}^m, \tag{5.5}$$

where $n_c = n$ and $\Delta\tilde{x}_z \ \forall z \in Z$, defined as the maximum deviation from the nominal value:

$$|\Delta x_z| \leq \min(x_z^{\mathrm{N}}, x_z^{\max} - x_z^{\mathrm{N}}) \equiv \Delta\tilde{x}_z, \quad \forall z \in Z\,. \tag{5.6}$$

Note that the original state and control constraints as used in IC (4.3) might be changed if they are not symmetrical in the first place. Hence, as a result, both state and control constraints can be in the best case the same, and in any other case tighter, in order not to violate the constraints and also to keep them symmetrical.

The state constraints (5.3a) can be transformed to LMI type of constraints [16] as follows

$$\begin{bmatrix} 1 & \tilde{F}_{xj}^T P_{el} \\ P_{el}\tilde{F}_{xj} & 1 \end{bmatrix} \succeq 0\,, \qquad \forall j = 1, \ldots, n\,, \tag{5.7}$$

where $el = a, b$, i.e. the constraints are for both the inner and outer ellipsoids P_a and P_b.

The ellipsoid set is also transformed into an LMI condition, where the demand for control invariant set is that any next state evolution $\Delta\mathbf{x}(k+1)$ starting from ellipsoid's boundary $\Delta\mathbf{x}(k) : \Delta\mathbf{x}(k)^T P_{el}^{-1} \Delta\mathbf{x}(k) = 1$ - will be either on the border or inside the ellipsoid:

$$\Delta\mathbf{x}(k+1)^T P_{el}^{-1} \Delta\mathbf{x}(k+1) \leq \Delta\mathbf{x}(k)^T P_{el}^{-1} \Delta\mathbf{x}(k), \tag{5.8}$$

which after applying the system dynamics (5.1) becomes:

$$(A(k)\Delta\mathbf{x}(k) + B(k)\Delta\mathbf{g}(k))^T P_{el}^{-1} (A(k)\Delta\mathbf{x}(k) + B(k)\Delta\mathbf{g}(k)) \leq \Delta\mathbf{x}(k)^T P_{el}^{-1} \Delta\mathbf{x}(k), \tag{5.9}$$

which is necessary and sufficient condition for each of the ellipsoids to be robustly invariant, given the controller.

Finally, the objective function is

$$J = \max_{X_{el}, Y_{el}, P_{el}} \{trace(P_{el})\} \tag{5.10}$$

which finds the largest ellipsoid' invariant set, in order to enclose more initial conditions. It can be achieved either by maximizing the trace of P_{el} as stated above, which means sum of lengths of ellipsoid's semi axes. Or by minimizing the determinant of P_{el}, which means maximizing the volume of the ellipse. Maximizing the trace problem is computationally an easier problem than minimizing the determinant.

The variables added to the problem in (5.10) X_{el}, Y_{el} are explained later.

5.2 Saturated linear controllers

It is important to clarify that SIC applies saturated linear controllers (also referred as a "convex hull of ellipsoids"), using a special control law, solved algebraically. In general, saturated linear controllers use a control law that solves LMI at each iteration, see [16].

The saturated controller relies on two saturating linear control laws of the form

$$\Delta\mathbf{g}(k) = \mathrm{sat}(L\Delta\mathbf{x}(k)) \equiv \sum_{q=1}^{2^m} \beta_q(k)(E_q L\Delta\mathbf{x}(k) + E_q^- H\Delta\mathbf{x}(k))\,, \qquad (5.11)$$

where $H \in \mathbb{R}^{m\times n}$ is unknown gain matrix and $E_q \in \mathbb{R}^{m\times m}$ is a square diagonal matrix, which holds in its diagonal the binary form of the number $q-1$, $E_q^- = I_{m\times m} - E_q$ where I is the unit matrix. For example for $m = 3, q = 4$:

$$E_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

In this case of saturated controller (5.11), the linear controller $\Delta\mathbf{g}(k) = L_{el}\Delta\mathbf{x}(k)$ is not constrained. However, the second linear controller $H_{el,j}\Delta\mathbf{x}(k)$ is constrained:

$$|\Delta\mathbf{g}(k)| = |H_{el,j}\Delta\mathbf{x}(k)| \le \Delta\mathbf{g}_{\mathrm{max,j}}\ \forall j = 1,\ldots,m\,. \qquad (5.12)$$

The LMI representing the control constraints

$$\begin{bmatrix} \Delta\mathbf{g}^2_{\mathrm{j,max}} & H_{el,j}P_{el} \\ P_{el}H_{el,j}^T & P_{el} \end{bmatrix} \succeq 0\ \forall j = 1,\ldots,m \qquad (5.13)$$

and the ellipsoid constraint (5.9) after using convex hull of the extreme or uncertain matrices (4.10) become:

$$\begin{bmatrix} P_{el} & \{A_i + B_i(E_q L_{el} + E_q^- H_{el})\}P_{el} \\ (\{A_i + B_i(E_q L_{el} + E_q^- H_{el})\}P_{el})^T & P_{el} \end{bmatrix} \succeq 0$$
$$\forall i = 1,\ldots,s, \forall q = 1,\ldots,2^m\,. \qquad (5.14)$$

To summarize, the problem formulation for saturated controller is (5.10), (5.7), (5.13), and (5.14), which represent the LMI objective function, state constraints, control constraints, and ellipsoidal constraints, respectively.

The inner problem is when L_a matrix is given as in (5.14), but H_a is unknown. Therefore, a new variable is added $Y_a \in \mathbb{R}^{m\times n}$ which replaces H_aP_a in (5.14), in-order to turn the LMI problem to be linear. After a solution is found, H_a retrieved using $H_a = Y_aP_a^{-1}$.

The outer problem is when both L_b and H_b are unknown, in-order to find the largest ellipsoidal set. Therefore, two new variables introduced $Y_b \in \mathbb{R}^{m\times n}, X_b \in \mathbb{R}^{m\times n}$ which replace H_bP_b, L_bP_b respectively, in (5.14), in-order to turn the LMI problem to be linear. After a solution is found, H_b, L_b retrieved using $H_b = Y_bP_b^{-1}, L_b = X_bP_b^{-1}$, respectively.

The off-line SIC calculation is shown in Algorithm 8.

5.3 SIC interpolating control law

A state $\Delta\mathbf{x}(k) \in \mathcal{E}(P_b) \setminus \mathcal{E}(P_a)$ can be decomposed similarly to (4.19), as

$$\Delta\mathbf{x}(k) = c(k)\Delta\mathbf{x}_b(k) + (1 - c(k))\Delta\mathbf{x}_a(k)\,, \tag{5.15}$$

where $\Delta\mathbf{x}_a(k) \in \mathcal{E}(P_a)$, $\Delta\mathbf{x}_b(k) \in \mathcal{E}(P_b)$ and $0 \leq c(k) \leq 1$. The inner and outer controllers are blended, similarly to (4.20), as

$$\Delta\mathbf{g}(k) = c(k)\text{sat}(L_b\Delta\mathbf{x}_b(k)) + (1 - c(k))\text{sat}(L_a\Delta\mathbf{x}_a(k))\,, \tag{5.16}$$

where at each step $\Delta\mathbf{x}_b(k)$, $\Delta\mathbf{x}_a(k)$, $c(k)$ are calculated simply algebraically, as described below, and not by solving any LP problem such as in IC case, see Section 4.3, or SDP (semi-definite problem) where an LMI solved at each iteration as in saturated case.

Assuming that $\mathcal{E}(P_a) \subset \mathcal{E}(P_b)$, and a line connecting the origin with $\Delta\mathbf{x}(k)$ one can calculate the following terms, see also Fig. 5.3,

$$a(k)^2 = \Delta\mathbf{x}(k)^TP_a^{-1}\Delta\mathbf{x}(k), \text{where } a(k) > 1 \Rightarrow \Delta\mathbf{x}_a(k) = \Delta\mathbf{x}(k)/a(k) \tag{5.17}$$

$$b(k)^2 = \Delta\mathbf{x}(k)^TP_b^{-1}\Delta\mathbf{x}(k), \text{where } b(k) \leq 1 \Rightarrow \Delta\mathbf{x}_b(k) = \Delta\mathbf{x}(k)/b(k) \tag{5.18}$$

where both $\Delta\mathbf{x}_a(k)$ and $\Delta\mathbf{x}_b(k)$ are lying on a line. Now using decomposition (5.15), we can find the interpolation coefficient $c(k)$:

$$c(k) = \frac{b(k)(a(k) - 1)}{a(k) - b(k)}\,. \tag{5.19}$$

Note that (5.16) and (5.19) do not guarantee closed loop stability, and limit cycles can occur, refer to [31].

Algorithm 8 Calculation of the maximal ellipsoidal invariant sets P_a, P_b

Input: the matrices $A, B, \tilde{F}_x, L_a$, $\Delta\mathbf{g}_{\max}$, the sets $\mathcal{X}$ and $\mathcal{U}$

$X_b, Y_b, P_b \leftarrow \text{SDP}\left(\max_{X_b,Y_b,P_b}\{trace(P_b)\}\right)$ $\Big\{$

for $i = 1, 2, \ldots s$ **do**

  for $q = 1, 2, \ldots 2_m$ **do**

$$\begin{bmatrix} P_b & A_iP_b + B_i(E_qX_b + E_q^-Y_b) \\ (A_iP_b + B_i(E_qX_b + E_q^-Y_b))^T & P_b \end{bmatrix} \succeq 0$$

  end for

end for

for $j = 1, 2, \ldots n$ **do**

$$\begin{bmatrix} 1 & \tilde{F}_{xj}^T P_b \\ P_b\tilde{F}_{xj} & 1 \end{bmatrix} \succeq 0$$

end for

for $j = 1, 2, \ldots m$ **do**

$$\begin{bmatrix} \Delta\mathbf{g}_{\text{j,max}}^2 & Y_{b,j} \\ Y_{b,j}^T & P_b \end{bmatrix} \succeq 0$$

end for $\Big\}$

$L_b \leftarrow X_bP_b^{-1}$

$H_b \leftarrow Y_bP_b^{-1}$

$Y_a, P_a \leftarrow \text{SDP}\left(\max_{Y_a,P_a}\{trace(P_a)\}\right)$ $\Big\{$

for $i = 1, 2, \ldots s$ **do**

  for $q = 1, 2, \ldots 2_m$ **do**

$$\begin{bmatrix} P_a & A_iP_a + B_i(E_qL_aP_a + E_q^-Y_a) \\ (A_iP_a + B_i(E_qL_aP_a + E_q^-Y_a))^T & P_a \end{bmatrix} \succeq 0$$

  end for

end for

for $j = 1, 2, \ldots n$ **do**

$$\begin{bmatrix} 1 & \tilde{F}_{xj}^T P_b \\ P_b\tilde{F}_{xj} & 1 \end{bmatrix} \succeq 0$$

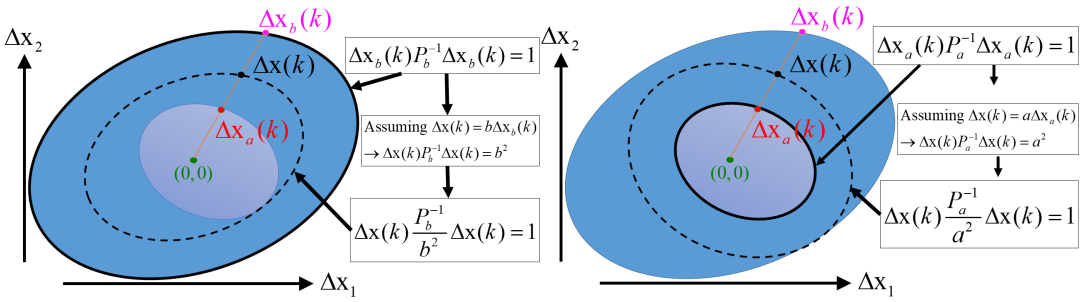
end for

for $j = 1, 2, \ldots m$ **do**

$$\begin{bmatrix} \Delta\mathbf{g}_{\text{j,max}}^2 & Y_{b,j} \\ Y_{b,j}^T & P_b \end{bmatrix} \succeq 0$$

end for $\Big\}$

$H_a \leftarrow Y_aP_a^{-1}$



(a) Finding $\Delta \mathbf{x}_b(k)$ (b) Finding $\Delta \mathbf{x}_a(k)$

Figure 5.3: Calculating $\Delta \mathbf{x}_a(k)$, $\Delta \mathbf{x}_b(k)$ using $\Delta \mathbf{x}(k)$, L_a, and L_b, P_a, P_b which derived from solving LMIs.

We can see also that $c(k) = 0 \ \forall \Delta \mathbf{x}(k) \in \mathcal{E}(P_a)$, and then there is no interpolation, and it behaves according to the inner ellipsoid's control law.

In SIC as in saturated controllers, the ellipsoids are found by (5.10), (5.7), (5.14), (5.13). Algorithm 9 calculates the SIC trajectory $\Delta \mathbf{x}(k)$ for $k = 1, 2, \dots K$.

In summary, comparing SIC with IC, then the SIC method has a simpler algorithm, which provides a good compromise between performance, computational load, and feasible region.

Algorithm 9 Calculation of the SIC trajectory $\Delta \mathbf{x}(k)$ for $k = 1, 2, \dots K$

Input: the matrices A, B, L_a, L_b, P_a, P_b, and the initial state $\Delta \mathbf{x}(k)$.

for $k = 1, 2, \dots K$ **do**
  $a(k) \leftarrow \sqrt{\Delta \mathbf{x}(k)^T P_a^{-1} \Delta \mathbf{x}(k)}$, $b(k) \leftarrow \sqrt{\Delta \mathbf{x}(k)^T P_b^{-1} \Delta \mathbf{x}(k)}$
  $\Delta \mathbf{x}_a(k) \leftarrow \Delta \mathbf{x}(k)/a(k)$, $\Delta \mathbf{x}_b(k) \leftarrow \Delta \mathbf{x}(k)/b(k)$
  $u_a(k) \leftarrow \text{sat}(L_a \Delta \mathbf{x}_a(k))$, $u_b(k) \leftarrow \text{sat}(L_b \Delta \mathbf{x}_b(k))$, $c(k) \leftarrow 0$
  if $a \neq b$ **then**
    $c(k) \leftarrow \frac{b(k)(a(k)-1)}{a(k)-b(k)}$
  end if
  if $a \leq 1$ **then**
    $c(k) \leftarrow 0$, $u_a(k) \leftarrow \text{sat}(L_a \Delta \mathbf{x}(k))$
  end if
  $\Delta \mathbf{g}(k) \leftarrow c(k) u_b(k) + (1 - c(k)) u_a(k)$
  $\Delta \mathbf{x}(k+1) \leftarrow f_{\text{plant}}(\Delta \mathbf{x}(k), \Delta \mathbf{g}(k))$
  $k \leftarrow k + 1$
end for

Chapter 6

Comparison between presented control methods

This chapter compares between four control methods used in this thesis, i.e. LQR, MPC, IC, and SIC.

6.1 Control implementation framework

In this section, the control implementation framework is presented. In this framework, the controllers which are designed by an ideal nominal model, should be implemented on the actual plant, i.e. reality, which here is represented as uncertain model. The general framework is shown in Fig. 6.1.

In this thesis, the designed controllers are classified into two groups: (i) those that need some online optimization performed every step of the simulation, i.e. MPC and IC, and (ii) those that do not need any optimization calculation, but rather a simple algebraic one, for obtaining the control signal, i.e. LQR and SIC. Note that MPC and LQR are designed based on the nominal model (2.1) without uncertainties, as already mentioned in Chapter 2, while IC as the main robust control method in this thesis, is designed based for an uncertain discrete-time linear model (3.6) with the state and control polytopic constraints, including the demand uncertainty range (4.3).

In this thesis, robust design has been implemented by the interpolating control approach. The robust MPC approach has not been implemented, since it is claimed in several papers [37–39] that robust MPC is too conservative and too heavy computationally.

In this thesis, the plant model simulated in the numerical results is the same as the designed model.

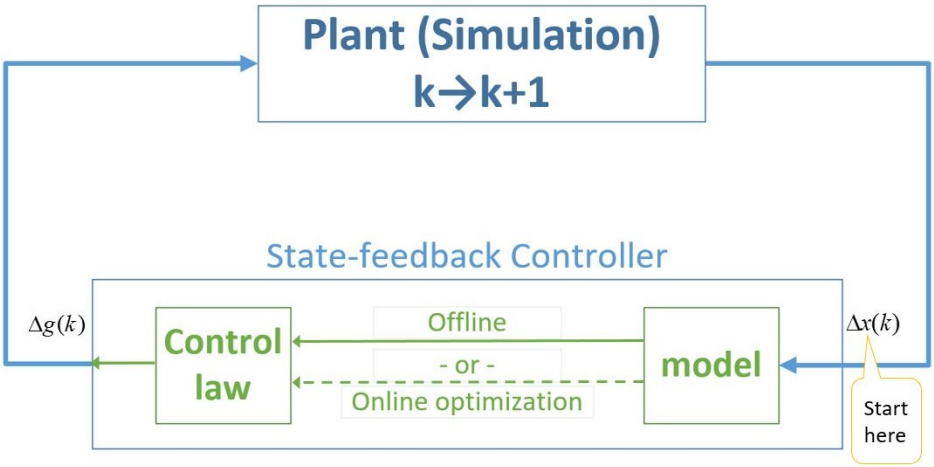


Figure 6.1: State-feedback controllers implementation on a plant.

6.2 Comparison between control methods

A brief summary of the main characteristics of the methods is shown in Table 6.1. The control laws in all methods are state feedback controllers, where the LQR is different from other methods by being linear, therefore, its performance is expected to be more sensitive to the distance from the origin.

In calculation for each step, we distinguish between the methods that solve optimization problem online, i.e. MPC which solves one QP, and IC which solves 2 LPs; and the methods that calculate what is needed offline, i.e. LQR which solves Riccati equation to calculate the gain matrix, and SIC which solves 2 LMIs to calculate inner and outer invariant ellipsoidal sets.

