RESTRICTED STRUCTURE NON-LINEAR GENERALIZED MINIMUM VARIANCE CONTROL

PhD Thesis

Cagatay Cebeci

Wind Energy and Control Centre

Department of Electronic and Electrical Engineering

University of Strathclyde, Glasgow

January, 2022

Declaration

Abstract

This research presents the Restricted Structure Non-linear Generalized Minimum Variance (RS-NGMV) algorithm for Linear Parameter-Varying (LPV) systems. The LPV systems are defined as linear plant subsystems within the control diagram and may include Non-linear (NL) input subsystems. The RS-NGMV control solution for the latter will be slightly different than the first one and have the capability of dealing with NL characteristics such as saturation, discontinuities and black-box terms. The controller is built in a low-order Restricted Structure (RS) in the form of a general z-transfer function. This brings forward two major advantages. First, it offers a high-order advanced control solution inside low-order control structures which are known for their natural robustness. Secondly, it is easier to operate and re-tune for the classically trained staff in the industry as it can be given the structures they are rather familiar with such as the PID. Another advantage of the RS-NGMV is its model-based design that enables a faster adaptation to implement different systems.

Features of the RS-NGMV are investigated throughout the thesis with case studies from trends in engineering like robotics, autonomous and electric vehicles. The results show that the RS-NGMV is highly capable of adapting to set-point changes, parameter variations with its ability to update the control gains rapidly by using optimizations. Some extensions of algorithms have also been studied following recent directions in optimal/predictive control resulting in a new preview control approach and Scheduled RS-NGMV control.

To the memory of my grandmother Sevim. Your compassion helped me find my way and I shall remember you, my dearest. My love for you is everlasting.

Acknowledgements

First and foremost, I would like to thank my supervisor Prof. Mike Grimble for sharing his research experience and wisdom to guide me throughout my studies. It has been a true pleasure to work with such an inspiring control scientist and a gentleman. I would like to express my great gratitude to Dr. Luis Recalde-Camacho who mentored me on my research and co-authored my papers. I am very thankful for his contribution and patience. I would like to thank Dr. Pawel Majecki and Dr. Luca Cavanini of the ISC Ltd. as well for sharing their software expertise and discussions on my work. I would also like to acknowledge our amazing admin Drew Smith at the Wind Energy & Control CDT Centre, for his hard work, friendship and being the person who makes magic possible when it is needed. Many thanks to my colleagues and friends at the Centre and the Technology & Innovation building for the support and encouragement with my research including: Giorgio Zorzi for teaching me how to use the Inkscape tool and assistance with LaTex which has helped me many times with my papers and thesis; Dr. Tim Rubert, Dr. Alexander Giles and Dr. Rafael Dawid for insightful comments about my progress; Dr. Jethro Browell, and Dr. Bruce Stephen for the discussions on the R programming language and statistical analysis; Dr. Sultan M. Alotaibi for our discussions and collaboration on the PhD process and RDP Programme courses; Dr. Rafael Peña Alzola for giving me scientific writing tips and persistently reminding me that writing regularly is a very important pillar of being a researcher. I am thankful to my reviewing academics Dr. Hong Yue and

Prof. Olimpo Anaya-Lara for their feedback and encouragement on my work, helping me notice some challenges but move towards solving them. I would like to extend my acknowledgement of Dr. Hong Yue with my thanks for being included as a teaching assistant in the Control Principles II class. Many thanks to Dr. Adam Stock and Dr. David Campos as well, for assigning me as a teaching assistant for the Control Principles I class. I would also like to acknowledge Sheila Campbell, Prof. Bill Leithead and Prof. Campbell Booth for their guidance and support with the PhD Programme.

I am grateful to Turkish Ministry of National Education for providing me with scholarship to carry out my postgraduate studies. Many thanks for giving me the opportunity to study abroad and learn from different cultures. It was a privilege to be a member of the University of Virginia and then the University of Strathclyde, having a chance to work with some remarkable scientists, and of course to see the evergreen Virginia, the Scottish Highlands and Glasgow city of arts, culture, people and music.

My warmest thanks to my love Megan MacDougall and her beautiful family for their amazing embrace and delightful moments that we shared. I am lucky to have met you and I hope you will always stand by my side.

Special thanks to my friends back in Turkey, the University of Strathclyde and Glasgow for their support, good times and laughs. I have never thought I would meet so many nice people and be remembered this much.

I would like to thank my parents Nalan & Zeynel and the rest of my family for whom words are not enough to express my love and gratitude. I would like to cherish the memory of our hamster Nutmeg for his encouragement during the most difficult times of my work and being my tiny super hero. Many thanks to Nutmeg's little sister Sunshine for being the joy of our home.

List of Figures

List of Tables

List of Symbols

Term	**Description**
A	Matrix: State
B	Matrix: Input
C	Matrix: Output
D	Matrix: State disturbance
A_t	LTV/LPV/SD Matrix: State
B_t	LTV/LPV/SD Matrix: Input
C_t	LTV/LPV/SD Matrix: Output
D_t	LTV/LPV/SD Matrix: State disturbance
k	Time delay instant
r	Subscript: Reference subsystem state-space matrix index
u	Subscript: Input subsystem state-space matrix index
d	Subscript: Disturbance subsystem state-space matrix or deterministic disturbance signal index
0	Subscript: Linear or LPV/SD subsystem state-space matrix index
t	As time instant or subscript: Time-varying subsystem state-space matrix index