It should be stressed that MPC and IC handle state and control constraint within the control design level. On the other hand, LQR does not include the state and control constraints, while SIC includes only some of them. Note that in LQR one needs to check if there is a violation of constraints, then one can solve QP to handle constraints. In SIC, only symmetric constraints can be integrated. All other constraints should be checked for violation, which, in case that occurs, then a QP has to be solved. In both SIC and LQR, applying constraints after control law may ruin the optimality and the stability. For more details about this QP, see Appendix A.2.

Now, the updated algorithms for on-line LQR, i.e. Algorithm 1, and on-line SIC, i.e. Algorithm 9, taking into account constraints are respectively algorithms 10 and 11.

Algorithm 10 Calculation of the LQR trajectory $\Delta\mathbf{x}(k)$ for $k = 1, 2, \dots K$

Input: the matrices A, B, L, the initial state $\Delta\mathbf{x}(k)$, the set $\mathcal{U}$.

for $k = 1, 2, \dots K$ **do**
  $\Delta\mathbf{g}(k) \leftarrow L\Delta\mathbf{x}(k)$
  if $\Delta\mathbf{g}(k) \notin \mathcal{U}$ **then**
    $\Delta\tilde{\mathbf{g}}(k) \leftarrow$ Algorithm 12$(\Delta\mathbf{g}(k), \mathcal{U})$
  else
    $\Delta\tilde{\mathbf{g}}(k) \leftarrow \Delta\mathbf{g}(k)$
  end if
  $\Delta\mathbf{x}(k+1) \leftarrow f_{\text{plant}}(\Delta\mathbf{x}(k), \Delta\tilde{\mathbf{g}}(k))$
  $k \leftarrow k + 1$
end for

Algorithm 11 Calculation of the SIC trajectory $\Delta\mathbf{x}(k)$ for $k = 1, 2, \dots K$

Input: the matrices A, B, L_a, L_b, P_a, P_b, the initial state $\Delta\mathbf{x}(k)$, the set $\mathcal{U}$.

for $k = 1, 2, \dots K$ **do**
  $a(k) \leftarrow \sqrt{\Delta\mathbf{x}(k)^T P_a^{-1} \Delta\mathbf{x}(k)}$, $b(k) \leftarrow \sqrt{\Delta\mathbf{x}(k)^T P_b^{-1} \Delta\mathbf{x}(k)}$
  $\Delta\mathbf{x}_a(k) \leftarrow \Delta\mathbf{x}(k)/a(k)$, $\Delta\mathbf{x}_b(k) \leftarrow \Delta\mathbf{x}(k)/b(k)$
  $u_a(k) \leftarrow \text{sat}(L_a\Delta\mathbf{x}_a(k))$, $u_b(k) \leftarrow \text{sat}(L_b\Delta\mathbf{x}_b(k))$, $c(k) \leftarrow 0$
  if $a \neq b$ **then**
    $c(k) \leftarrow \frac{b(k)(a(k)-1)}{a(k)-b(k)}$
  end if
  if $a \leq 1$ **then**
    $c(k) \leftarrow 0$, $u_a(k) \leftarrow \text{sat}(L_a\Delta\mathbf{x}(k))$
  end if
  $\Delta\mathbf{g}(k) \leftarrow c(k)u_b(k) + (1 - c(k))u_a(k)$
  if $\Delta\mathbf{g}(k) \notin \mathcal{U}$ **then**
    $\Delta\tilde{\mathbf{g}}(k) \leftarrow$ Algorithm 12$(\Delta\mathbf{g}(k), \mathcal{U})$
  else
    $\Delta\tilde{\mathbf{g}}(k) \leftarrow \Delta\mathbf{g}(k)$
  end if
  $\Delta\mathbf{x}(k+1) \leftarrow f_{\text{plant}}(\Delta\mathbf{x}(k), \Delta\tilde{\mathbf{g}}(k))$
  $k \leftarrow k + 1$
end for

6.2.1 Control methods implementation

Summary of the algorithms which are used in the numerical examples in the following chapters is as follows:

- LQR on-line: Algorithm 10.
- MPC on-line: Algorithm 2.
- IC:
  - off-line: Algorithm 3,
    - Minkowski-norm: Algorithm 4,
  - on-line:
    - Minkowski-norm: Algorithm 5.
    - IIC: Algorithm 6.
    - meIIC: Algorithm 7.
- SIC:
  - off-line: Algorithm 8,
  - on-line: Algorithm 11.

where for methods that deploy off-line algorithms, execute first off-line and then their on-line counterparts. If considering invariant sets calculation only, the off-line algorithms for IC and SIC are algorithms 3 and 8, respectively.

All IC and SIC algorithms hold for the nominal model (2.1) with A, B. While for the uncertain model without disturbance in demand (5.1): (i) A_i, B_i are used instead of A, B for calculation, and (ii) $A(k)$, $B(k)$ are used instead of A, B for state evolution using the plant model (which in our case is the designed model).

LQR and MPC both seek to explicitly optimize objective functions, unlike IC and SIC which have the objective of switching control from the outer one to the inner one. However, it is important to mention that LQR and MPC are only relatively optimal, comparing to IC and SIC. In reality they are also sub-optimal, since their predictive horizon is limited to some N steps and not for the duration of the actual traffic signal control problem, which might be much longer.

MPC can be less optimal in cases of $N_m < N$ or/and if we add a terminal term to ensure stability and recursive feasibility. LQR has a fixed horizon, while MPC and IIC both have rolling receding horizon, which is expressed by the fact that both solve on-line optimization each step. The difference between MPC and IIC is that the control sequence of MPC that drives initial state into MAS (if a terminal term is added) is optimal, while IC is not. IC

and SIC has no horizon at all, since polyhedral and ellipsoid invariant sets are calculated off-line once. The IIC has receding horizon, since the main LP in IIC is solved at each step for such horizon.

LQR, MPC, and IC have the most general polyhedral type of feasible region to represent constraints. Since SIC has ellipsoidal invariant sets, it requires a symmetrical constraints only. If there are non-symmetric constraints in the problem, then they have to be symmetrically adjusted or to be applied after control law, as described in Appendix A.2.

	Feature	LQR	MPC	IC	SIC
Control	State Feedback	✓	✓	✓	✓
	$\Delta\mathbf{g}(k)$ law	$L\Delta\mathbf{x}(k)$	$f(\Delta\mathbf{x}(k))$	$f(\Delta\mathbf{x}(k))$	$f(\Delta\mathbf{x}(k))$
	Calculation each step	algebraic offline	QP online	2 LP's online	algebraic offline
	Constraints implementation	after control law	embedded	embedded	after control or embedded
	Optimality	Optimal w/o constraints	Optimal	sub-Optimal	sub-Optimal
	Predictive horizon	✓ fixed	✓ receding	✓ receding	-
	Feasible region	polyhedral	polyhedral	polyhedral	ellipsoidal
	Constraint type	regular	regular	regular	symmetric

Table 6.1: Methods comparison table.

6.3 Computational efforts analysis

The computation effort of the controller is an important comparison criteria. Previous analysis of the computational efforts for the IC controller, and comparison with MPC and LQR controllers, have been conducted in [17,20]. These works present the computational advantage of IC controllers over MPC and LQR controllers.

In the following, we conduct a further comparison oriented to the traffic control problem dealt with in this thesis. We would like to compare some complexity characteristics, in order to evaluate the tradeoff between performance and computation load/time.

The computational efforts are compared by examining the number of constraint equations and the number of variables in the problems of the LQR, MPC, IC, and SIC methods. The results of this analysis are summarized in Table 6.2, where the notations are as follows: *dlqr* is the discrete-time Riccati equation solver, *LMI* is the linear matrix inequalities, *LP* is the Linear Programming, *QP* is the Quadratic Programming, *nx* is the total number of states (total links in all intersections), *nu* is the total number of control inputs (total phases in all intersections), *njuncs* is the number of intersections, N is the MPC prediction and control horizon, *ngi* and *ngo* are the number of half-planes defining the sets $\Omega_{\max}$ (inner) and C_N (outer), respectively.

Moreover, for the calculation of total number of constraints in IIC (improved IC) and meIIC (more efficient IIC), we use IC's number of state and control constraints, taken from (4.4), where $nfx = 2nx$, and $nfu = 2nu + njuncs$.

The QP of MPC and the QP of LQR or SIC are different, since LQR's (or SIC's) QP is for the purpose of finding the closest control signal to the optimal one, only now also constrained. While in MPC the QP is the actual optimization problem. The QP is applied after the control law of LQR or SIC, while MPC's QP is applied during control law design including the constraints in the problem.

	Feature	LQR	MPC	IC	IIC	meIIC	SIC
	offline solver	dlqr	–	$\Omega_{\max}$, C_N computation	$\Omega_{\max}$ computation	$\Omega_{\max}$, $\Omega_{\max,\text{Low}}$ computation	2(2nu+nx) LMI's
Online Solver	online solver 1	QP	QP	LP	LP	LP	QP (if needed)
	# of constraints	njuncs	nx$N$	ngo+ngi+1	2ngi+1+ (nfx+nfu)N	ngi+ngi_{Low}+1 +(nfx+nfu)	njuncs
	# of variables	nu	nu$N$	nx+1	nu$N$+nx+1	nu+nx+1	nu
	online solver 2	–	–	LP	–	–	–
	# of constraints	–	–	ngo+2nu+1	–	–	–
	# of variables	–	–	nu+1	–	–	–

Table 6.2: Complexity comparison table.

Chapter 7

Numerical examples for 2-dimensional state space

The performances of the interpolating controllers are examined via an application of traffic signal control for an isolated signalized intersection, schematically shown in Fig. 7.1. The intersection has two movements/states, and two phases/controllers.

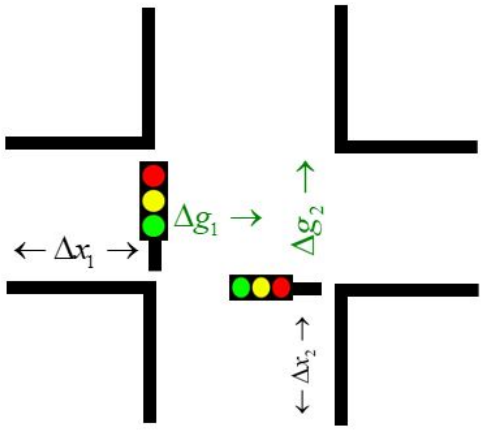


Figure 7.1: A single intersection illustration.

Three numerical examples are presented: example 1 deals with a nominal case scenario, while examples 2 and 3 deal with an uncertain case scenario, for IC and SIC, respectively. In example 1, comparison is carried out between the results obtained from applying the interpolating controllers, LQR, and MPC controllers, which are described in detail in Chapter 2.

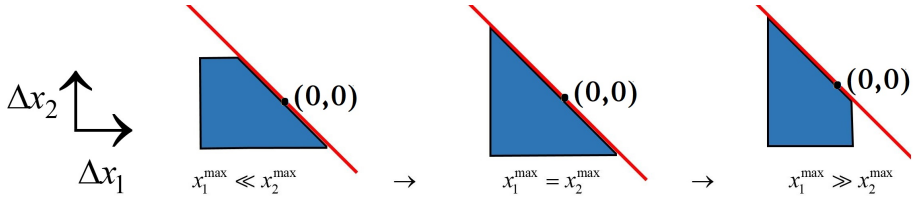
The following default values are used for all numerical results, unless specified otherwise: the prediction horizon for IC and MPC methods is $N =$

10, the maximum saturation flow is $S = 1.42$ [veh/sec] for all links, the cycle period is $C = 120$ [sec], and the lost time is $L = 8$ [sec] for all intersections. The nominal state value is assumed to be $\mathbf{x}^{\mathrm{N}} = \mathbf{x}^{\max}/2$ for all $\mathbf{x}^{\max}$. $\rho = 0.01$ in the weighting matrix $R = \rho I$, in the quadratic objective function of LQR and MPC. For more details see Appendix A.1.

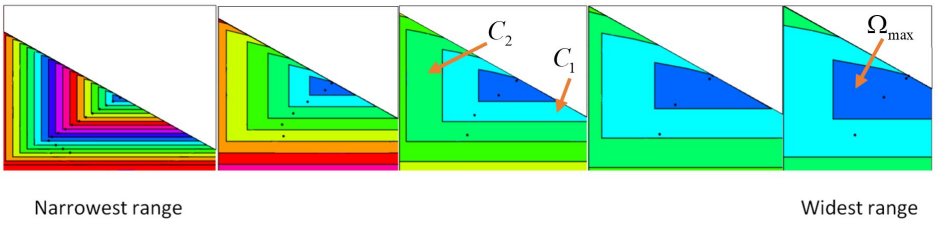
7.1 Numerical example 1: Nominal case

First, a sensitivity analysis is carried out to study how changing the constraints (3.16) and (3.17) influences the invariant sets $\Omega_{\max}$, C_N, and the trajectories.

The effect of changing the relative values of $x_1^{\max}$ and $x_2^{\max}$ in (3.16) on the $\Omega_{\max}$ is shown in Fig. 7.2(a). Identical control nominal values for all phases and symmetrical range constraints are assumed. It is shown that changing the relative values of $x_1^{\max}$ and $x_2^{\max}$ changes the shape of the $\Omega_{\max}$ in different directions. Note that the sum equality constraint (3.18) is shown in the figure as a red line.



(a) Changing the relative values of $x_1^{\max}$, $x_2^{\max}$ in (3.16), while control nominal values and range constraints are symmetrical.



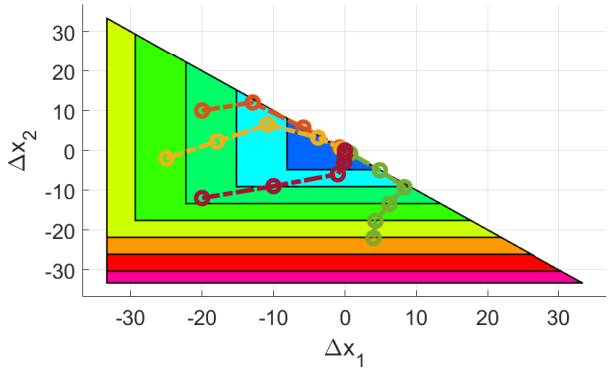
(b) Changing the lower and upper bounds of the constraint (3.17) by $g_{j,i,\min}$ and $g_{j,i,\max}$.

Figure 7.2: Example 1: The effect of the constraints (3.16) and (3.17) on the invariant sets $\Omega_{\max}$, C_N, and the trajectories.

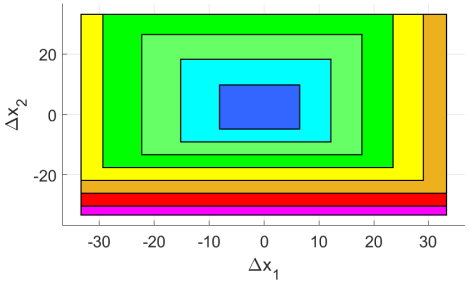
The effect of changing the values of $g_{j,i,\min}$ and $g_{j,i,\max}$ in (3.17) on the $\Omega_{\max}$, C_N, and trajectories are shown in Fig. 7.2(b), where all the outer triangular sets have exactly the same size, while the inner triangular sets

and the trajectories' step sizes are different. It is shown that relaxing the limits of constraint (3.17), i.e. widening the control feasible set, results in a larger $\Omega_{\max}$ region, and accordingly a larger C_N. The black dots in each subfigure show a trajectory that starts from an initial state and ends at the origin. Comparing the trajectories of narrow and wide ranges, it is shown that less number of control steps is needed to reach the origin for wide ranges. It should be stressed that a wider range means less constrictive process, which implies a larger $\Omega_{\max}$, and as a result a larger control admissible region C_N is obtained at each step, which in its turn allows a faster convergence to the origin via less steps.

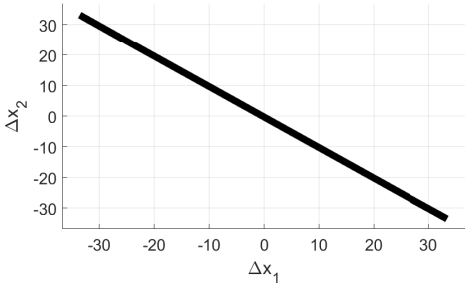
The results for IC trajectories from different initial states, for the case of $x_1^{\max} = x_2^{\max}$, are shown in Fig. 7.3(a).



(a) Trajectories from different initial states.



(b) IC invariant sets without sum constraint (3.18).



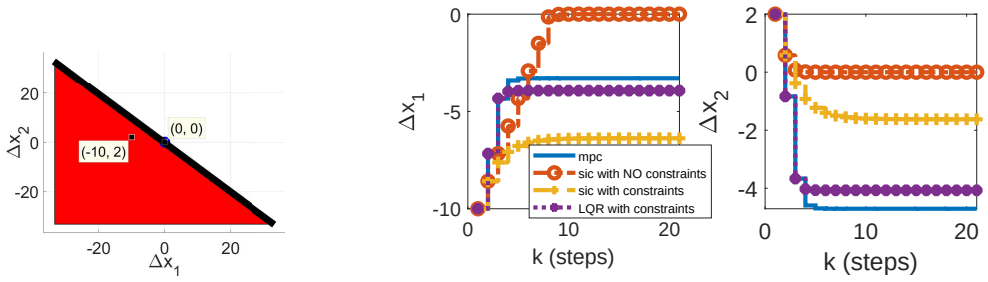
(c) IC invariant sets with equality sum constraint (3.18).

Figure 7.3: Example 1: IC invariant sets and trajectories.

If we apply only the range constraints, for control inputs and states, see (3.16) and (3.17), without the sum constraint (3.18), then the triangle shaped invariant sets as shown in Fig. 7.3(a) become rectangle shaped as in

Fig. 7.3(b). Furthermore, if we apply equality sum constraints (3.18) instead of inequality, then all invariant sets become a line (no area) as shown in Fig. 7.3(c). In other words, imposing range constraints only, one gets rectangle shaped invariant sets, and after imposing the inequality sum constraint, the sets change to triangles since the sum constraint "cuts" the rectangle sets and create triangle shaped invariant sets.

In the case that the equality sum constraint is applied, the trajectories obtained from SIC, LQR, and MPC converge but not to the origin, as shown in Fig. 7.4(b), when the initial state is outside the invariant set which is the line, state (-10,2), see Fig. 7.4(a). By "outside" it means that the initial state is inside the region of invariant set of the *inequality* sum constraint (the red region in Fig. 7.4(a)). However, when the initial state is on the line, i.e. (-7,7), then the obtained trajectories from SIC, LQR, and MPC do converge to the origin.



(a) Invariant sets $\Omega_{\max}$, C_N for equality sum constraint.

(b) Comparing all solutions with initial points outside the invariant sets.

Figure 7.4: Example 1: Comparison between control methods under equality sum constraint (3.18).

The results shown in Fig. 7.4 are given for the nominal model in (2.1) with the following parameters and initial state, for each of the n links, and for each of the m phases:

$$\mathbf{x}^{\max} = [46.67, 66.67]\,, \quad \mathbf{g}^{\min} = [51, 53]\,, \quad \mathbf{g}^{\max} = [59, 56]\,,$$
$$\mathbf{g}^{\mathrm{N}} = [58, 54]\,, \quad \Delta\mathbf{x}(0) = [-10, 2]\,, \quad n = m = 2\,.$$

From this point until the end of the thesis, the inequality type of sum constraint is considered for all numerical results. In addition, it is shown in Fig. 7.4(b) that SIC without constraints converges to the origin. This emphasizes the fact that including constraints after control law in SIC case

(see Appendix A.2) might ruin the performance, i.e. not converging to origin. And with that conclusion, from this point until end of the thesis, the SIC and LQR methods are considered with the state and control constraints for all numerical results.

Trajectories and control inputs over time results, i.e. $\Delta x_1(k)$, $\Delta x_2(k)$ and $\Delta g_1(k)$, $\Delta g_2(k)$, are shown in Fig. 7.5 for (i) IC, (ii) SIC, (iii) MPC, and (iv) LQR. All methods take into account the constraints (3.16)–(3.18). Note that the implemented LQR takes into account the constraints by imposing them after control input calculations, while SIC does so only if needed.

The design of LQR and MPC controller require to determine the weighting matrices Q, R, see objective function in (2.2). The diagonal elements of Q are set equal to $1/\Delta x_{z,\max}$ where $\Delta x_{z,\max} = x_{z,\max} - x^N$ derived from (3.6), unlike (5.4), in order to minimize and balance the relative number of vehicles of the network links. And the strength of the control influence expressed using the weighting matrix $R = \rho I$, where I is the unit matrix. To this end, the choice of ρ may be performed via a trial-and-error procedure. For more details see Appendix A.1.

As mention in Chapter 2, in LQR, at each step we apply constraints only after calculating the optimal signal control inputs, which might ruin optimality, whilst in MPC all three types of constraints (3.16)–(3.18) are embedded in the problem formulation.

The results show that the MPC and LQR controllers converge faster to the origin compared with the IC and SIC. Actually, the MPC, when it is unconstrained, has identical optimal solution to the LQR, given the same objective function. Note that SIC with constraints converges even slower than the IC, probably because of the constraint symmetry requirements of the method, see (5.3), which render the SIC to be more conservative than IC. Also, might be due to constraints applied after SIC's control law.

Indeed, the IC and SIC might provide sub-optimal solutions in terms of convergence, however, the IC and SIC controllers have advantages over the MPC and LQR controllers in the context of computational complexity, as considered in Section 6.3.

The results shown in Fig. 7.5 are given for the nominal model in (2.1) with the following parameters and initial state, for each of the n links, and for each of the m phases:

$$\mathbf{x}^{\max} = [46.67, 66.67]\,, \quad \mathbf{g}^{\min} = [51, 52]\,, \quad \mathbf{g}^{\max} = [59, 62]\,,$$
$$\mathbf{g}^{\mathrm{N}} = [58, 54]\,, \quad \Delta\mathbf{x}(0) = [-20, -12]\,, \quad n = m = 2.$$

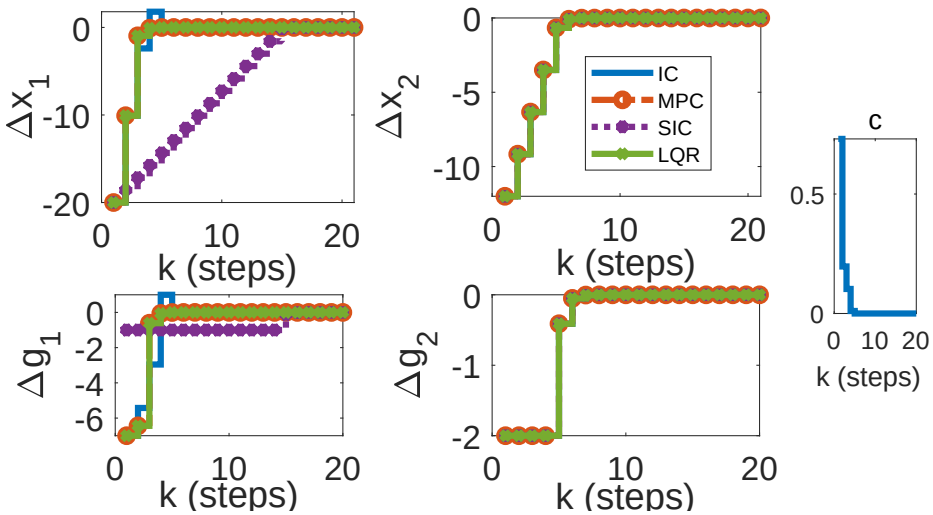


Figure 7.5: Example 1: Trajectories and control inputs over time results obtained from (i) IC, (ii) SIC, (iii) MPC, and (iv) LQR.

7.2 Numerical example 2: Robust IC case

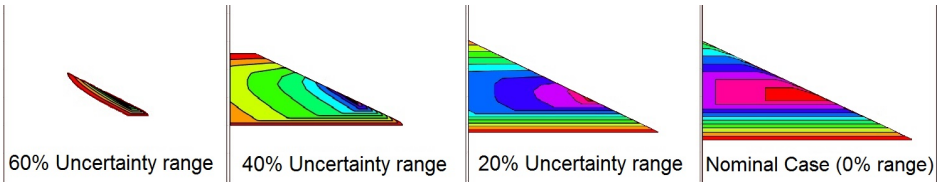
As mentioned in Section 3.3, several uncertainties might exist in the urban road networks. In this numerical example, uncertainty in saturation flows is considered, see (3.9). Note that in this thesis, when we study robust cases, the interpolation-based methods IC and SIC are considered, while the MPC and LQR methods are considered only in the nominal cases.

To evaluate the performance of robust IC controllers, we compare between a robust IC controller and an IC controller designed for nominal conditions, i.e. when we design the IC controller, the values of the saturation flow S_z are assumed to be equal to the middle of the uncertain range of S_z. However, the saturation values in the system plant, i.e. the reality, are uncertain, which can randomly vary within the set (3.9), determined by the parameter $\alpha(t)$, see (4.2), where $\alpha = 1$ represents the case of $S_{z,\max}$, $\alpha = 0$ represents the case of $S_{z,\min}$, and $\alpha = 0.5$ represents the nominal case of S_z which is the middle of the uncertainty range.

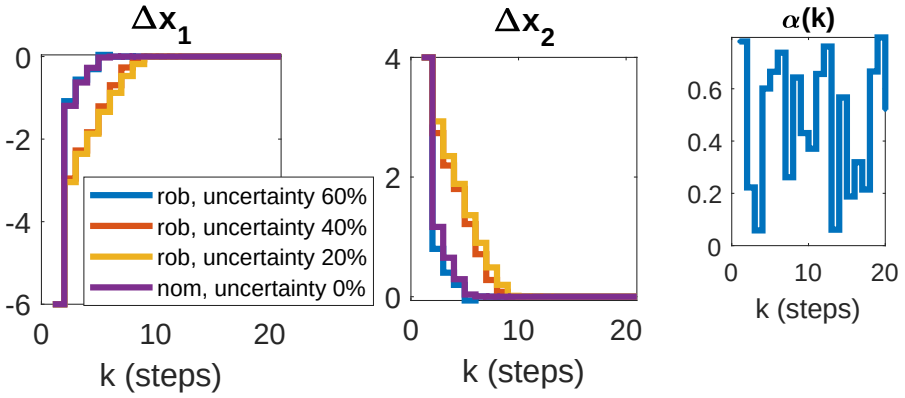
The effect of uncertainty on the invariant sets $\Omega_{\max}$ and C_N is shown in Fig. 7.6(a). As the uncertainty in saturation flows increases, i.e. increasing the range of the set (3.9), the invariant sets $\Omega_{\max}$, C_N become smaller. I.e. larger uncertainty in S_z implies smaller state set for feasible initial queue lengths at the intersection that can be steered to equilibrium queue lengths.

Fig. 7.6(b) shows the trajectory results $\Delta x_1(k)$ and $\Delta x_2(k)$ for a nominal controller, and robust controllers designed for different sizes of uncertainty

set, applied to an uncertain plant which its saturation flows S_z change randomly according to $\alpha(k)$ in the figure. The results show that all controllers succeed to bring the initial queue lengths to the equilibrium lengths even under uncertainties in saturation flows. The results in Fig. 7.6(b) also show that the nominal IC controller can perform well under a small uncertainty compared with the robust controllers, in terms of reaching equilibrium in faster times. For a larger uncertainty, the nominal IC controller gets worse.



(a) Invariant sets for different sizes of uncertainty set, for robust controller.



(b) State evolutions for controllers designed for different sizes of uncertainties, 0% for nominal controller and 20%-60% for robust controllers.

Figure 7.6: Example 2: Effects of different sizes of saturation flow uncertainties.

The results shown in Fig. 7.6(b) are given for the nominal model in (2.1) and the uncertain model in (5.1) with the following parameters and initial state, for each of the n links, and for each of the m phases:

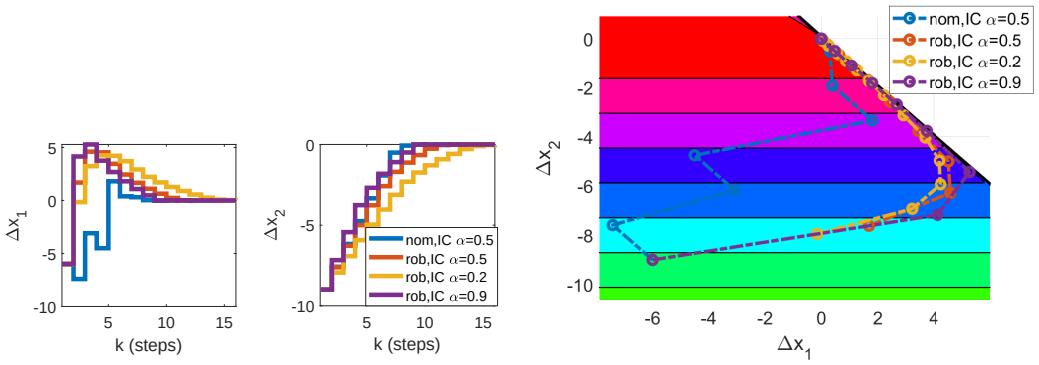
$$\mathbf{x}^{\max} = [40, 66.67]\,, \quad \mathbf{g}^{\min} = [52, 53]\,, \quad \mathbf{g}^{\max} = [59, 56]\,, \quad \mathbf{g}^{\mathrm{N}} = [58, 54]\,,$$
$$\mathrm{un} = 5\%\,, \quad \Delta\mathbf{x}(0) = [-6, 4]\,, \quad n = m = 2.$$

Remark. *The labels "nom" and "rob" respectively describe nominal and robust control.*

The saturation uncertainty set $[S_{z,min}, S_{z,max}]$ is denoted by "un" which is the plus and minus percentages of the nominal value, i.e. in the middle of the set range.

To explore further this observation, for a fixed given size of uncertainty set, we compare the performance results of a nominal IC controller with a robust IC controller, designed accordingly to the given uncertainty set, for three different fixed plant uncertainties, $\alpha = 0.2, 0.5, 0.9$, see Fig. 7.7. The trajectories obtained from applying the nominal controller converge to the origin faster than the robust controllers. The robust controllers seem to be more conservative and cautious not to violate the constraints in expense of faster convergence.

From now until the end of the numerical results, we use fixed plant uncertainties, and not time varying, as in Fig. 7.6, in order to study the effect of the boundaries as representatives of extreme cases (and the mid of them) instead of some random changes within them.



(a) State evolutions for different fixed plant uncertainties under saturation flows uncertain set.

(b) Trajectories for different fixed plant uncertainties under saturation flows uncertainty set.

Figure 7.7: Example 2: Nominal controller vs. Robust controller IC performances, for given uncertainties.

The results shown in Fig. 7.7 are given for the nominal model in (2.1) and the uncertain model in (5.1) with the following parameters and initial state, for each of the n links, and for each of the m phases:

$$\mathbf{x}^{\max} = [40, 66.67]\,, \quad \mathbf{g}^{\min} = [52, 53]\,, \quad \mathbf{g}^{\max} = [59, 56]\,,$$
$$\mathbf{g}^{\mathrm{N}} = [58, 54]\,, \quad \Delta\mathbf{x}(0) = [-6, -9]\,, \quad n = m = 2,\ \mathrm{un} = 40\%.$$

In spite of the well performance of the nominal IC controllers indicated above, i.e. in Fig. 7.6(b) and Fig. 7.7, the results in Fig. 7.8 demonstrate the importance of designing robust controllers.

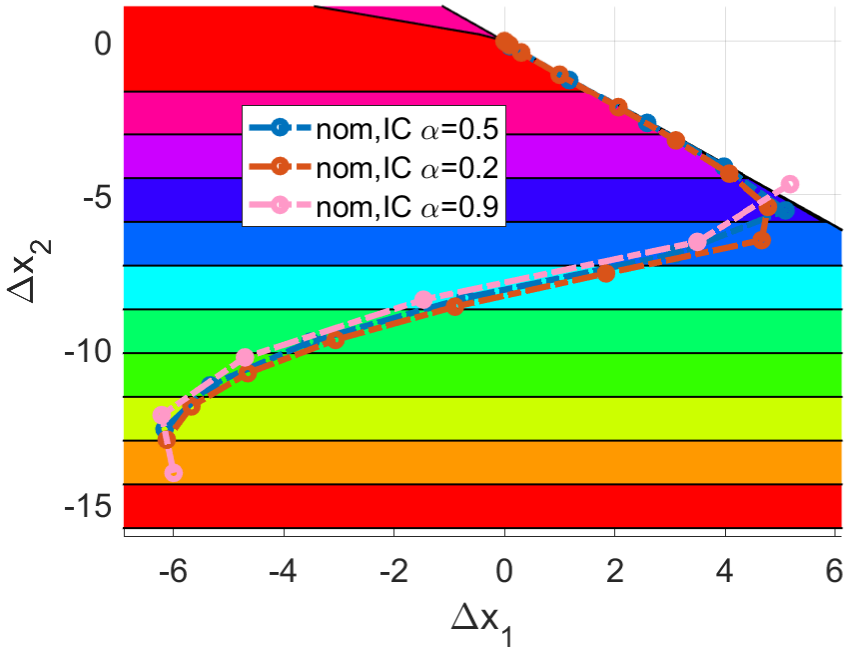


Figure 7.8: Example 2: Trajectories obtained from IC controller, designed based on a nominal model, applied to plant models under different fixed saturation uncertainties.

The results shown in Fig. 7.8 are given for the nominal model in (2.1) with the following parameters and initial state, for each of the n links, and for each of the m phases:

$$\mathbf{x}^{\max} = [40, 66.67]\,, \quad \mathbf{g}^{\min} = [52, 53]\,, \quad \mathbf{g}^{\max} = [60, 57]\,,$$
$$\mathbf{g}^{\mathrm{N}} = [58, 54]\,, \quad \Delta\mathbf{x}(0) = [-6, -14]\,, \quad n = m = 2,\ \mathrm{un} = 40\%.$$

We now explore the effect of mismatch between the system plant model and the control design model. The mismatch is considered as differences between B-matrix values, obtained from different saturation flow values. The IC is designed for nominal conditions, and then applied on three different plants: (i) the plant model is the same as the control design model, and two plants with different fixed saturation uncertainties (ii) $\alpha = 0.2$, and (iii) $\alpha = 0.9$.

The obtained trajectory results are shown in Fig. 7.8. The results show that for large mismatch between B matrices of the plant and control design models, the trajectories might not converge to the origin, see trajectories obtained from IC applied to plant model with $\alpha = 0.9$. As expected and shown in the figure, the nominal controller might fail in bringing the initial state to the the equilibrium state when it is applied for an uncertain plant, e.g. with $\alpha = 0.9$. Such a failure situation would not have happened if a robust controller was applied, see Fig. 7.7(b). The results also show clearly that the step size trajectories moving towards the origin are different for the different plants.

7.3 Numerical example 3: Robust SIC case

In this numerical example, we evaluate the performance of the robust SIC. Similar to the numerical example 2, uncertainty in saturation flows is considered.

Trajectory results of $\Delta x_1(t)$, $\Delta x_2(t)$ over time for nominal SIC and robust SIC are shown in Fig. 7.9. The results show that the nominal SIC and robust SIC trajectories coincide. Recall that if robust IC has different invariant sets Ω_{max}, C_N than Nominal IC, then this should influence both LP problems, and eventually the obtained solutions. On the other hand, in robust SIC, the ellipsoids influence the solution. Since the robust SIC and nominal SIC have the same ellipsoids, this implies similar nominal SIC and robust SIC trajectories.

Note that although the results in Fig. 7.9 are given for a specific uncertainty range, however, the behavior of trajectories that coincide holds for any uncertainty range.

The results shown in Fig. 7.9 are given for the nominal model in (2.1) and the uncertain model in (5.1) with the following parameters and initial state, for each of the n links, and for each of the m phases:

$$\mathbf{x}^{\max} = [40, 66.67]\,, \quad \mathbf{g}^{\min} = [52, 53]\,, \quad \mathbf{g}^{\max} = [59, 56]\,,$$
$$\mathbf{g}^{\mathrm{N}} = [58, 54]\,, \quad \Delta\mathbf{x}(0) = [-6, -9]\,, \quad n = m = 2,\ \mathrm{un} = 40\%.$$

The numerical results show that: (i) the SIC inner and outer ellipsoids in the nominal case are almost identical, (ii) the robust SIC inner and outer are also similar, and (iii) the nominal SIC and robust SIC inner and outer ellipsoids are respectively alike. The reason can be explained while comparing IC to SIC, when we show trajectories both for nominal and robust cases at uncertainty conditions.