Term	**Description**
w	Subscript: Deterministic portion for signal index
m	Subscript: Measurements signal $z(t)$ state-space matrix index
p	Subscript: Error weighting P_c subsystem state-space matrix index
i	Subscript: Iteration index
j	Subscript: Iteration index
W_0	Linear plant subsystem
W_{1k}	Unstructured Non-linear input subsystem
P_c	Weighting: NGMV/RS-NGMV cost function error term
F_{ck}	Weighting: NGMV/RS-NGMV cost function control term
$\phi(t)$	Variance term for the NGMV/RS-NGMV cost functions
$r(t)$	Signal: Reference
$e(t)$	Signal: Tracking error
$y(t)$	Signal: Output
$u(t)$	Signal: Input
$p(t)$	Signal: Scheduling parameters
$x(t)$	Signal: States
$\hat{x}(t)$	Signal: State estimates
$d(t)$	Signal: Disturbance
$v(t)$	Signal: Measurement noise
$z(t)$	Signal: Observations
x	Signal: State notation $x(t)$ simplified.
$\rho(t)$	Signal: Varying-parameter

Term	Description
ρ	Signal: Parameter $\rho(t)$ simplified.
$\xi(t)$	Signal: Zero-mean white noise.
$k_c(t)$	Gain: Total RS-NGMV gain
$\overline{k}_c$	Gain: Fixed component of RS-NGMV gain
$\widetilde{k}_c(t)$	Gain: Deviation component of RS-NGMV gain
K	Gain: Total gain vector for LQ and preview controllers
K_{fb}	Gain: Feedback gain for LQ and preview controllers
K_{ff}	Gain: Feedforward gain for LQ and preview controllers
N_p	Control Horizon: Preview
N	Control Horizon: LQ or Predictive
Q	Control Weighting: LQ type
R	Control Weighting: LQ type
$\mathcal{A}$	Matrix: State for the augmented LQ/Preview system
A_r	Matrix: Road state
$\overline{A}_r$	Matrix: Road state for the augmented LQ/Preview system
B_r	Matrix: Road state input
$\overline{B}_r$	Matrix: Road state input for the augmented LQ/Preview system
C_r	Matrix: Road state output
$\mathcal{B}$	Matrix: Input for the augmented LQ/Preview system
$\mathcal{B}_r$	Matrix: Road input for the augmented LQ/Preview system
$\mathcal{C}$	Matrix: Output for the augmented LQ/Preview system
$\mathcal{X}$	Signal: State for the augmented LQ/Preview system

Term	**Description**
Y	Signal: Output for the augmented LQ/Preview system
x_r	Signal: Road state
y_r	Signal: Road state output
Y_r	Signal: Road state output for the augmented LQ/Preview system
w_r	Signal: Road state stochastic white noise
$\overline{w}_r$	Signal: Road state stochastic white noise for the augmented LQ/Preview system

Acronyms

Term	**Description**
ADAS	Advanced Driver Assistance Systems
ANN	Artificial Neural Network
ARE	Algebraic Riccati Equation
BMS	Battery Management Systems
CARMA	Contemporaneous Auto-Regressive Moving-Average
CAD	Computer Aided Design
DOF	Degree-of-Freedom
DPPC	Dynamic Performance Preview-Predictive Control
ECM	Equivalent Circuit Model
EKF	Extended Kalman Filter
ENGMV	Extended Non-linear Generalized Minimum Variance
EV	Electric Vehicle

Term	**Description**
FWD	Forward Wheel Drive
GMV	Generalized Minimum Variance
GPC	Generalized Predictive Control
HWFET	Highway Fuel Economy Test
KF	Kalman Filter
LFT	Linear-Fractional Transformation
LMI	Linear Matrix Inequality
LPV	Linear Parameter-Varying
LPV-RE	Linear Parameter-Varying Riccati Equation
LPVKF	Linear Parameter-Varying Kalman Filter
LQ	Linear Quadratic
LQG	Linear Quadratic Gaussian
LQR	Linear Quadratic Regulator
LTI	Linear Time-Invariant
LTV	Linear Time-Varying
MIMO	Multi Input Multi Output
MPC	Model Predictive Control
MV	Minimum Variance
NGMV	Non-linear Generalized Minimum Variance

Term	**Description**
NL	Non-linear
NPGMV	Non-linear Predictive Generalized Minimum Variance
NQGMV	Non-linear Quadratic Generalized Minimum Variance
OCV	Open-Circuit Voltage
PPID	Predictive PID
qLPV	Quasi Linear Parameter-Varying
RE	Riccati Equation
RLS	Recursive Least Squares
RS	Restricted Structure
RS-GPC	Restricted Structure Generalized Predictive Control
RS-LQG	Restricted Structure Linear Quadratic Gaussian
RS-NGMV	Restricted Structure Non-linear Generalized Minimum Variance
RS-PID	Restricted Structure PID
SAE	Society of Automotive Engineers
SD	State-Dependent
SD-NGMV	State-Dependent NGMV
SDC	State-Dependent Co-efficient
SDRE	State-Dependent Riccati Equation
SI	Spark Ignition

Term	**Description**
SISO	Single Input Single Output
SOC	State-of-Charge
SS	State-Space
UDDS	Urban Dynamometer Driving Schedule

Contents

Chapter 1

Introduction

The RS-NGMV control can trace its roots back to several decades ago. In 1969, Åström [1] introduced the Minimum Variance (MV) control method, followed by some process control applications. The MV strategy is to minimize a cost function that is the expected value of the output variance of a stochastic system, the solution of which is derived by using k-steps ahead predictions and then cancelling out the stochastic terms that cannot be influenced by the control signal. The method showed success in paper machine applications where it was used to minimize the variance of the moisture content of the paper, improving its quality.