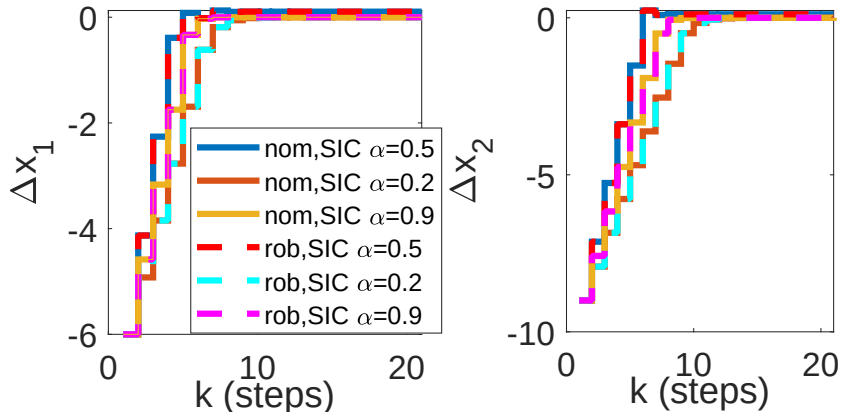


Figure 7.9: Example 3: Robust SIC and SIC trajectories, for an uncertainty range.

The results shown in Fig. 7.10 are given for the nominal model in (2.1) and the uncertain model in (5.1) with the following parameters and initial state, for each of the n links, and for each of the m phases:

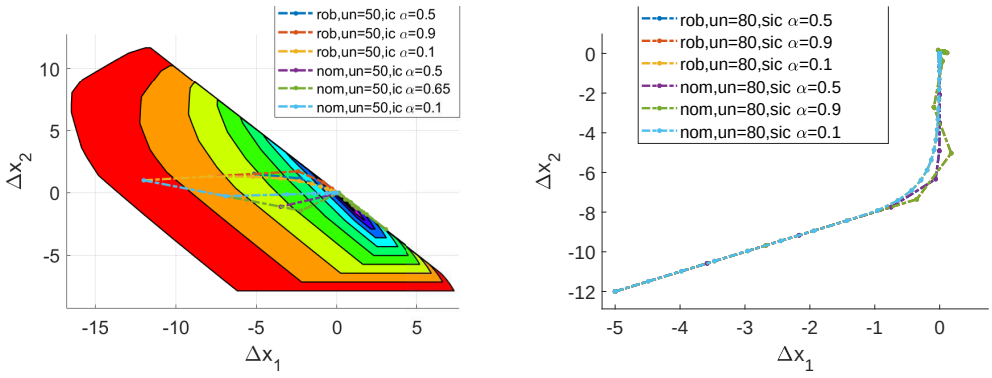
$$\mathbf{x}^{\max} = [40, 66.67]\,, \quad \mathbf{g}^{\min} = [52, 53]\,, \quad \mathbf{g}^{\max} = [59, 56]\,,$$
$$\mathbf{g}^{\mathrm{N}} = [58, 54]\,, \quad \Delta\mathbf{x}(0) = [-12, 1] \text{ for (a) and } [-5, -12] \text{ for (b)}\,, \quad n = m = 2.$$

Similar to Fig. 7.7, different nominal IC and robust IC trajectories obtained are shown in Fig. 7.10(a). The plant model is uncertain, where the saturation uncertainty set $[S_{z,\min}, S_{z,\max}]$ is determined by "un" which is the plus and minus percentages of the nominal value, i.e. in the middle of the set range.

Note that in the simulation runs, the plant model has fixed uncertainty corresponding to α, as was explained in Section 7.2.

Fig. 7.10(b) shows nominal SIC and robust SIC trajectories, zoomed inside the ellipsoids, which as stated before are almost identical for nominal and robust controllers. However, trajectories for different αs differ, especially near the origin as shown in the figure.

But the reason for nominal IC and robust IC being different while nominal SIC and robust SIC being similar is the violation of constraints that occurs in IC case (see Fig. 7.8) and do not occurs in SIC case. When a violation occurs for the nominal controller at uncertainty conditions, then the robust controller must find different trajectory to overcome this violation. This is why nominal IC and robust IC have different invariant sets. And in SIC case, where no violation occurs, then there is no reason to change anything, since



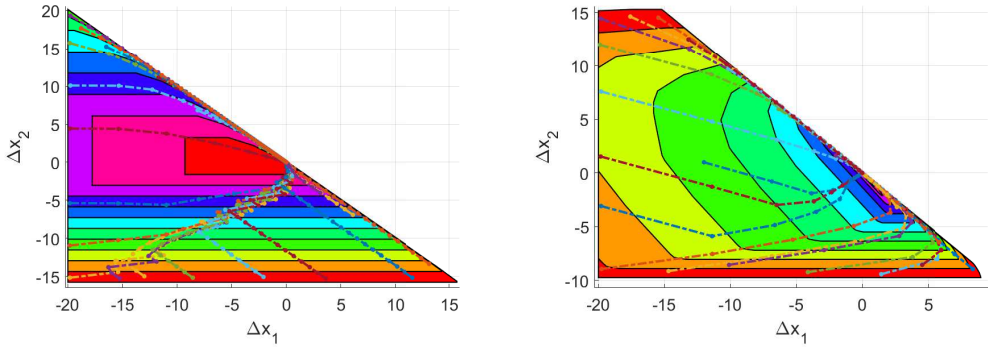
(a) IC trajectories.

(b) SIC trajectories.

Figure 7.10: Example 3: Nominal vs Robust trajectories under uncertainty conditions.

the nominal controller performs better than a robust one, because it is less conservative (has to take less scenarios into account).

As it is already shown in Fig. 7.10(a), in IC case, the nominal trajectories are different from the robust ones, comparing different αs. We further explore the behavior of nominal IC and robust IC trajectories for different uncertainty conditions - different uncertainty range and different αs, as shown in Fig. 7.11.



(a) Nominal IC trajectories, with 60% uncertainty and $\alpha = 0.1$.

(b) Robust IC trajectories, with 40% uncertainty and $\alpha = 0.5$.

Figure 7.11: Example 3: Nominal IC vs Robust IC trajectories, from different initial states.

Note that on the other hand, there are no differences between trajectories of nominal SIC and robust SIC as shown in Fig. 7.12, as concluded earlier.

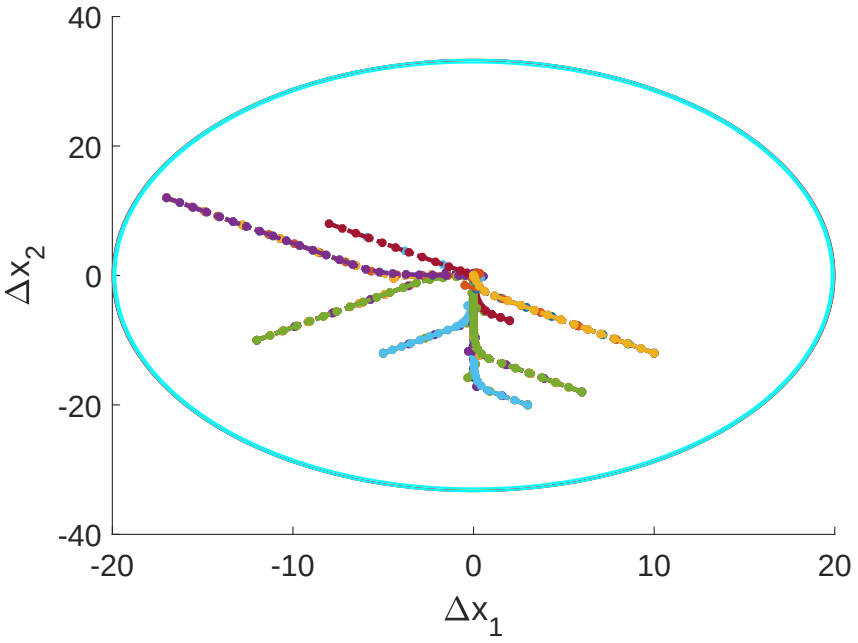
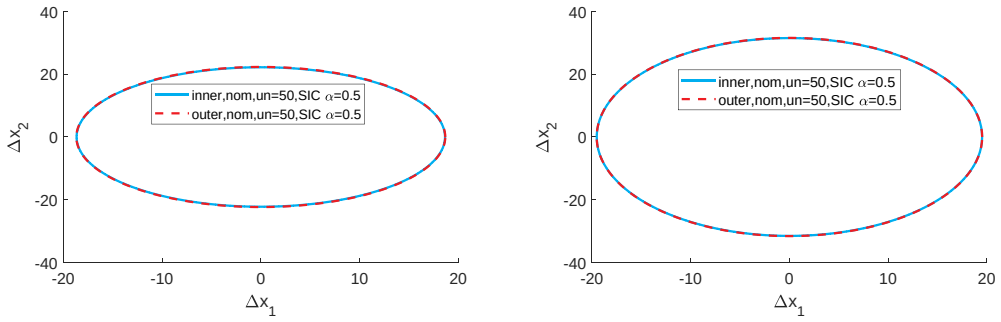


Figure 7.12: Example 3: Nominal SIC vs Robust SIC trajectories, from different initial states.

Finally, a sensitivity analysis is conducted on minimum and maximum control constraints for SIC. Fig. 7.13 shows the effect of changing the range of minimum and maximum control constraints. It is demonstrated that one gets a larger ellipsoidal invariant set when the range of control constraints is larger.

The results shown in Fig. 7.11, Fig. 7.12 and Fig. 7.13 are given for the nominal model in (2.1) and the uncertain model in (5.1) with the following parameters and initial state, for each of the n links, and for each of the m phases:

$$\mathbf{x}^{\max} = [40, 66.67]\,, \quad \mathbf{g}^{\min} = [52, 53]\,, \quad \mathbf{g}^{\max} = [59, 56]\,,$$
$$\mathbf{g}^{\mathrm{N}} = [58, 54]\,, \quad n = m = 2.$$



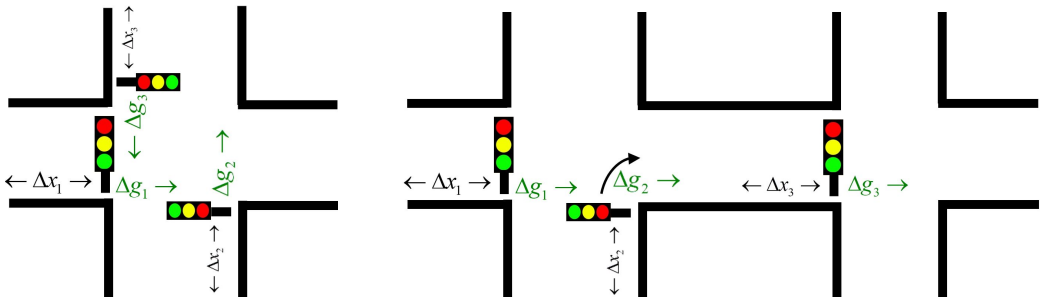
(a) Widering control constraints by ±20%. (b) Widering control constraints by ±40%.

Figure 7.13: Example 3: The effect of changing the limits of the control constraints (3.17), on the inner and outer ellipsoidal invariant sets, for the nominal case.

Chapter 8

Numerical examples for 3-dimensional state space

The performances of the interpolating controllers are examined in this chapter for a 3-dimensional state space. Several applications of traffic signal control can result a 3-dimensional state space problem. In this thesis, we consider (i) an isolated signalized intersection, which has three movements/states and three phases/controllers, as schematically shown in Fig. 8.1(a), and (ii) two interconnected signalized intersections, as schematically shown in Fig. 8.1(b).



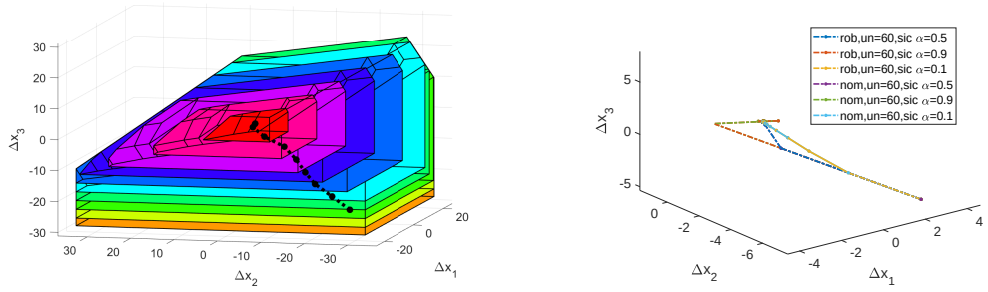
(a) A single intersection. (b) Two interconnected intersections.

Figure 8.1: An illustration of 3-dimensional state space systems.

Two numerical examples are presented: example 4 considers the isolated signalized intersection structure, and example 5 deals with the two interconnected intersections.

The results for Example 4 are shown in Fig. 8.2. A nominal IC trajectory with controlled invariant sets C_N are shown in Fig. 8.2(a), while the SIC nominal and robust trajectories are shown in Fig. 8.2(b). One can compare

with the 2-links intersection in Fig. 7.10(b). SIC's nominal and robust trajectories behave similarly in both cases, and for different alphas trajectories differ only in their ending, near the origin.



(a) (b)

Figure 8.2: Example 4: (a) A nominal IC trajectory with controlled invariant sets C_N, (b) SIC nominal vs. robust trajectories, inside the ellipsoids, under uncertainty conditions.

The results shown in Fig. 8.2 are given for the nominal model in (2.1) and the uncertain model in (5.1) with the following parameters and initial state, for each of the n links, and for each of the m phases:

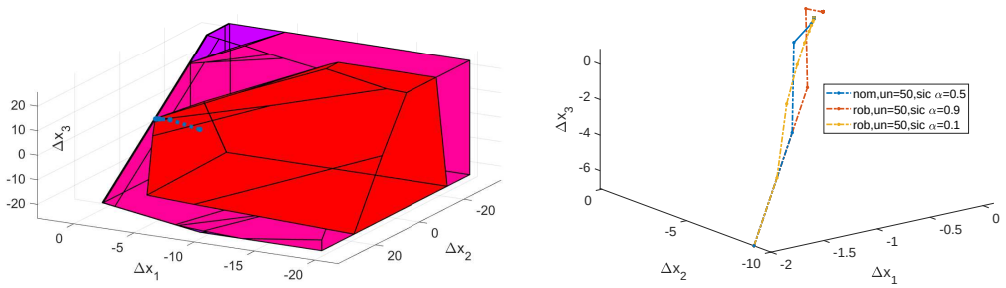
$$\mathbf{x}^{\max} = [46.67, 66.67, 56.67]\,, \quad \mathbf{g}^{\min} = [37, 27, 36]\,, \quad \mathbf{g}^{\max} = [45, 37, 41]\,,$$
$$\mathbf{g}^{\mathrm{N}} = [43, 31, 38]\,,$$
$$\Delta\mathbf{x}(0) = [-21, -29, -21] \text{ for (a) and } [3, -6, -5] \text{ for (b)}\,, \quad n = m = 3.$$

It should be noted that extending the isolated signalized system from 2 states to 3 states does not imply system behaviour changes. In other words, the behaviour is similar, only with 3 links instead of 2.

The results of IC and SIC trajectories for example 5 are shown in Fig. 8.3, where Fig. 8.3(a) shows a nominal IC trajectory with controlled invariant sets C_N, while the SIC nominal and robust trajectories are shown in Fig. 8.3(b).

The results shown in Fig. 8.3 are given for the nominal model in (2.1) and the uncertain model in (5.1) with the following parameters and initial state, for each of the n links, and for each of the m phases:

$$\mathbf{x}^{\max} = [46.67, 66.67, 56.67]\,, \quad \mathbf{g}^{\min} = [37, 27, 36]\,, \quad \mathbf{g}^{\max} = [45, 37, 41]\,,$$
$$\mathbf{g}^{\mathrm{N}} = [43, 31, 38]\,, \quad \Delta\mathbf{x}(0) = [-2, -9, -7] \text{ for (a) and (b)}\,, \quad n = m = 3.$$



(a) (b)

Figure 8.3: Example 5: (a) A nominal IC trajectory with controlled invariant sets C_N, (b) SIC nominal vs. robust trajectories, inside the ellipsoids, under uncertainty conditions.

Note that a robust IC trajectory does not appear in the figure, because already in this case we had no success in computing control invariant set C_N which uses projection. Therefore, we have to use either IIC (improved IC) or meIIC (more efficient IIC) method to overcome this problem, and to avoid calculating C_N explicitly. During investigation we encounter problems using IIC, because of the fact that too many variables and much more constraints resolve an error of using too much memory. Hence, we choose to use meIIC for the robust control from now on.

Finally, comparison between robust meIIC and SIC has been conducted for examples 4 and 5. Trajectories obtained from robust meIIC and robust SIC, under uncertainties of $\pm 30\%$ relatively to the nominal value, are respectively shown for examples 4 and 5 in Fig. 8.4 and Fig. 8.5. The results show that SIC converges faster than meIIC.

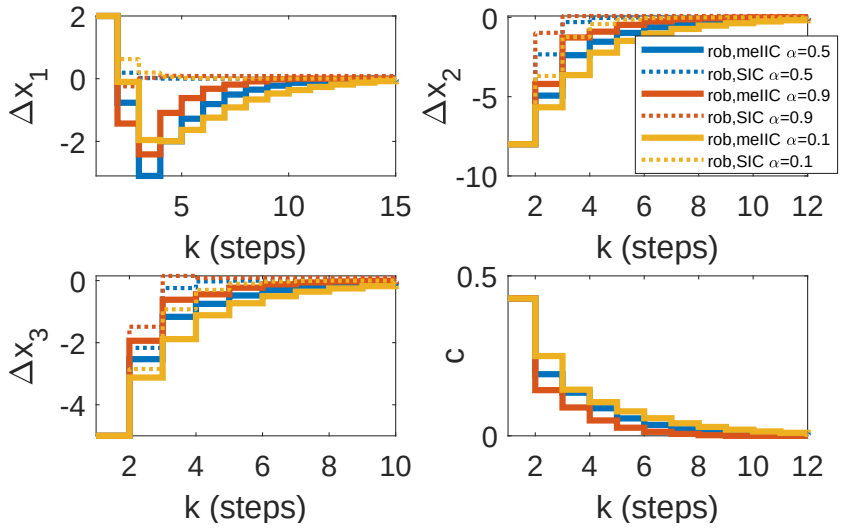


Figure 8.4: Example 4: Robust meIIC vs SIC, under uncertainties of 30%.