The MV controller design was based on the assumption that the plant is of the minimum-phase. As a result, the output can grow unbounded when dealing with non-minimum phase systems. The Generalized Minimum Variance (GMV) control [2] was proposed as a solution to this problem by simply extending the MV control cost function with the introduction of a weighted control variance term. GMV controllers have shown success in process control applications as well. In addition, they have been used in a number of power, automotive and robotics implementations [3−5] in later studies. Some versions of GMV controllers are able to deal with unknown system parameters by using methods like Recursive Least Squares (RLS) estimations [6]. They are recognized as self-tuning controllers [7, 8] in the literature. The original GMV method was also studied by Grimble [9], by introducing a modified cost function of weighted error and control terms whose solution returns the control law.

The techniques mentioned so far considered solutions only for linear systems. The Non-linear Generalized Minimum Variance (NGMV) control algorithm was introduced for discrete-time non-linear, multi-variable, possibly time-varying systems [11, 12]. The NGMV uses a closed-loop feedback control system structure very similar to that of the GMV in [9]. However, the plant model is allowed to be non-linear and if linear, the controller has the ability to revert to the GMV. The systems and the control law in the initial version of the NGMV were represented by polynomial expressions like the standard or early MV and GMV controllers. The NGMV control structure can also be related to the Smith predictor [10], a well-known process control strategy. However, it does not share the same problem of stability for open-loop unstable systems. Moreover, it does not have the robustness problem since in its natural form it does not try to cancel out the plant dynamics.

Later, the State-Space (SS) version with continuous-time design was proposed [13] (for discrete-time adaptations see [16]) followed by the State-Dependent (SD) version State-Dependent NGMV (SD-NGMV), shortly after [14]. In [15], hybrid system implementations were considered. These state-space based designs have enabled the utilization of the Kalman Filter (KF) techniques for the NGMV, providing the benefit of estimating states that cannot be measured and exploiting the system information even more efficiently. The Linear Parameter-Varying Kalman Filter (LPVKF) has given possibility to consider solutions for the LPV systems as well.

As a branch of state-space techniques, LPV and SD models are increasingly used to represent or approximate non-linear systems, including the control problems in this thesis. Traditionally, the State-Dependent Riccati Equation (SDRE) approach [17, 18] has been used for the optimal control of such non-linear systems owing to its well-defined formulation and good stability characteristics. Basically, its principle is to encapsulate a non-linear system in a linear structure and then employ the well-known Linear Quadratic (LQ) optimal control solution. The technique has inspired the controller [19] that will be explored in a later chapter. In fact, the LQ type controllers are not too far-off compared to the MV type (refers to all MV, GMV and NGMV) controllers. They have similar cost-functions leading to different solutions. There is already an approach within the NGMV family of controllers called Non-linear Quadratic Generalized Minimum Variance (NQGMV) related to the Linear Quadratic Gaussian (LQG) controller [16]. For instance, the NQGMV can reduce to LQG in its limiting case. It could be summarized that while the LQ methods have the advantage of being able to offer guaranteed stability, the MV methods remain simpler and are easier to implement. Regarding the scope of this thesis, we are not concerned with further theoretical comparison between the two methods.

The stability of the NGMV methods has been briefly discussed in some earlier studies by explaining the conditions ensuring it. First of all, the inverse of the non-linear operator term $(P_C W_k - F_{ck})$, a combination of cost-function weights and delay free plant model, must exist and it must be stable. It has been shown that the stability of the inverse non-linear operator term is directly related to the stability of the NGMV feedback loop. With the correct choice of weights, it is possible to make sure these requirements are met. However, the type of the NGMV controller also needs to be considered carefully. According to the NGMV control solution and the plant models used, the stability conditions and the assumptions may slightly differ. For example, the SD-NGMV designs in [14, 15] were capable of stabilizing open-loop unstable processes with both input and output non-linearities unlike the previous NGMV designs which were based on the assumption that the non-linear subsystem had to the stable.

An NGMV controller is predictive in the sense that it uses k-step ahead predictions, however the method does not use the full future tracking error and control signal information like the Model Predictive Control (MPC) methods. Thus we cannot define it as a purely predictive control method. The problem has created the motivation for extensions such as the Non-linear Predictive Generalized Minimum Variance (NPGMV) controller $[20-23]$ derived by modifying the standard Generalized Predictive Control (GPC) cost function for the NGMV form to gain some of the advantages of the predictive control. The NPGMV theory was first based on general polynomial and state-space formulations. Later, LPV adaptations were formulated and verified with non-linear industrial control applications such as robotics, ship steering and wind turbine control in $[24-26]$.

Another attempt to enhance the predictive capabilities of the NGMV was to consider the use of future set-point information. This was first realized with the Extended Non-linear Generalized Minimum Variance (ENGMV) control method [27]. The future reference information was incorporated using a Two-Degree-of-Freedom (DOF) structure that improved the predictive capabilities but came at the expense of some complexity. The ENGMV method also used the future reference information for k-steps only.