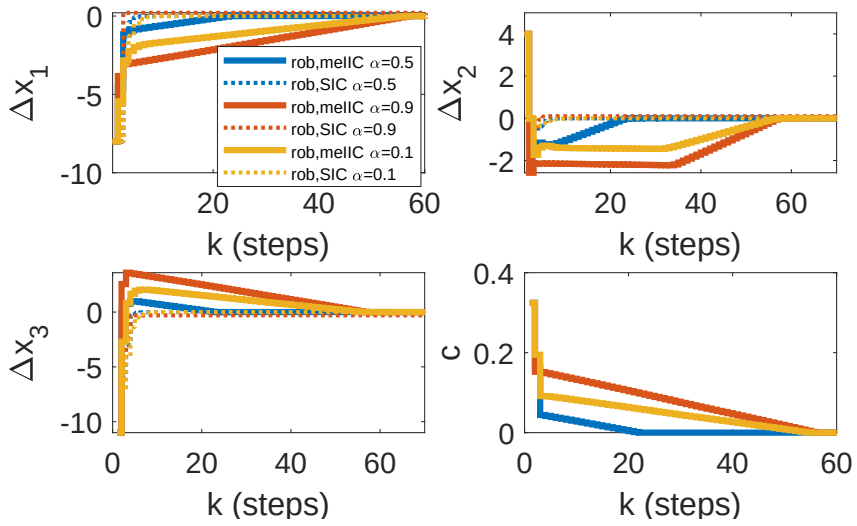


Figure 8.5: Example 5: Robust meIIC vs SIC, under uncertainties of 30%.

The results shown in Fig. 8.4 are given for the uncertain model in (5.1) with the following parameters and initial state, for each of the n links, and for each of the m phases:

$$\mathbf{x}^{\max} = [46.67, 66.67, 56.67]\,, \quad \mathbf{g}^{\min} = [37, 27, 36]\,, \quad \mathbf{g}^{\max} = [45, 37, 41]\,,$$
$$\mathbf{g}^{\mathrm{N}} = [43, 31, 38]\,, \quad \Delta\mathbf{x}(0) = [2, -8, -5]\,, \quad n = m = 3.$$

The results shown in Fig. 8.5 are given for the uncertain model in (5.1) with the following parameters and initial state, for each of the n links, and for each of the m phases:

$$\mathbf{x}^{\max} = [40, 66.67, 46.67]\,, \quad \mathbf{g}^{\min} = [32, 43, 67]\,, \quad \mathbf{g}^{\max} = [69, 76, 119]\,,$$
$$\mathbf{g}^{\mathrm{N}} = [58, 54, 112]\,, \quad \Delta\mathbf{x}(0) = [-8, 4, -11]\,, \quad n_1 = m_1 = 2,\ n_2 = m_2 = 1.$$

Note that it was shown in Section 7.1, see Fig. 7.5, that IC performs better than SIC. However, as the state and control dimensions of the problem get lager, then the calculation of IC becomes more computationally heavy. Hence, meIIC method is a suitable alternative of IC, which is on the one hand calculates faster than IC, but on the other hand it produces less optimal solution with a slower convergence, even more than SIC.

As mentioned in Section 4.4.1, meIIC converges slowly or equivalently compared to IIC or IC, respectively. This is shown for the 2-links case in Fig. 8.6.

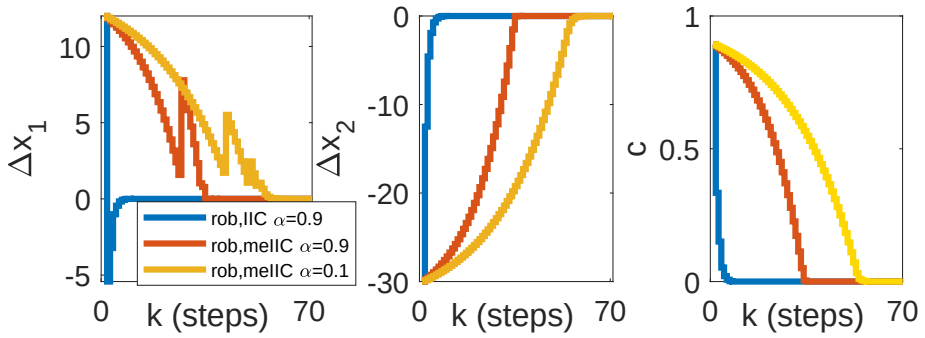


Figure 8.6: Robust IIC compared to robust meIIC, under uncertainties of 30%, in 2-links case

The results shown in Fig. 8.6 are given for the uncertain model in (5.1) with the following parameters and initial state, for each of the n links, and for each of the m phases:

$$\mathbf{x}^{\max} = [40, 66.67]\,, \quad \mathbf{g}^{\min} = [32, 43]\,, \quad \mathbf{g}^{\max} = [69, 76]\,,$$
$$\mathbf{g}^{\mathrm{N}} = [58, 54]\,, \quad \Delta\mathbf{x}(0) = [12, -30]\,, \quad n = m = 2.$$

The performance of LQR, MPC, and IC for example 5 are compared considering nominal case for different initial states in Fig. 8.7. The performance was evaluated by computing the values of the quadratic objective

function of LQR, see (2.2). The y-axis in Fig. 8.7 represents the fraction between the objective function value J_i of a specific method and the sum of objective values obtained from all methods, i.e. sum of LQR, MPC, and IC: $\sum_{i \in \text{Methods}} J_i$, Methods={LQR,MPC,IC}. E.g., a lower fraction denotes a better performance.

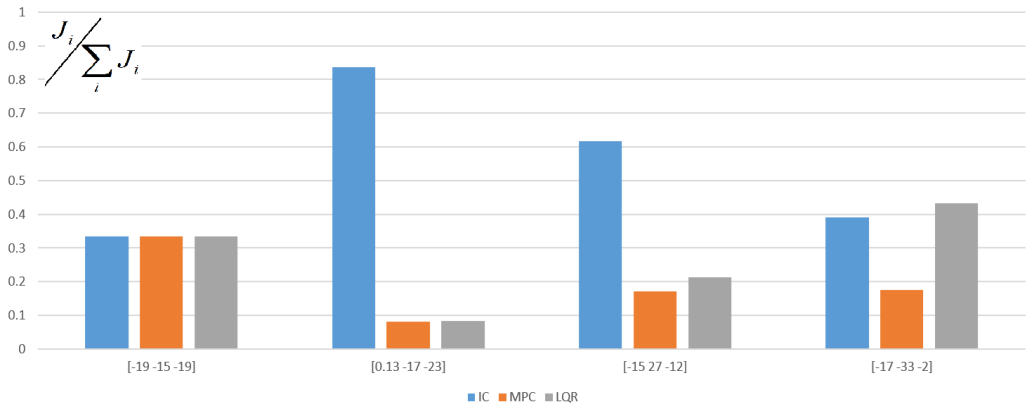


Figure 8.7: Example 5: Relative performance comparison between Nominal LQR, MPC, and IC methods for different initial states.

As shown, some initial states, e.g. $[-19\ -15\ -19]$, have similar objective values, while for other initial states, e.g. $[0.13\ -\ 17\ -\ 23]$, the IC has a very poor performance. The latter holds since the initial state is near the border of the C_N outer invariant set, where the interpolating coefficient c starts very high near the value 1, as shown in Fig. 8.8. Moreover, in other cases, e.g. [-15 27 -12], IC does not perform optimally, and MPC is more (or as) optimal than LQR, since probably these are the cases where constraints applied after control law and ruin optimality (see Appendix A.2). In some other cases [-17 -33 -2] IC performs better than LQR, again from the same reason that we cannot anticipate the amount of effect on optimality when applying constraints after control law. Hence, even though the LQR is more optimal than the IC in this example, after applying the constraints, LQR gets either to be more or less optimal compared to IC.

The results shown in Fig. 8.7 and Fig. 8.8 are given for the nominal model in (2.1) with the following parameters and initial state, for each of the n links, and for each of the m phases:

$$\mathbf{x}^{\max} = [40, 66.67, 46.67]\,, \quad \mathbf{g}^{\min} = [39, 45, 77]\,, \quad \mathbf{g}^{\max} = [69, 65, 119]\,,$$
$$\mathbf{g}^{\mathrm{N}} = [58, 54, 112]\,, \quad n_1 = m_1 = 2,\ n_2 = m_2 = 1.$$

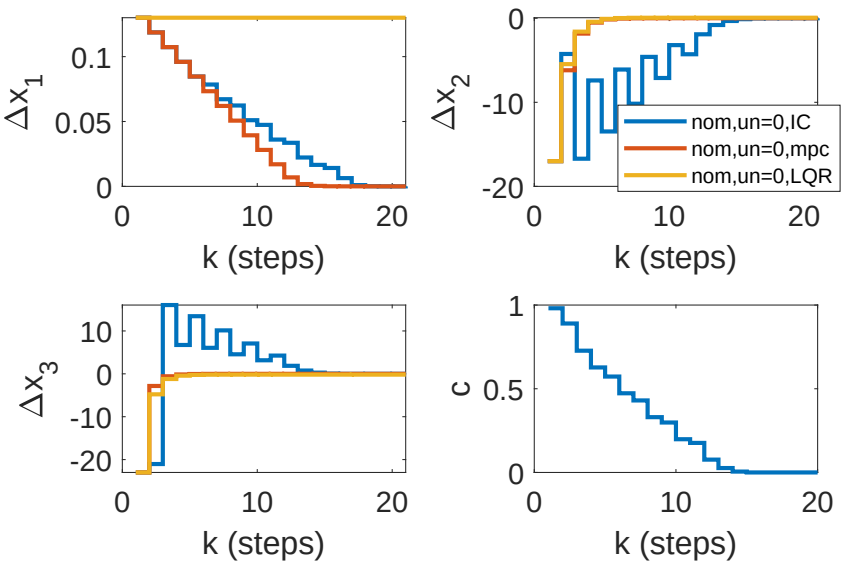


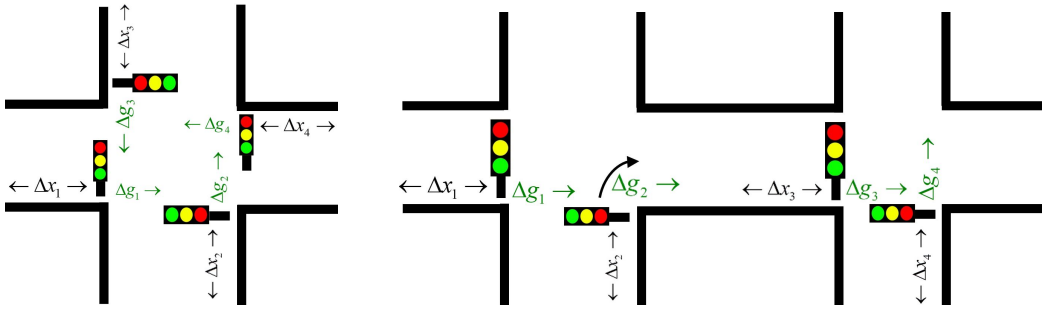
Figure 8.8: Example 5: Trajectories comparison between Nominal LQR, MPC, and IC methods for the initial state [0.13, -17, -23].

Note that results of SIC are not shown, since SIC has a very small feasible set (near origin), in which for any feasible initial state, there is no difference between all the four methods.

Chapter 9

Numerical examples for 4-dimensional state space

The performances of the interpolating controllers are examined in this chapter for a 4-dimensional state space. Several applications of traffic signal control can result a 4-dimensional state space problem. In this thesis, we consider (i) an isolated signalized intersection, which has four movements/states and four phases/controllers, as schematically shown in Fig. 9.1(a), and (ii) two interconnected signalized intersections, as schematically shown in Fig. 9.1(b).



(a) A single intersection. (b) Two interconnected intersections.

Figure 9.1: An illustration of 4-dimensional state space systems.

Two numerical examples are presented: example 6 considers the isolated signalized intersection structure, and example 7 deals with the two interconnected intersections.

9.1 Nominal case

In example 6, a single intersection with 4 links and with wide range of symmetrical state and control constraints is considered. Example 6 is tested for two types of initial state, one close and one far from the origin. The trajectory results obtained from LQR, MPC, IC, SIC, and meIIC for an initial state which is close to the origin are shown in Fig. 9.2, while the results in Fig. 9.3 are for an initial state which is far from the origin. Both results are shown for nominal case without uncertainties. As shown, all trajectories coincide, which implies that all methods act the same when the initial state is close to the origin. However, if the initial state is far from the origin, then SIC is the first to diverge, since it has the smallest feasible range. In the latter case, it is shown in Fig. 9.3 that even though the initial state is inside the MAS of the IC, i.e. $c = 0$, SIC diverges, since SIC has a different invariant sets compared to IC, see Fig. 5.1, while the other methods LQR and MPC converge, similarly. Also IC's MAS considers all constraints while SIC's MAS considers only range constraints.

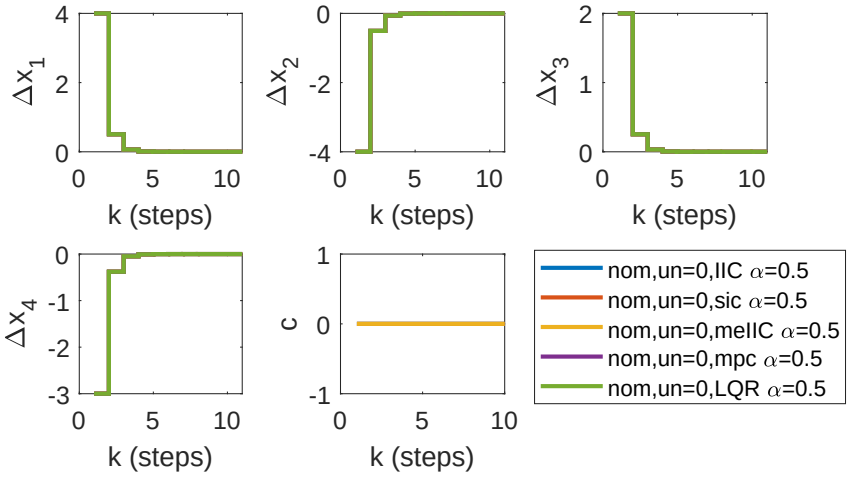


Figure 9.2: Example 6: initial state [4, -4, 2, -3] nearby the origin, with wide constraints.

The results shown in Fig. 9.2 and Fig. 9.3 are given for the nominal model in (2.1) with the following parameters and initial state, for each of the n links, and for each of the m phases:

$$\mathbf{x}^{\max} = [66.67, 66.67, 66.67, 66.67]\,, \quad \mathbf{g}^{\min} = [6, 6, 6, 6]\,, \quad \mathbf{g}^{\max} = [44, 44, 44, 44]\,,$$
$$\mathbf{g}^{\mathrm{N}} = [28, 28, 28, 28]\,, \quad n = m = 4.$$

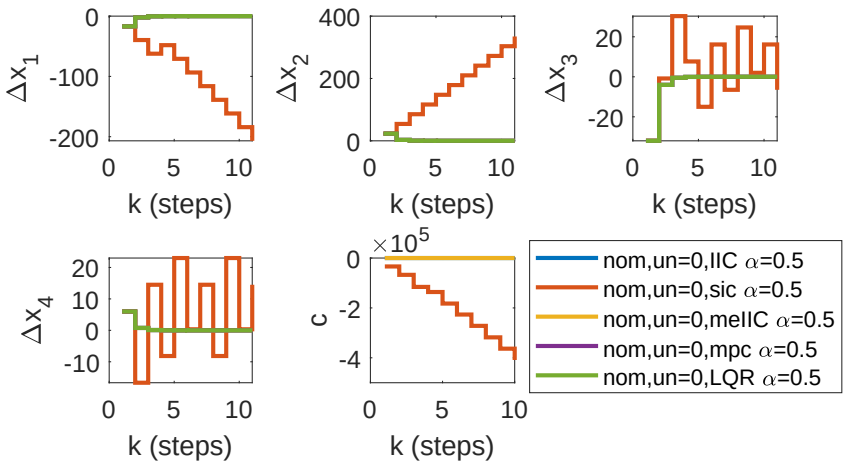


Figure 9.3: Example 6: initial state [-17, 23, -32, 6] is far from the origin, with wide constraints.

From now on until the end of this chapter, we change ranges of constraints to be asymmetrical and narrow.

The results for an initial state which is close to the origin, with *narrow* constraints are shown in Fig. 9.4. It is shown in Fig. 9.4 that there is no meaningful difference between the methods, though unlike Fig. 9.2, for the same initial state we have $c > 0$, since in the wider constraints the MAS region is larger. Therefore, $c = 0$ holds for a larger region. But narrowing constraints yields a smaller region, such that the initial point now is outside the MAS, and $c > 0$.

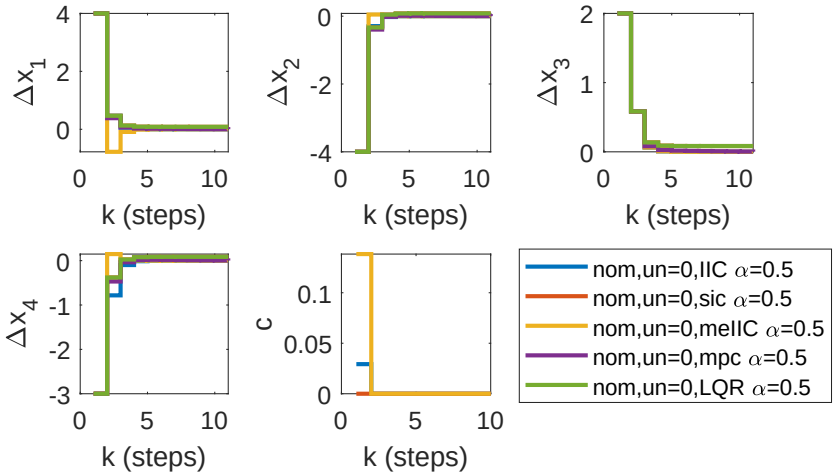


Figure 9.4: Example 6: initial state [4, -4, 2, -3] is close to the origin, with narrow constraints.