The use of future set-point information, can be referred to as the "preview" or "look-ahead" in the control systems literature. The concept is that the future reference information could be incorporated in the system model and if the control solution is derived properly by utilizing the information, then the controller has the preview feature or is called a preview controller. To clarify the last bit, the type of optimal control that is not necessarily predictive but uses the future reference information is called as the preview control in the literature. On the other hand, if a predictive method like the MPC has the preview feature it may or may not be defined MPC with preview based on the authors' choice. The standard preview controller, proposed by Tomizuka [28], uses the LQ framework and the cost function to obtain the control solution. Thus, the technique is sometimes defined as the LQ-Preview control. The State-Dependent Riccati Equation and Linear Parameter-Varying Riccati Equation (LPV-RE) Preview controllers of Chapter 4 are related to this method.

These optimal and/or predictive control methods are classified as advanced control methods. They are usually of high-order and come with a higher level of computational burden due to the optimizations performed. On the other hand, the control methods like the classical PID are of low-order thus simpler to design and tune, with little computational complexity. A demonstration of the industrial control hierarchy is shown in Fig. 1.1 in four layers: Operational management, local optimizations and advanced control, PID controllers and the process.

To this day, there have been countless PID designs and applications. In the process control industry, the overwhelming majority of the controllers are still PID [16]. They are dominantly used for the flow, pressure, temperature and level controls. However, the latest breakthroughs in technology and the computing power available today extend the range of advanced control applications. For example, in the 1980s the MPC controllers were only used in process control industry where dynamics tend to be slow but recently they are becoming very popular for even high-speed applications like robotics and automotive. Unfortunately, these advancements do not simply outweigh some disadvantages of the advanced controllers. The fact itself is a call for research, that is, if an advanced controller can offer its superior performance while being presented in a low-order structure that is easier to maintain and tune like the classical PID. The so-called RS approach including the RS-NGMV control of this thesis, is an effort to answer the question. The core concept of the approach lies within the use of a pre-defined controller structure (like PID) whose gains are determined with optimizations.

These milestones summarize the time-line of the evolution of the NGMV method until the start of this thesis which aims to carry the torch forward. The

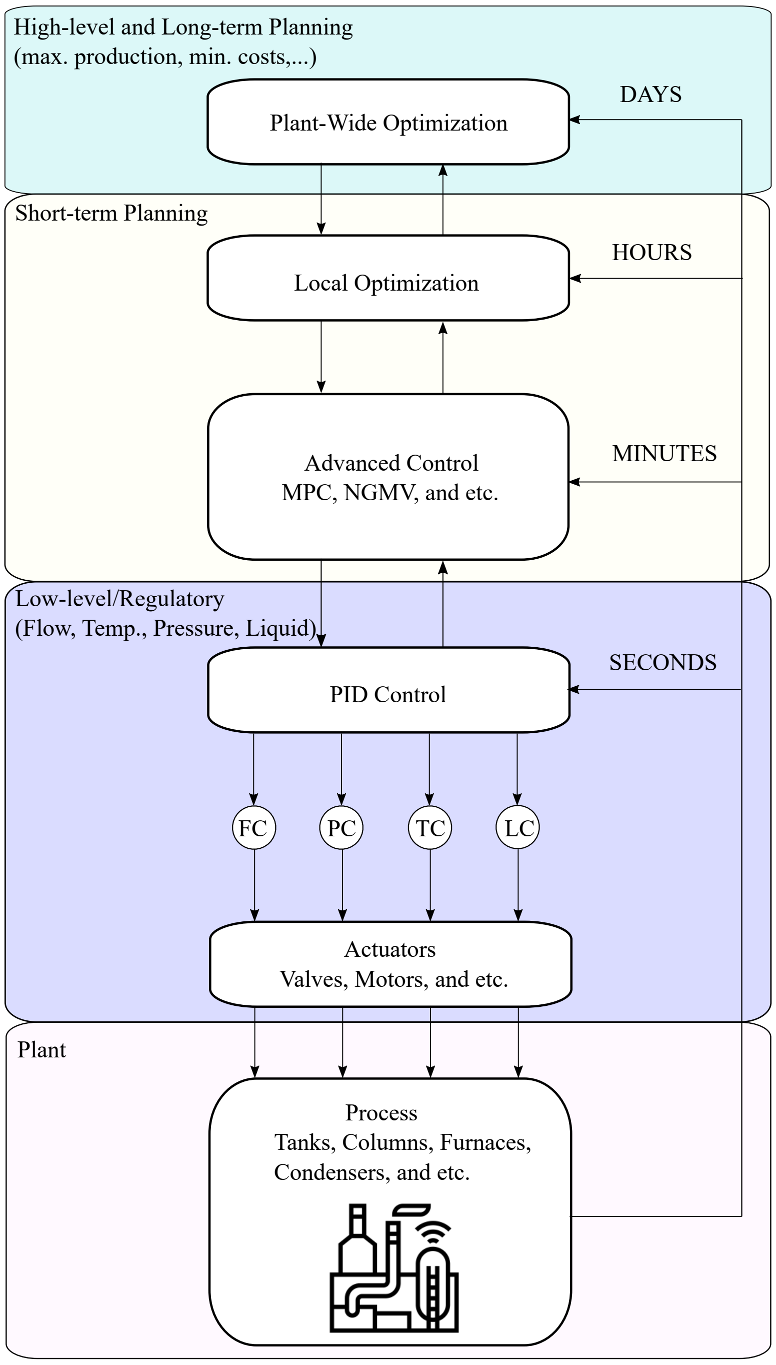


Figure 1.1: Industrial control hierarchy.

RS-NGMV control is introduced for LPV systems (including quasi-LPV) and its properties along with possible enhancements have been investigated. The results of the latter are presented as the SDRE/LPV-RE Preview controllers, the Scheduled RS-NGMV method and the proposal for RS-NPGMV. The designs have been verified with modern engineering applications in tune with today's trends including a Robotic Manipulator, Autonomous Car and Electric Vehicle (EV).

1.1 Research Objectives and Motivations

Although simpler to implement compared to most other advanced control algorithms, the Non-linear Generalized Minimum Variance, is also of high-order and come with a level of complexity. When these methods are used in the industry, despite they offer superior results, it might be an overwhelming task to operate them for the plant engineers. In most cases, the staff are more familiar with classical methods such as the PID and their tuning to adjust the controllers depending on their needs. This is where the idea of the Restricted Structure NGMV flourishes, to provide the efficiency of an advanced control algorithm but within a low-order restricted structure instead of the full-order optimal control solution. The restricted structure is of the general transfer function form. For instance, it can be chosen as the PID controller which will allow much easier tuning of the optimal controller for the classically trained staff eliminating the disadvantage of the complexity to operate. Motivated by these ideas, the objectives of this research are summarized as:

- Develop scalar and multi-variable Restricted Structure NGMV control solutions for state-space represented LPV systems and the LPV systems that contain non-linear black-box input subsystem by using real-time optimizations based on Kalman filter estimations,
- Investigate the introduction of the algorithm with additional features,

- Verify the methods by software implementation where different categories of Linear Parameter-Varying systems are considered, paired with suitable control applications. Explore the features and advantages of the Restricted-Structure NGMV algorithm under variable circumstances in these case studies and discuss the simulation results.

1.2 Thesis Contributions

The contributions of the research are composed of the outputs listed below:

- Scalar and multi-variable Restricted Structure NGMV controllers for LPV systems and LPV systems with non-linear black-box input subsystem. The control solutions are introduced in chapter 3.

- Use of Linear Parameter-Varying Kalman filter for exploiting the information available to the system continuously thus enabling adaptability of the RS-NGMV controller. The Kalman filter information is used for calculating the RS-NGMV solutions introduced in chapter 3.

- The use of State-Dependent Riccati Equation and Linear Parameter-Varying Riccati Equation approaches for the first time in the preview control of LPV or State-Dependent systems. The approaches are introduced in chapter 4. The second time in the literature, a LPV system being considered for the preview control problem shown in chapter 6.

- Proposal to a simplification to the RS-NGMV algorithm by using a fixed gains approach considering processes with limited processing capabilities (e.g. the microcontrollers with limited computational power). This approach is called the Scheduled RS-NGMV control and is introduced in chapter 8. It stores selected gains from the previous simulation runs and use them for the designated conditions instead of performing real-time optimization all the time.

- Software verification of the controllers were made using simulations in the Matlab/Simulink environment. To realize this, four different application examples three of which being based on Linear Parameter-Varying models were studied. The results have been analysed to understand the properties of the control algorithms. Chapters $5-8$ presents these examples.
- Ideas for future work have emerged:
  - The Restricted Structure Non-linear Predictive GMV (RS-NPGMV) controller with the preview feature is the first of these. The controller uses an additional degree-of-freedom to utilize the future information of the reference inputs. The concept and some preliminary work are discussed in the Appendix A.
  - Secondly, automating the Scheduled RS-NGMV gains in a smart way to give the best results in different operating conditions. Machine learning methods like classification or clustering to pick up the best conditions will also be considered for this technique. The steps are being taken for these objectives and progress will be made.

1.3 Thesis Organization

The thesis is organized as below. For an illustration of the conceptual thesis flow, refer to the Fig. 1.2 where the coloured areas that are titled methods and applications represent the contributions of the thesis.

- **Chapter 1:** Sets the scene for the research area and describes the place where the topic of the thesis stands. It summarizes the research objectives, motivations and the contributions that have been made.