However, when the initial state is different but close to the origin, see Fig. 9.5, and also is outside the MAS region, then the LQR and SIC converge but not to the origin. Note that convergence of LQR and SIC is not guaranteed as shown in Fig. 9.6, SIC diverges for another initial state which is farther from the origin.

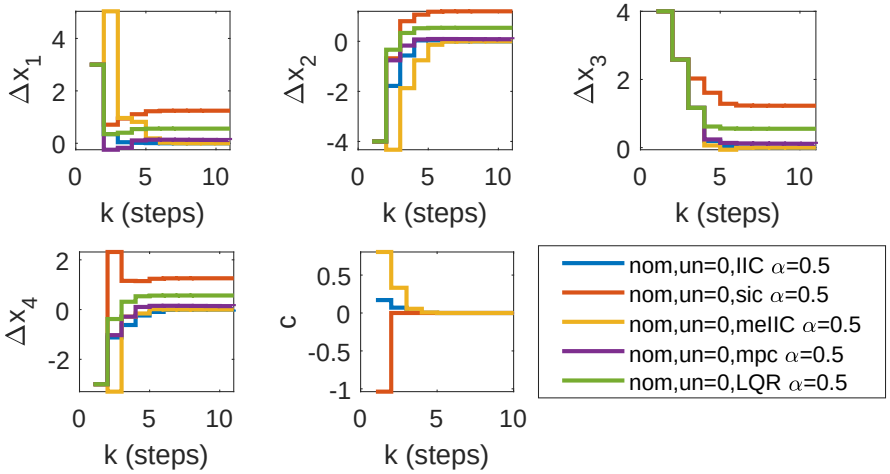


Figure 9.5: Example 6: initial state [3, -4, 4, -3] is close to the origin, with narrow constraints.

The results for the same initial state as in Fig. 9.6, which is far from the

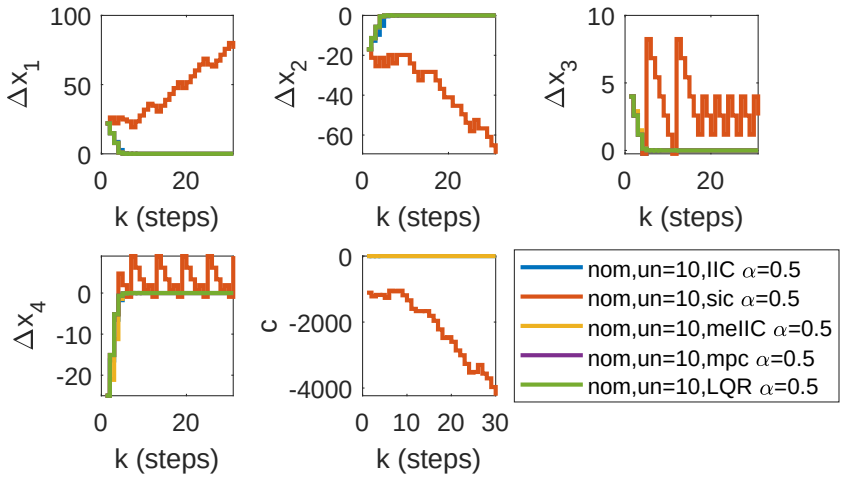


Figure 9.6: Example 6: initial state [22, -17, 4, -25] is farther from the origin, with narrow constraints.

origin, only now under uncertainties, are shown in Fig. 9.7, and again SIC diverges, while IIC and meIIC have similar behavior.

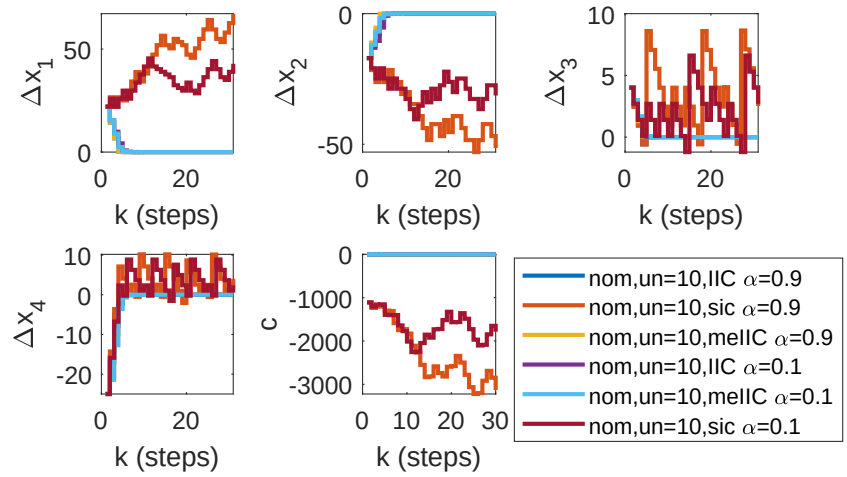


Figure 9.7: Example 6: Comparing Nominal IIC, meIIC, and SIC under uncertainties: initial state [22, -17, 4, -25] is far from the origin.

We also examine an initial state near the origin, as shown in Fig. 9.8. The results show that IIC performs better than meIIC. The figure excludes SIC as it diverges.

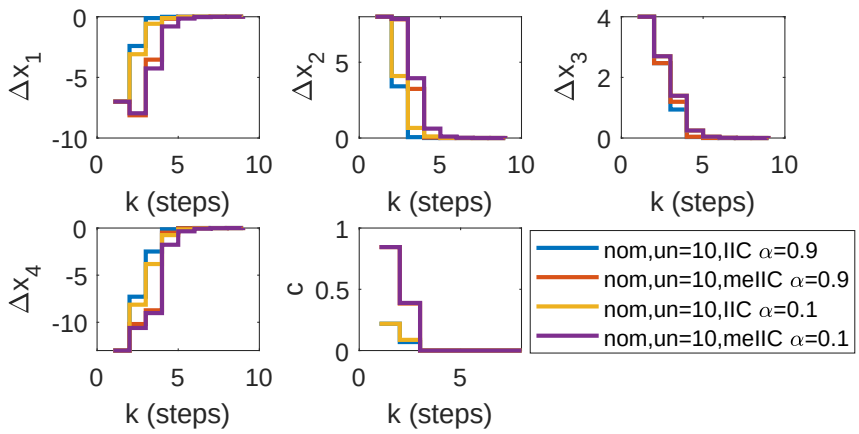
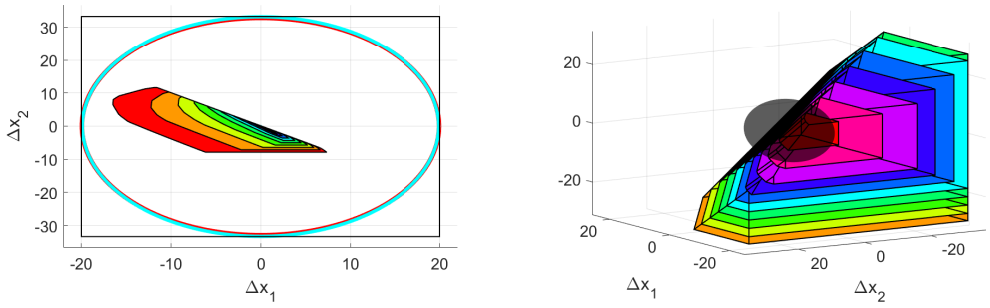


Figure 9.8: Example 6: Comparing Nominal IIC and meIIC under uncertainties: initial state [-7, 8, 4, -13] is near the origin.

The results shown in Fig. 9.4, Fig. 9.5, Fig. 9.6, Fig. 9.7 and Fig. 9.8 are given for the nominal model in (2.1) with the following parameters and initial state, for each of the n links, and for each of the m phases:

$$\mathbf{x}^{\max} = [60, 40, 53.33, 66.67]\,, \quad \mathbf{g}^{\min} = [25, 24, 22, 21]\,, \quad \mathbf{g}^{\max} = [33, 31, 29, 30]\,,$$
$$\mathbf{g}^{\mathrm{N}} = [28, 28, 28, 28]\,, \quad n = m = 4.$$

Another evidence for the ellipsoidal invariant sets being shrinked with additional dimensions can be seen in Fig. 9.9, when we move from 2D to 3D.



(a) 2D similar case as Fig. 7.10(a). (b) 3D similar case as Fig. 8.2(a).

Figure 9.9: Change of ellipsoidal invariant set comparing to IC invariant sets, going from 2D Example 3 to 3D Example 4.

The black frame in Fig. 9.9(a) represents the state constraints, and the blue and red ellipsoids are the inner and outer SIC invariant sets, respectively.

The ellipsoids are bounded by state constraints, since in this particular example, the control range constraints are less restrictive compared to the state range constraints. The black ellipsoid in Fig. 9.9(b) is the inner (same as outer) SIC invariant set. Unfortunately, for 4D we cannot show the invariant sets of IC and SIC.

9.2 Robust case

In the robust case, we continue from where we ended in the previous section of the nominal case, comparing two types of IC methods: IIC and meIIC, again under different uncertain conditions, but this time using a robust controller.

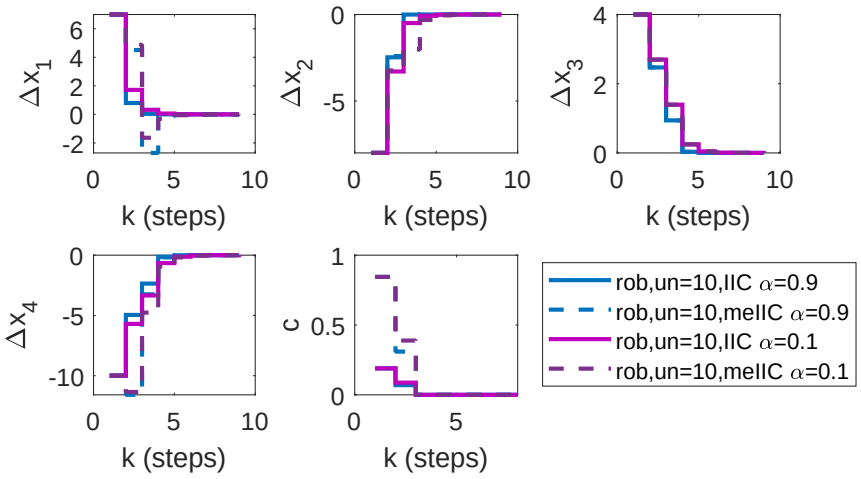


Figure 9.10: Example 6: Comparing robust IIC and meIIC under uncertainties: initial state [7, -8, 4, -10] is far from the origin.

The results shown in Fig. 9.10 and Fig. 9.11 are given for the uncertain model in (5.1) with the following parameters and initial state, for each of the n links, and for each of the m phases:

$$\mathbf{x}^{\max} = [60, 40, 53.33, 66.67]\,, \quad \mathbf{g}^{\min} = [25, 24, 22, 21]\,, \quad \mathbf{g}^{\max} = [33, 31, 29, 30]\,,$$
$$\mathbf{g}^{\mathrm{N}} = [28, 28, 28, 28]\,, \quad n = m = 4.$$

It can be concluded that IIC has a good performance in terms of convergence as shown in Fig. 9.10, similar to the nominal case as shown in Fig. 9.8. So from now until the end of this chapter we will use IIC to represent robust IC. But for the full integrity, there were some cases where some of the

links were less converging in IIC comparing to meIIC, especially when the interpolating coefficient c of the initial state is high.

Now, we compare between robust IIC and SIC, but for that we need to start near the origin, in order to guarantee the convergence of SIC. As shown in Fig. 9.11, SIC converges faster comparing to IIC. But this happens only in a very tight region around the origin.

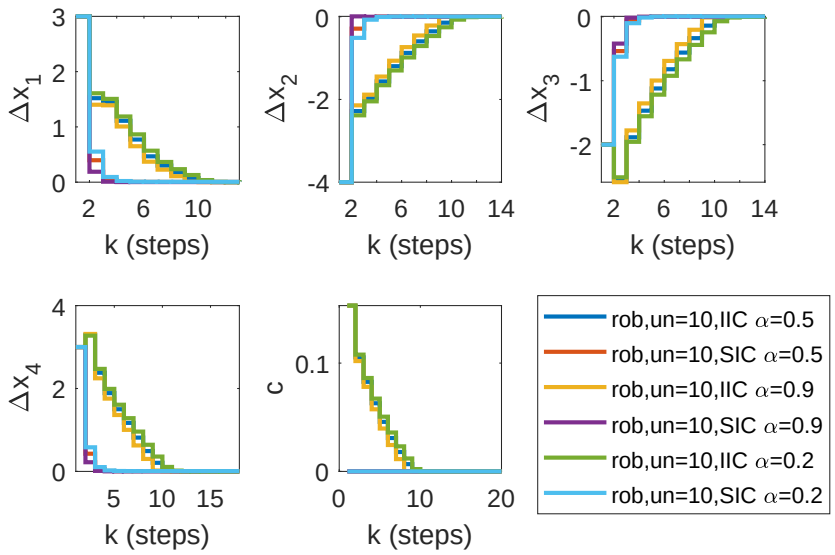
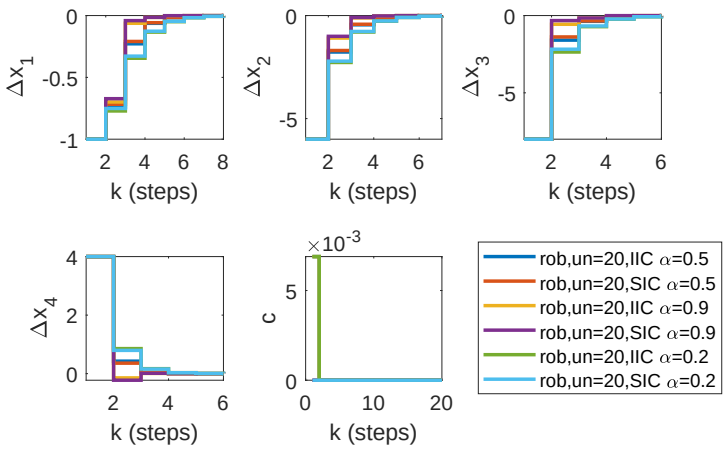


Figure 9.11: Example 6: Comparing robust IIC and SIC under uncertainties: initial state [3, -4, -2, 3] is near the origin.

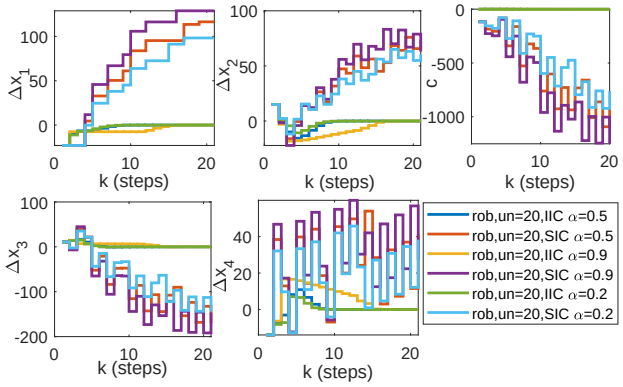
Now we turn to the more complex system of two junctions interconnected, as shown in Fig. 9.1(b), referred as Example 7. As in robust case, we analyse initial states near and far from the origin.

As shown in Fig. 9.12(a) when the initial state is near the origin, IC and SIC behaves pretty similarly. But farther from the origin, as shown in Fig. 9.12(b), SIC diverges while IC converges, since IC probably has a larger feasible region, see Section 5.1.

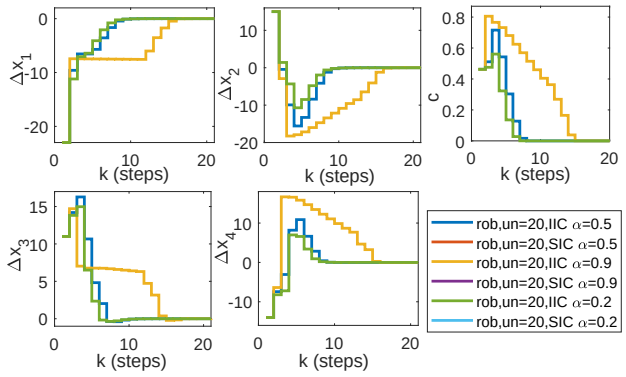
If one looks closer, in the case far from origin, as shown in Fig. 9.12(c), then one can see that IC behavior under uncertainty is chaotic and has slow convergence, which might be due to the more complex/strict case of Example 7, compared to Example 6.



(a) Initial state [-1, -6, -8, 4] near the origin.



(b) Initial state [-23, 15, 11, -14] far from the origin: full plot.



(c) Initial state [-23, 15, 11, -14] far from the origin: zoom-in plot.

Figure 9.12:

Example 7: Comparing robust IIC and SIC under uncertainties.

The results shown in Fig. 9.12 are given for the uncertain model in (5.1) with the following parameters and initial state, for each of the n links, and for each of the m phases:

$$\mathbf{x}^{\max} = [56.67, 46.67, 66.67, 43.33]\,, \quad \mathbf{g}^{\min} = [32, 43, 28, 31]\,, \quad \mathbf{g}^{\max} = [69, 76, 68, 86]\,,$$
$$\mathbf{g}^{\mathrm{N}} = [58, 54, 47, 68]\,, \quad n = m = 4.$$

Remark. *It should be mentioned, that there is a big problem to find explicit initial space from which one can start, for each of the methods (SIC, IIC, meIIC), for 4 or higher dimensional cases. All methods have different feasible space regions.*

Chapter 10

Conclusions and future work

In this thesis, an uncertain store-and-forward model has been developed as parameter uncertainties were integrated in the queuing dynamics for traffic signalized intersections. Two variants of interpolation-based controller have been designed. The developed IC and SIC controllers guarantee robustness against all parameter uncertainties, and treat control and state constraints directly in the design phase. As far as the authors know, this is the first time that the interpolation-based approach is explored to design traffic control for signalized intersections. Comparison of numerical results between the developed interpolating controllers and other controllers in the literature for an isolated signalized intersection and for two interconnected intersections demonstrate the performance advantages from applying the robust interpolating controllers.