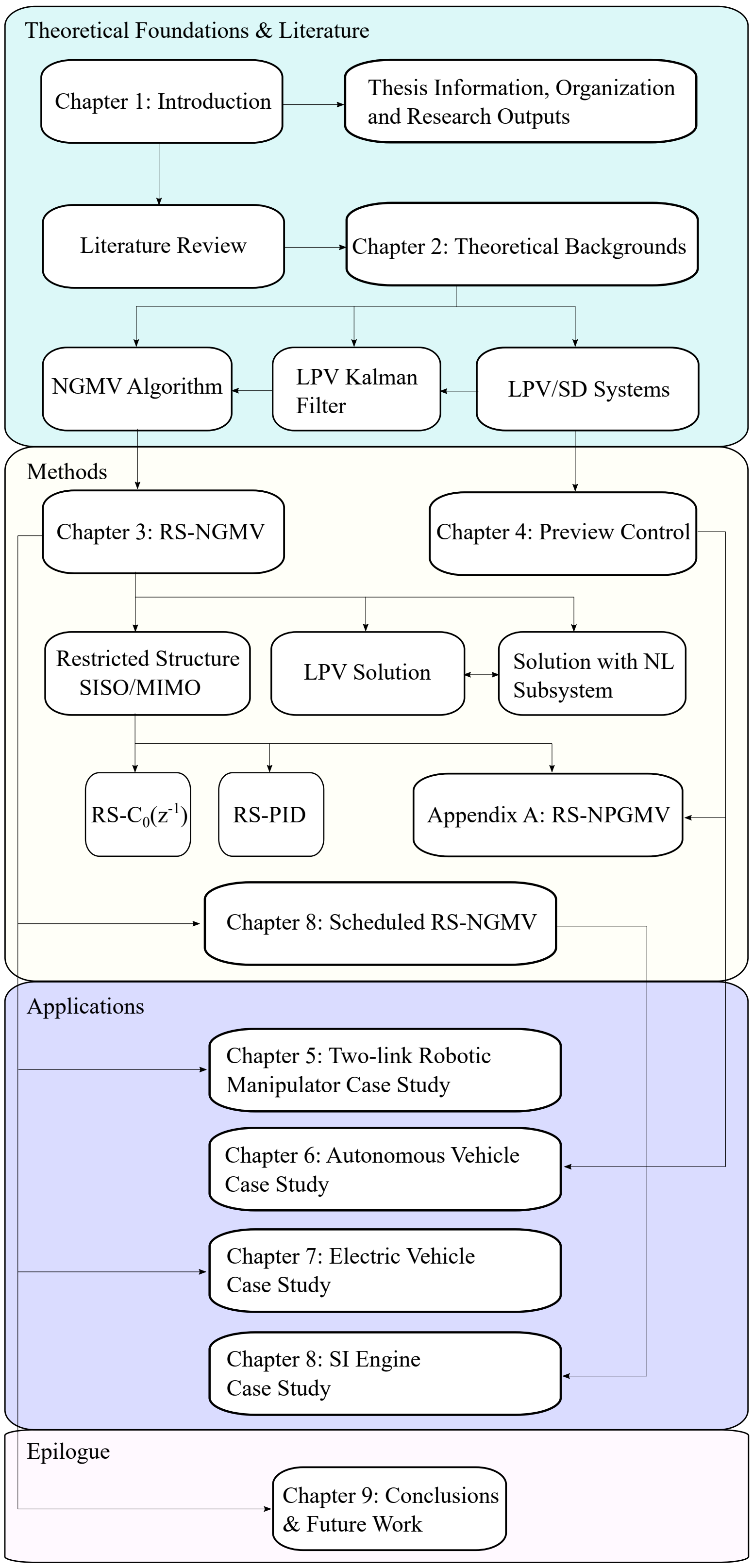

Theoretical Foundations & Literature
Chapter 1: Introduction
Thesis Information, Organization and Research Outputs
Literature Review
Chapter 2: Theoretical Backgrounds
NGMV Algorithm
LPV Kalman Filter
LPV/SD Systems
Methods
Chapter 3: RS-NGMV
Chapter 4: Preview Control
Restricted Structure SISO/MIMO
LPV Solution
Solution with NL Subsystem
RS-C$_0$(z^{-1})
RS-PID
Appendix A: RS-NPGMV
Chapter 8: Scheduled RS-NGMV
Applications
Chapter 5: Two-link Robotic Manipulator Case Study
Chapter 6: Autonomous Vehicle Case Study
Chapter 7: Electric Vehicle Case Study
Chapter 8: SI Engine Case Study
Epilogue
Chapter 9: Conclusions & Future Work

Figure 1.2: Conceptual flow of the thesis.

- **Chapter 2:** Discusses the theoretical backgrounds of the thesis. The methods behind the development of the NGMV control algorithm have been explained. The state-space system representations, which will be used throughout the thesis, have been introduced.

- **Chapter 3:** The Restricted Structure NGMV controller is derived in this chapter. The fundamental Restricted Structure control structure is formulated for both Single Input Single Output (SISO) and Multi Input Multi Output (MIMO) cases. Next, the RS-NGMV optimization problem is presented and solved for first LPV systems and then systems that include input non-linearities. The control structure is illustrated and a design procedure is suggested.

- **Chapter 4:** Presents a new approach to the preview control. As an introduction to the preview control theory, a literature review is made. Then, the SDRE approach has been introduced for the problem and the SDRE/LPV-RE Preview controllers are derived.

- **Chapter 5:** Covers the two-link robotic manipulator case study. The manipulator model dynamics are described using the quasi-LPV form (a special class in LPV systems). Then the RS-NGMV controller is designed with the PID structure. Simulation results discuss the performance of the controller for the reference tracking problem of the robot link positions.

- **Chapter 6:** Presents the LPV-RE Preview control implementation for a LPV-modelled autonomous car performing lane-change manoeuvres under varying longitudinal speeds. The controller uses the future reference input for calculating the optimal signals. Its performance is demonstrated with the simulation results and compared to the LQ-Preview control.

- **Chapter 7:** Takes on the longitudinal speed tracking problem for an EV and implements the RS-NGMV control method for handling known disturbances to the system such as the road grade and aerodynamic drag forces. The simulation results show the RS-NGMV's performance under well-known driving cycles and constant speed cruise-control. Compared to the chapter 5, the state-space quasi-LPV model in chapter 7 contains input disturbances in addition. This is therefore an extra feature for the controller to handle, as the impact of input disturbances are significant especially if they are time-varying.

- **Chapter 8:** Proposes the Scheduled RS-NGMV controller and its case study on a linearised Spark-Ignition engine model.
- **Chapter 9:** Finalizes the thesis findings with a summary of the key results and the discussion of future works.
- **Appendix A:** Proposes the RS-NPGMV control structure.
- **Appendix B-F:** Provides the Matlab/SIMULINK codes and tips developed for the application case studies in Chapter 5-8.

1.4 List of Publications

The list below presents the contributions of this thesis that are published and submitted. The remaining contributions are currently in preparation and will be submitted for publication.

- C. Cebeci, M.J. Grimble, R. Katebi and L.F. Recalde, "Restricted Structure Non-Linear Generalized Minimum Variance Control of a 2-Link Robot Arm," *UKACC 12th International Conference on Control*, pages 367–372, Sheffield, UK, 2018.

- C. Cebeci, M.J. Grimble, L. Recalde-Camacho and R. Katebi, "SDRE Preview Control for a LPV Modelled Autonomous Vehicle," *3rd IFAC Workshop on Linear Parameter-Varying Systems*, Eindhoven, The Netherlands, 2019.
- C. Cebeci, M.J. Grimble, "Longitudinal Speed Tracking of an Electric Vehicle Using Restricted Structure NGMV Control Method," *European Control Conference*, London, UK, 2022. Submitted.

Chapter 2

Theoretical Backgrounds

"*The ancients stole all our great ideas from us.*"

Mark Twain, Autobiography of Mark Twain.

In this chapter, the methods and concepts that make the foundations of this thesis will be analysed from a general perspective. Firstly, the basic MV, GMV and NGMV structures will be summarized briefly. Since the algorithms developed in this thesis aim to deal with the control of non-linear systems classified as the LPV and SD systems, an introduction to these systems will then follow.

Next, these two paths will merge under the final NGMV formulation which also contains the augmented state-space model that is used throughout the entire thesis to represent the systems that are controlled.

2.1 MV Control

MV control systems are represented by polynomial models. The scalar system representation for this example is the Contemporaneous Auto-Regressive Moving-Average (CARMA) polynomial model in (2.1), a combination of the plant model (minimum-phase/invertible) and noise terms,

$$y(t) = \frac{B}{A}u(t-k) + \frac{C}{A}\xi(t), \tag{2.1}$$

where CARMA polynomial terms A,B,C and G are functions of z^{-1}. However, the indication will not be used including the polynomial terms of sections 2.2 and 2.3 to avoid notational complexity and because these are the only polynomial examples used in this thesis. The white noise term $\xi(t)$ contains the past information of the signals up until the current time instant t.

The optimal k-steps ahead output is expressed by,

$$y(t+k) = \frac{B}{A}u(t) + \frac{C}{A}\xi(t+k),$$

note that the future noise terms $\xi(t+k)$ are random and cannot be estimated so their correlation with other terms will be considered zero. The output $y(t+k)$ can be rewritten using the Diophantine equation $C = AF + z^{-k}G$,

$$y(t+k) = F\xi(t+k) + \frac{B}{A}u(t) + \frac{G}{A}\xi(t).$$

Extracting the $\xi(t)$ from the output in (2.1),

$$\xi(t) = \frac{B}{A}y(t) - \frac{B}{C}u(t-k),$$

and eliminating it from the expanded $y(t+k)$ above,

$$y(t+k) = F\xi(t+k) + (\frac{B}{A} - z^{-k}\frac{BG}{AC})u(t) + \frac{G}{C}y(t),$$

and finally by applying the Diophantine equation the k-steps ahead output is given as,

$$y(t+k) = F\xi(t+k) + \frac{BF}{C}u(t) + \frac{G}{C}y(t). \tag{2.2}$$

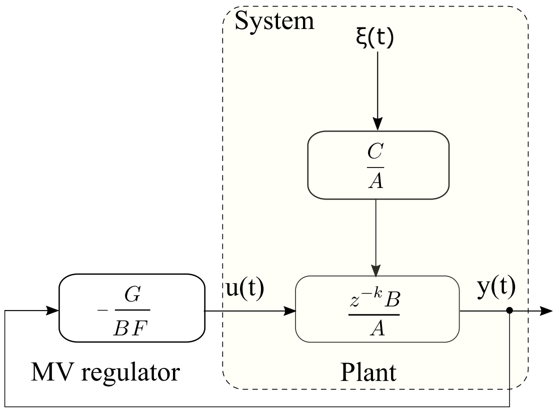


Figure 2.1: MV controller.

The MV cost function is then arranged and simplified in the steps below,

$$\begin{aligned} J &= E\{y^2(t+k)\} \\ &= E\{F\xi(t+k) + \frac{BF}{C}u(t) + \frac{G}{C}y(t)\}^2, \end{aligned}$$

because the future noise terms are independent of control and outputs, their cross products vanish and the cost function becomes,

$$\begin{aligned} J &= E\{\frac{BF}{C}u(t) + \frac{G}{C}y(t)\}^2 + E\{F\xi(t+k)\}^2 \\ &= E\{\frac{BF}{C}u(t) + \frac{G}{C}y(t)\}^2 + (1 + {f_1}^2 + \ldots + {f_{k-1}}^2)\sigma_e^2. \end{aligned}$$

It is seen that the best way of minimizing J is to set the output terms $\frac{BF}{C}u(t) + \frac{G}{C}y(t) = 0$ which is possible by the control strategy,

$$u(t) = -\frac{G}{BF}y(t), \tag{2.3}$$

leaving only the minimum possible output variance $J_{min} = (1+{f_1}^2+\ldots+{f_{k-1}}^2)\sigma_e^2$, thus giving the method its name as Minimum Variance control. The control block diagram for the MV control is shown in Fig. 2.1.

2.2 GMV Control

The MV control theory became more prominent with the introduction of the GMV control. The original GMV control method [2] was offered as an extension of the MV controller [1] by means of modifying its cost function to include weighted control terms as shown in,

$$J = E\{y^2(t+k) + \lambda u^2(t)\}, \tag{2.4}$$

minimization of which provides the GMV control law. The main motivation was to enhance the robustness characteristics of the MV method, as it was unable to handle the non-minimum phase plants. Therefore, the increased performance and the simplicity to implement made popular with industrial process control applications.