A few points can be concluded from the results of this research:

- **Effect of the range constraints.** Widening the range constraints, i.e. control constraints (3.17) or state constraint (3.16) or both, would result a larger MAS Ω_{max} and a larger control invariant set C_N.
- **Imposing equality sum constraint** implies a feasible line set. If the initial state belongs to the feasible set, then the state converges to the origin.
- **MPC and LQR comparison.** Applying LQR and MPC controllers to our traffic signal problem show that both controllers operate the same when the state and control constraints are not active. However, when one of the constraints becomes active, then the LQR converges, but not to the origin.
- **Nominal IC and Robust IC comparison.** Some results show that the Nominal IC controller can perform better than the Robust IC con-

troller, as the Nominal IC might converge faster to the origin than the Robust IC for the same plant uncertainties. However, in most cases, the Robust IC controller has advantages in handling uncertainties compared with the Nominal IC.

- **Nominal SIC and Robust SIC comparison.** It is shown that for our traffic control problem, the inner and outer ellipsoidal invariant sets of Nominal SIC are similar, and those of the Robust SIC are also similar. In fact, both ellipsoidal invariant sets of Nominal SIC and Robust SIC are also similar.

- **Nominal IC and Robust IC vs. Nominal SIC and Robust SIC invariant sets comparison.** In Nominal SIC and Robust SIC, the control inputs depend only on the ellipsoidal sets. Therefore, if the Nominal SIC and Robust SIC sets are similar, then obtained trajectories form Nominal SIC and Robust SIC will be identical for the same plant uncertainty. On the other hand, in Nominal IC and Robust IC, the controller depends, not only, on the invariant sets. Therefore, the obtained trajectories might not be similar if the invariant sets of Nominal IC and Robust IC are different.

- **Computational complexity in IC.** When the dimensionality of the problem increases, calculating the control invariant set C_N might be an issue. In such cases, one can utilize either IIC or meIIC, where IIC can be still computationally expensive sometimes. Indeed, meIIC has a low computational complexity, but its flaw that the trajectories converges slowly, even slower than SIC.

- **IC and SIC comparison.** SIC invariant sets are more conservative than IC invariant sets, due to tighter constraints caused by the ellipsoidal shaped invariant sets, which represent less accurately the invariant sets compared to polyhedral invariant sets in IC. Hence, in complex problems, such as in 4-dimensional case, it frequently occurs that the calculated SIC ellipsoidal sets are so small around the origin, that they are practically useless.

  Moreover, in SIC, some constraints applied after the control law, which might ruin optimality.

Finally, the conclusions are compactly summarized in Fig. 10.1. Fig. 10.1 shows the trade-off between computation load and optimality or convergence rate. For instance, MPC is most computationally heavy but the optimal one between all methods, on the other hand meIIC is easily implemented

with the least computational effort but it is least optimal and slowest in convergence. It should be stressed that this figure is based on the presented traffic signal problem and the results of the numerical examples obtained in this thesis, and should be further checked before generalizing it to other problems domain.

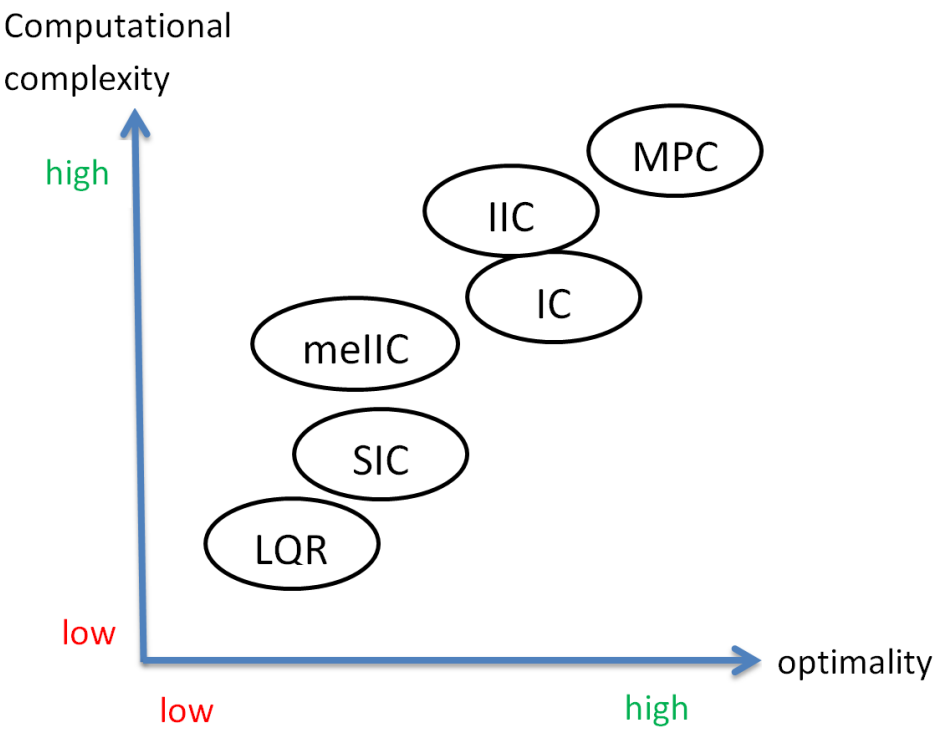


Figure 10.1: A roughly estimation of the comparison of the methods, as concluded and confirmed from the traffic signal problem results.

10.1 Contributions

The main contributions of the current thesis are summarized as follows:

- Modeling an urban road network in the form of an uncertain discrete-time linear model, which includes new type of uncertainty, with state and input constraints.
- Implementing different interpolation-based control laws on the traffic signal control problem, among them the recently introduced in the literature Simple Interpolating Control.

- Comparing between several control approaches (LQR, MPC, IC, SIC) via numerical results analysis.

10.2 Future work

Scalability analysis of implementing the traffic interpolating controllers to large-scale urban networks should be studied. Even though the current thesis presents numerical study on simple structures of signalized intersections, a more complex structure of isolated intersections and/or arterials with several intersections can be considered, where a decentralized control is implemented for the whole network, with IC or SIC implemented locally, for each individual intersection.

The current thesis focused on investigating the effect of saturation flow uncertainty. Future research will be dedicated to study all other parameter uncertainty that have been mentioned in the thesis, such as demand parameter, turning rates, and exit rates.

The problem of small invariant sets of SIC or the inability to use IC for complex problems, can be dealt by using a decentralized control over a network, and using either SIC or IIC or meIIC, at each junction, where the number of links is limited and usually small enough.

Appendix A

A.1 Nominal LQR and MPC design matrices

Here we transform the conservation equation for each link (3.4), from Section 3.2

$$x_z(k+1) = x_z(k) + T[q_z(k) - s_z(k) + d_z(k) - u_z(k)], \tag{A.1}$$

into the vector equation linear state-space form of deviated variables (3.6):

$$\Delta\mathbf{x}(k+1) = A\Delta\mathbf{x}(k) + B\Delta\mathbf{g}(k) + T\Delta\mathbf{d}(k)\,. \tag{A.2}$$

It is also shown in detail, the weighting matrices in the objective function of LQR and MPC, in Section 2. For the sake of better visualization, it is presented here in this appendix the N-step quadratic objective function starting from initial step 0, instead of k. All parameters in the matrices below are defined in Section 3.2.

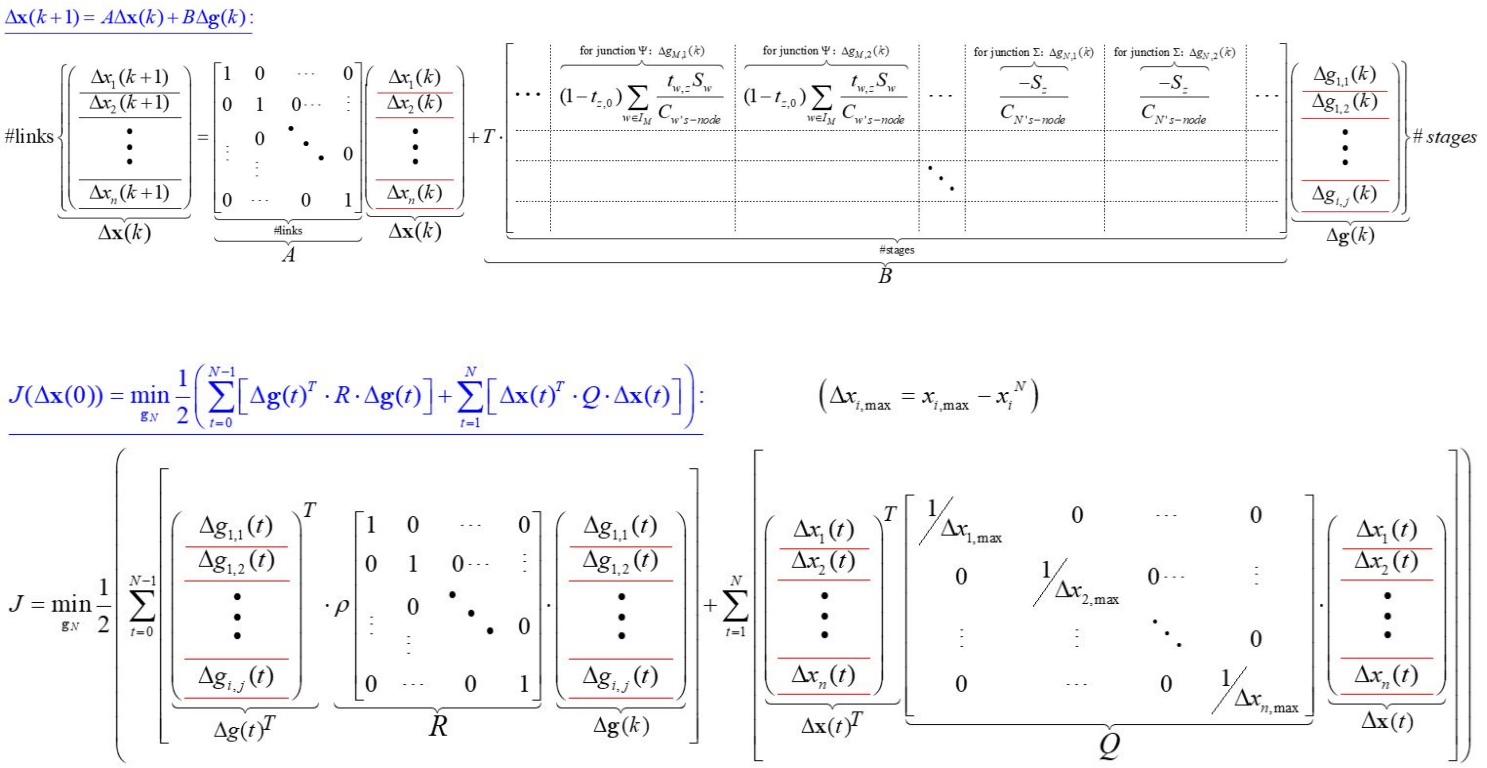


Figure A.1: SF model (A,B) and cost function (Q,R) matrices.

A.2 QP for obtaining feasible control

For methods that do not include control constraints (3.14), (3.15) in their control law, such as LQR and SIC, we use QP algorithm, see [25], to find most closest control signal to the unconstrained one. This control now will be constrained. But this method deals only with control constraints, not with state constraints. And we optimize the control signals actual values and not the deviating values respect to nominal values. So we use $\Delta\mathbf{x}(k) = \mathbf{x}(k) - \mathbf{x}^{\mathrm{N}}$, $\Delta\mathbf{g}(k) = \mathbf{g}(k) - \mathbf{g}^{\mathrm{N}}$, to obtain $\mathbf{x}(k), \mathbf{g}(k)$. The QP problem can either minimize the absolute difference between a given unconstrained control signal $\mathbf{g}(k)$ and the constraint control signal $\tilde{\mathbf{g}}(k)$ (which is the variable in the problem), or minimize the relative (or weighted absolute) difference between those two values:

Absolute difference objective

$$J_{\mathrm{abs}} = \frac{1}{2}\sum_{i\in F_j}(\tilde{g}_{j,i} - g_{j,i})^2 \to \min \tag{A.3}$$

Relative difference objective

$$J_{\mathrm{rel}} = \frac{1}{2}\sum_{i\in F_j}\frac{(\tilde{g}_{j,i} - g_{j,i})^2}{g_{j,i}} \to \min \tag{A.4}$$

A more detailed formulation of these methods:

$$J_{\mathrm{rel}} = \frac{1}{2}\sum_{i\in F_j}\frac{(\tilde{g}_{j,i} - g_{j,i})^2}{g_{j,i}} = \frac{1}{2}\left[\tilde{g}_{j,i}\cdot\frac{1}{g_{j,i}}I\cdot\tilde{g}_{j,i}^{\,T} - 2\cdot\tilde{g}_{j,i} + \underbrace{\cancel{g_{j,i}}}_{\substack{\text{constant}\\ \text{for minimization}}}\right]$$

$$J_{\mathrm{abs}} = \frac{1}{2}\sum_{i\in F_j}(\tilde{g}_{j,i} - g_{j,i})^2 = \frac{1}{2}\left[\tilde{g}_{j,i}\cdot I\cdot\tilde{g}_{j,i}^{\,T} - 2g_{j,i}\cdot\tilde{g}_{j,i} + \underbrace{\cancel{\left(g_{j,i}\right)^2}}_{\substack{\text{constant}\\ \text{for minimization}}}\right]$$

The idea behind the relative objective function is simple: for a large unconstrained control $\mathbf{g}(k)$ we do not mind that the absolute difference will be small, but rather only the relative difference, while if $\mathbf{g}(k)$ is small we would prefer a bigger weight to make the difference small as possible. Meaning eventually, that when take all control signals into account, we'd like to minimize more strongly the small signals' difference (also from security point of view) at the expense of the larger signals (or the longer green lights).

In this thesis, an absolute difference objective is implemented (A.3), as shown in Algorithm 12, where the arguments in QP solver are respectively: the quadratic matrix, the linear matrix and the constraints.

Algorithm 12 Calculation of feasible control $\Delta\tilde{\mathbf{g}}(k)$

Input: the optimal unconstrained vector $\Delta\mathbf{g}(k)$, the set $\mathcal{U}$.

1: $\mathbf{g}(k) \leftarrow \Delta\mathbf{g}(k) + \mathbf{g}^{\mathrm{N}}$
2: $\tilde{\mathbf{g}}(k) \leftarrow \mathrm{QP}\,(I,\, -2g_{j,i},\, \mathcal{U})$
3: $\Delta\tilde{\mathbf{g}}(k) \leftarrow \tilde{\mathbf{g}}(k) - \mathbf{g}^{\mathrm{N}}$

When there is the necessity to apply constraints after control law, then the original implementation shown in Fig. 6.1 is changed into the one in Fig. A.2.

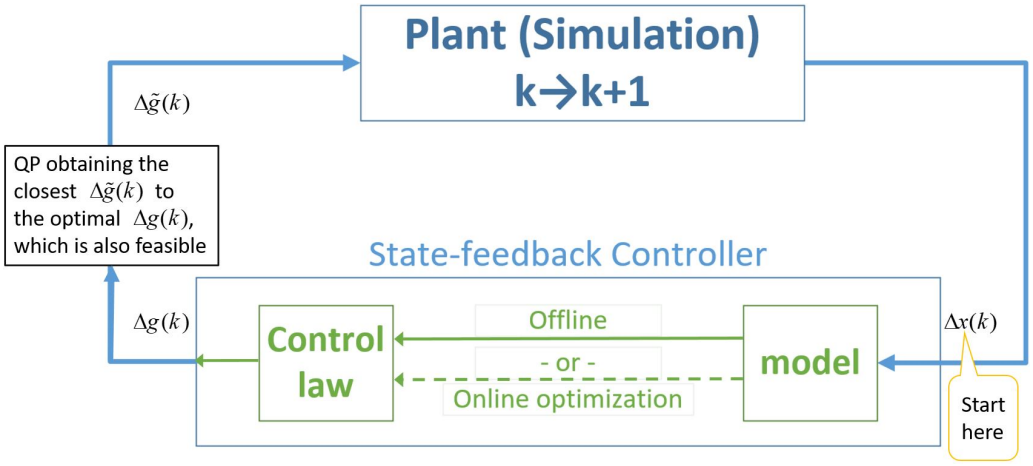


Figure A.2: State-feedback controllers implementation on a plant, in case where control law retrieved from the model is not feasible.

A.3 Feasibility and stability of MPC

Recursive feasibility (that guarantees feasibility of time step $k+1$ if at the previous step k, the solution is feasible) and stability (convergence to the origin) are not assured assuming (2.11) because of finite horizon. Even though if there is a feasible solution till now, it does not automatically mean that next step there will be feasible solution.

There are several methods to guarantee recursive feasibility and stability, e.g. adding final state constraint $\Delta\mathbf{x}(k+N)=0$. The method used in this thesis is the dual mode MPC [16]. A representative infinite horizon term $V(\Delta\mathbf{x}(k+N))$ is added, which is not constrained (except by dynamics) and requires that the final step will obey $\Delta\mathbf{x}(k+N)\in\Omega_{\max}$, and thus ensuring that the optimal control sequence will drive the state to the origin. $\Omega_{\max}$ in this case is the MAS of linear control derived from solving ARE (2.6) and assigning the solution P_{k+N} into (2.5a,b).

Then, the new objective function changes from (2.11) to:

$$\begin{aligned} J(\Delta\mathbf{x}(k)) &= \min_{\mathbf{g}_N}\frac{1}{2}\Bigg\{\sum_{\tau=k}^{k+N-1}\left(\Delta\mathbf{g}(\tau)^T R\Delta\mathbf{g}(\tau)+\Delta\mathbf{x}(\tau)^T Q\Delta\mathbf{x}(\tau)\right) \\ &+\underbrace{\sum_{\tau=k+N}^{\infty}\left(\Delta\mathbf{g}(\tau)^T R\Delta\mathbf{g}(\tau)+\Delta\mathbf{x}(\tau)^T Q\Delta\mathbf{x}(\tau)\right)}_{V(\Delta\mathbf{x}(k+N))}-\cancel{\Delta\mathbf{x}(k)^T Q\Delta\mathbf{x}(k)}\Bigg\} \\ &= \min_{\mathbf{g}_N}\frac{1}{2}\left\{\sum_{\tau=k}^{\infty}\left(\Delta\mathbf{g}(\tau)^T R\Delta\mathbf{g}(\tau)+\Delta\mathbf{x}(\tau)^T Q\Delta\mathbf{x}(\tau)\right)\right\} \end{aligned} \tag{A.5}$$

After solving ARE (2.7) for unconstrained $V(\Delta\mathbf{x}(k+N))$ and finding $P=P_{k+N}$ and $L=L_{k+N}$, from (2.5b), the V is changed due to some stabilizing controller (2.3):

$$V(\Delta\mathbf{x}(k+N))=\frac{1}{2}\sum_{\tau=k+N}^{\infty}\left(\Delta\mathbf{x}(\tau)^T(Q+L^T RL)\Delta\mathbf{x}(\tau)\right). \tag{A.6}$$

But due to (2.5d), V is also equal to:

$$V(\Delta\mathbf{x}(k+N))=\frac{1}{2}\Delta\mathbf{x}(k+N)^T P\Delta\mathbf{x}(k+N). \tag{A.7}$$

Hence, the final cost function is (2.11), only with a terminal term that ensures

recursive feasibility and stability:

$$J(\Delta\mathbf{x}(k)) = \min_{\mathbf{g}_N} \frac{1}{2} \sum_{\tau=k}^{k+N-1} \left(\Delta\mathbf{g}(\tau)^T R \Delta\mathbf{g}(\tau) + \Delta\mathbf{x}(\tau)^T Q \Delta\mathbf{x}(\tau)\right) + \frac{1}{2}\Delta\mathbf{x}(k+N)^T P \Delta\mathbf{x}(k+N) \tag{A.8}$$

and with a terminal constraint, expressed by changing the state constraints from (2.11) into:

$$\Delta\mathbf{x}(r+1) \in \mathcal{X} = \{\Delta\mathbf{x} \in \mathbb{R}^n : F_x \Delta\mathbf{x}(t+1) \leq h_x\}, \tag{A.9}$$

$$\Delta\mathbf{x}(k+N) \in \Omega_{\max} = \{\Delta\mathbf{x} \in \mathbb{R}^n : F_0 \Delta\mathbf{x}(k+N) \leq h_0\}, \tag{A.10}$$

$$\forall r = k, k+1, \ldots, k+N-2\,. \tag{A.11}$$

Therefore, using the previous QP formulation (2.15), we change only the $\mathbf{Q}_N, \mathbf{F}_{x,N}, \mathbf{g}_{x,N}$ matrices from (2.14) into:

$$\mathbf{Q}_N = \begin{bmatrix} Q & 0 & \cdots & 0 \\ 0 & \ddots & \cdots & 0 \\ \vdots & \vdots & Q & \vdots \\ 0 & 0 & \cdots & P \end{bmatrix}, \tag{A.12a}$$

$$\mathbf{F}_{x,N} = \begin{bmatrix} F_x & 0 & \cdots & 0 \\ 0 & F_x & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & F_0 \end{bmatrix}, \mathbf{g}_{x,N} = \begin{bmatrix} h_x \\ h_x \\ \vdots \\ h_0 \end{bmatrix}. \tag{A.12b}$$

Bibliography

[1] Alberto Bemporad and Manfred Morari. Robust model predictive control: A survey. In *Robustness in identification and control*, pages 207–226. Springer, 1999.

[2] Konstantinos Aboudolas, Markos Papageorgiou, and E Kosmatopoulos. Store-and-forward based methods for the signal control problem in large-scale congested urban road networks. *Transportation Research Part C: Emerging Technologies*, 17(2):163–174, 2009.

[3] Markos Papageorgiou, Christina Diakaki, Vaya Dinopoulou, Apostolos Kotsialos, and Yibing Wang. Review of road traffic control strategies. *Proceedings of the IEEE*, 91(12):2043–2067, 2003.

[4] PCM Ribeiro. Handling traffic fluctuation with fixed-time plans calculated by transyt. *Traffic engineering & control*, 35(6):362–366, 1994.

[5] Yafeng Yin. Robust optimal traffic signal timing. *Transportation Research Part B: Methodological*, 42(10):911–924, 2008.

[6] Satish V Ukkusuri, Gitakrishnan Ramadurai, and Gopal Patil. A robust transportation signal control problem accounting for traffic dynamics. *Computers & Operations Research*, 37(5):869–879, 2010.

[7] Tamás Tettamanti, Tamas Luspay, Balazs Kulcsar, Tamas Peni, and István Varga. Robust control for urban road traffic networks. *IEEE Transactions on Intelligent Transportation Systems*, 15(1):385–398, 2014.

[8] Lihui Zhang, Yafeng Yin, and Yingyan Lou. Robust signal timing for arterials under day-to-day demand variations. *Transportation Research Record: Journal of the Transportation Research Board*, (2192):156–166, 2010.

[9] Daniel Tabak. Applications of mathematical programming techniques in optimal control: a survey. *IEEE Transactions on Automatic Control*, 15(6):688–690, 1970.

[10] Stephen Frank, Ingrida Steponavice, and Steffen Rebennack. Optimal power flow: a bibliographic survey i. *Energy Systems*, 3(3):221–258, 2012.

[11] B Francis, OA Sebakhy, and WM Wonham. Synthesis of multivariable regulators: The internal model principle. *Applied Mathematics and Optimization*, 1(1):64–86, 1974.

[12] H Kwatny and K Kalnitsky. On alternative methodologies for the design of robust linear multivariable regulators. *IEEE transactions on Automatic Control*, 23(5):930–933, 1978.

[13] Jan Marian Maciejowski. *Predictive control: with constraints*. Pearson education, 2002.

[14] K Aboudolas, M Papageorgiou, A Kouvelas, and E Kosmatopoulos. A rolling-horizon quadratic-programming approach to the signal control problem in large-scale congested urban road networks. *Transportation Research Part C: Emerging Technologies*, 18(5):680–694, 2010.

[15] Karl J. Astrom and B. Wittenmark. *Adaptive control*. Courier Corporation, 2013.

[16] Hoai-Nam Nguyen. Constrained control of uncertain, time-varying, discrete-time systems. *An Interpolation-Based Approach (Cham: Springer)*, 2014.

[17] Hoai-Nam Nguyen, Per-Olof Gutman, Sorin Olaru, and Morten Hovd. Implicit improved vertex control for uncertain, time-varying linear discrete-time systems with state and control constraints. *Automatica*, 49(9):2754–2759, 2013.

[18] H-N Nguyen, P-O Gutman, and Romain Bourdais. More efficient interpolating control. In *2014 European Control Conference (ECC)*, pages 2158–2163. IEEE, 2014.

[19] Jack Haddad. Robust constrained control of uncertain macroscopic fundamental diagram networks. *Transportation Research Part C: Emerging Technologies*, 59:323–339, 2015.

[20] Alon Tuchner and Jack Haddad. Vehicle platoon formation using interpolating control: A laboratory experimental analysis. *Transportation Research Part C: Emerging Technologies*, 84:21–47, 2017.

[21] Christina Diakaki, Markos Papageorgiou, and Kostas Aboudolas. A multivariable regulator approach to traffic-responsive network-wide signal control. *Control Engineering Practice*, 10(2):183–195, 2002.

[22] D Subbaram Naidu. *Optimal control systems*. CRC press, 2002.

[23] Tung Le, Hai L Vu, Yoni Nazarathy, Bao Vo, and Serge Hoogendoorn. Linear-quadratic model predictive control for urban traffic networks. *Procedia-Social and Behavioral Sciences*, 80:512–530, 2013.

[24] David Q Mayne, James B Rawlings, Christopher V Rao, and Pierre OM Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[25] Christina Diakaki. Integrated control of traffic flow in corridor networks. *Ph. D. Thesis, Department of Production Engineering and Management, Technical University of Crete*, 1999.

[26] John Bay. Fundamentals of linear state space systems. 1999.

[27] Robert L Williams, Douglas A Lawrence, et al. *Linear state-space control systems*. Wiley Online Library, 2007.

[28] Alberto Bemporad, Manfred Morari, Vivek Dua, and Efstratios N Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.

[29] Jack Haddad, Bart De Schutter, David Mahalel, Ilya Ioslovich, and Per-Olof Gutman. Optimal steady-state control for isolated traffic intersections. *IEEE Transactions on automatic control*, 55(11):2612–2617, 2010.

[30] Jack Haddad, Per-Olof Gutman, Ilya Ioslovich, and David Mahalel. Discrete dynamic optimization of n-stages control for isolated signalized intersections. *Control Engineering Practice*, 21(11):1553–1563, 2013.

[31] Pedro Mercader, Daniel Rubin, Hoai-Nam Nguyen, Alberto Bemporad, and Per-Olof Gutman. Simple interpolating control. In *9th IFAC Symposium on Robust Control Design (ROCOND'18) and 2nd IFAC Workshop on Linear Parameter Varying Systems (LPVS'18)*, Florianopolis, Brazil, September 3-5 2018.

[32] HN Nguyen, P-O Gutman, Sorin Olaru, Morten Hovd, and F Colledani. Improved vertex control for time-varying and uncertain linear discrete-time systems with control and state constraints. In *American Control Conference (ACC), 2011*, pages 4386–4391. IEEE, 2011.

[33] Hoai-Nam Nguyen and Per-Olof Gutman. Model predictive control for constrained uncertain, time-varying systems. *IFAC-PapersOnLine*, 48(11):930–935, 2015.

[34] P-O Gutman and Michael Cwikel. Admissible sets and feedback control for discrete-time linear dynamical systems with bounded controls and states. *IEEE transactions on Automatic Control*, 31(4):373–376, 1986.

[35] H-N Nguyen, P-O Gutman, Sorin Olaru, and Morten Hovd. An interpolation approach for robust constrained output feedback. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pages 5516–5521. IEEE, 2011.

[36] Tingshu Hu and Zongli Lin. Composite quadratic lyapunov functions for constrained control systems. *IEEE Transactions on Automatic Control*, 48(3):440–450, 2003.

[37] Jack Haddad. Robust constrained control of uncertain macroscopic fundamental diagram networks. *Transportation Research Part C: Emerging Technologies*, 59:323–339, 2015.

[38] Mayuresh V Kothare, Venkataramanan Balakrishnan, and Manfred Morari. Robust constrained model predictive control using linear matrix inequalities. *Automatica*, 32(10):1361–1379, 1996.

[39] Wilbur Langson, Ioannis Chryssochoos, SV Raković, and David Q Mayne. Robust model predictive control using tubes. *Automatica*, 40(1):125–133, 2004.

תקציר

בעיית העומס בדרכים היא בעיה אקטואלית ואחת הבעיות העיקריות בבקרת תנועה כיום. לטיפול בבעיה זו, קיימות אסטרטגיות רבות של בקרת תנועה (רמזורים) בספרות מבוססות על מודלים שונים של זרימת תנועה. אם כי, הן אינן כוללות אי-ודאויות של פרמטרים בדינמיקת התורים בצומת.

לפיכך, אסטרטגיות בקרה אלה אינן מסוגלות להתמודד עם תנאים בהם ערכי הפרמטרים המשוערים משתנים באופן משמעותי מהערכים הריאליים שלהם. יתר על כן, הרבה גישות של בקרת משוב מצב המשומשים לפיתוח בקרי רמזורים חסרים את הטיפול באילוצי מצב ובקרה ישירות בשלב התכנון.

תזה זו נותנת מענה לבעיה המתוארת לעיל בעזרת בקרת רמזורים איתנה עבור רשתות עירוניות עם אי-וודאויות.

המטרה הראשונה בתזה, היא לשלב אי-ודאויות של הפרמטרים בתוך מודל ה(SF) store-and-forward אשר משתמשים בו בתזה הזו כדי לתאר את דינמיקת התורים לצמתים מרומזרים.

אי-הודאות במודל תוגדר בטווח מסויים בין ערכי מינימום וערכי מקסימום, ותתייחס לפרמטרים הבאים: זרמי רוויה וביקוש.

מודל הSF הוא מודל מצב לינארי, אשר מפשט את התיאור המתמטי של תהליך זרימת התנועה בצמתים מרומזרים, בעזרת המרת אופן דיסקרטי של מאורעות (זרימה ברמזור ירוק ואי-זרימה ברמזור אדום) לאופן רציף, ע"י מיצוע הזרימה לאורך זמן המחזור בצומת. במילים אחרות, מודל הSF מתעלם מהדינמיקה הפנימית במהלך המחזור.

מודל הSF מורכב ממשתני מצב: מספר הרכבים בכל קטע דרך, ומשתני בקרה: זמני פאזת ירוק במחזור. כמו כן, מודל הSF כולל אילוצי מצב: בכל קטע דרך יש בין אפס רכבים לבין הקיבולת של הקטע, ואילוצי בקרה: ישנם מינימום ומקסימום לזמנים

הירוקים ואילוץ המחייב שסכום הזמנים הירוקים בכל מחזור ביחד עם זמני הבין-ירוקים יהיה קטן או שווה לזמן המחזור בכל צומת.

הפורמולציה הסופית של המודל מפותחת מתוך משוואת שימור של כלל הרכבים בכל קטע דרך ברשת, עם שילוב של יחסים בין הפרמטרים השונים בבעיה. הפרמטרים הם: זרמי כניסה ויציאה של קטעי דרך אחרים מ- ואל- קטע דרך נתון, וזרמי כניסה ויציאה בתוך קטע הדרך הנתון, כגון מ- ואל- חנייה או עקב החלפת נתיבים.

המטרה השנייה בתזה, היא עיצוב בקרת משוב איתנה המשלבת אילוצי מצב ובקרה, בעזרת המודל SF הכולל אי-וודאויות שפותח בשלב הראשון של התזה. בקרת המשוב הזו ממומשת על ידי גישה מבוססת אינטרפולציה. בתזה זו נדגים שימוש בשתי גרסאות של הגישה הזאת: IC וSIC.

גישה זו (i) מבטיחה עמידות/חסינות כנגד כל אי-הוודאויות של הפרמטרים, (ii) מטפלת באילוצי מצב ובקרה, וכן (iii) מציגה פתרון זול מבחינה חישובית.

לבסוף, תוצאות נומריות עבור צומת מרומזר בודד מציגות השוואה בין בקרי אינטרפולציה שפותחו (IC, SIC) לבין בקרים אחרים בספרות: MPC (model predictive control) וLQR- (linear quadratic regulator).

MPC וLQR הן שיטות בעלי פונקציית מטרה ריבועית זהה, המנסה להקטין את ריבועי משתני המצב (עבור רגולציה לערך נומינלי) ולהקטין את ריבועי משתני הבקרה (להקטנת המאמץ של הבקר). שתיהן משתמשות במודל לינארי דיסקרטי. אך הן נבדלות אחת מהשניה בכך שLQR יש חוק בקרת משוב לינארי וקבוע והוא לא מתחשב באילוצי הבעיה, ואילו MPC כן מתחשב באילוצים ומיושם על ידי שיטת אופק מתגלגל, שבו יש חיזוי לכמות מסויימת של צעדים קדימה בזמן ובכל פעם יישום אות הבקרה הראשון ברצף צעדים זה.

שיטות האינטרפולציה, לעומת השיטות מהספרות, משתמשות במודל לינארי דיסקרטי עם אי-וודאויות, ומתחשבת באילוצים. רעיון הבקרה בשיטות מסוג זה הוא

הפעלת קומבינציה לינארית על שני בקרים: בקר פנימי האחראי על ביצועים טובים ככל הניתן/אופטימליים, ובקר חיצוני האחראי להגדיל את מרחב הפתרונות האפשריים.

ההבדל העקרוני בין IC לSIC, הוא שבIC האילוצים הם סט של אילוצים לינאריים עם מרחב מצב פוליהדרי, ואילו בSIC האילוצים חייבים להיות סימטריים עקב הדרישה על מרחב המצב להיות אליפסואידי. כמו כן, SIC משתמש בבקרי רוויה לינאריים המייצגים את הבקרים הפנימי והחיצוני.

אם נשווה בין הבקרים השונים, נראה כי MPC הוא הבקר הכי אופטימלי בין כולם. LQR הוא אופטימלי כל עוד איננו פוגע באילוצי הבעיה, וכשהוא כן זה עלול לפגוע בביצועיו. IC פחות אופטימלי מMPC אך פחות כבד חישובית. SIC כולל אילוצים רק סימטריים ואלו שלא או שמצרים את האילוצים או שמיישמים אותם לאחר חוק הבקרה, מה שבכל מקרה עלול לפגוע בביצועים. היתרון בSIC הוא שהוא קל חישובית.

כל הבקרים המתוארים לעיל מתוכננים עם המטרה להביא את מצב דגם המערכת לראשית. הם כולם בקרי משוב, כאשר רק LQR הוא לינארי ואילו האחרים הם לינאריים למקוטעין או לא-לינאריים.

התוצאות כוללות מקרים שונים של צמתים בגדלים שונים וקונפיגורציות שונות, שחלקם ממומשים בעזרת בקרים נומינליים וחלקם ממומשים בעזרת בקרים רובסטיים.

בסופו של דבר, התוצאות מדגימות את יתרונות הביצועים עקב יישום של בקרת אינטרפולציה איתנה.

המחקר נעשה בהנחיית פרופסור חבר ג'אק חדאד בפקולטת הנדסה אזרחית וסביבתית.

אני מודה לטכניון ול- Andrea & Alfredo Pollitzer Memorial Fellowship by Mr. and Mrs. Lepri על התמיכה הכספית הנדיבה בהשתלמותי.

תודה מיוחדת על התמיכה והעזרה של פרופסור ג'אק חדאד, של צוות מעבדת TSMART, של משפחתי ושל הוריי שהביאו אותי עד הלום

בקרת רמזורים איתנה עבור רשת דרכים עם אי ודאות

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר מגיסטר למדעים במדעי התחבורה

שמעון קומרובסקי

הוגש לסנט הטכניון - מכון טכנולוגי לישראל

תשרי ה'תש"פ חיפה אוקטובר 2019

בקרת רמזורים איתנה עבור רשת דרכים עם אי ודאות

שמעון קומרובסקי