Majority of the GMV type controllers use polynomial models including this example. The GMV cost function in (2.4) may be modified to include weighted reference signals or can be formulated based on the variance of weighted error and control terms.

For example, Grimble [9] revisits the GMV problem by proposing the cost function $J = E\{\phi^2(t+k)\}$ that is the variance of the pseudo output signal defined as in,

$$\phi(t) = P_c e(t) + F_c u(t), \tag{2.5}$$

demonstrated in Fig. 2.2. The weighting terms P_c and F_c are polynomial transfer functions that are given for the scalar case,

$$P_c = \frac{P_{cn}}{P_{cd}}, F_c = z^{-k}\frac{F_{cn}}{F_{cd}}, \tag{2.6}$$

where subscripts *cn* and *cd* stand for the numerator and denominator respectively, and let $F_c = z^{-k}F_{ck}$.

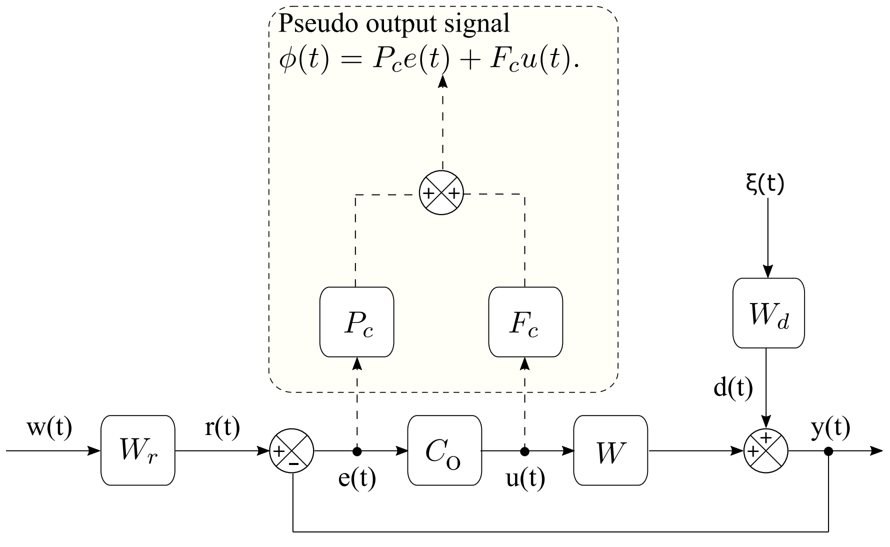


Figure 2.2: GMV controller.

The output in (2.5) can be expanded in a similar form to that of the MV system output in (2.1),

$$\begin{aligned}\phi(t) &= P_c\big(-z^{-k}W_k u(t) + Y_f\xi(t)\big) + F_c u(t)\\ &= z^{-k}\big(F_{ck} - P_c W_k\big)u(t) + P_c Y_f \xi(t) \qquad (2.7)\end{aligned}$$

consisting of input and white-noise terms, where $z^{-k}\big(F_{ck} - P_c W_k\big)$ is the generalized plant model and W_k is the delay free plant model. Then, using the delay operator z^{-k} and the Diophantine equation, reformulated as $P_c Y_f = \mathcal{F} + z^{-k}R$, the (2.7) becomes,

$$\begin{aligned}\phi(t) &= \big(P_c W_k - F_{ck}\big)u(t-k) + P_c Y_f \xi(t)\\ &= \mathcal{F}\xi(t) + \big(P_c W_k - F_{ck}\big)u(t-k) + R\xi(t-k). \qquad (2.8)\end{aligned}$$

It is known that future noise terms are statistically independent of the rest. When minimizing the cost function $J = E\{\phi^2(t+k)\}$, the product of the statistically independent terms can be cancelled out hence the control signal for obtaining the minimum variance can be derived in a very similar strategy to the MV control solution.

Consider the future output signal,

$$\phi(t+k) = \mathcal{F}\xi(t+k) + \big(P_c W_k - F_{ck}\big)u(t) + R\xi(t),$$

and extract the noise term $\xi(t)$ from (2.7),

$$\xi(t) = \frac{\phi(t) - \big(P_c W_k - F_{ck}\big)u(t-k)}{P_c Y_f}.$$

When the $\xi(t)$ term above is substituted in the output $\phi(t+k)$ it becomes,

$$\phi(t+k) = \mathcal{F}\xi(t+k) + \frac{P_c Y_f \big(P_c W_k - F_{ck}\big) - z^{-k}\big(P_c W_k - F_{ck}\big)R}{P_c Y_f}u(t) + \frac{R\phi(t)}{P_c Y_f},$$

Then, using the Diophantine equation as well, it is clear that the GMV control solution to minimize the $J = E\{\phi^2(t+k)\}$ is given by,

$$u(t) = -\frac{R}{\big(P_c W_k - F_{ck}\big)\mathcal{F}}\phi(t). \tag{2.9}$$

When the term $\phi(t)$ is substituted and using the polynomials G,H and the cost function weights defined before, the solution can be further simplified as,

$$u(t) = -\frac{R}{\big(F_{ck} - \mathcal{F}{Y_f}^{-1}W_k\big)Y_f}e(t). \tag{2.10}$$

The end result is the GMV controller whose structure is very similar to its predecessor the MV controller. Note that this section only aims to explain the control philosophy, thus some of the formulation have been skipped for the sake of simplicity. For a detailed derivation, references [7, 9] are suggested. The GMV controller can be implemented as illustrated in the control diagram in Fig. 2.3 with the inner loop that contains the delay free plant model W_k. Although there

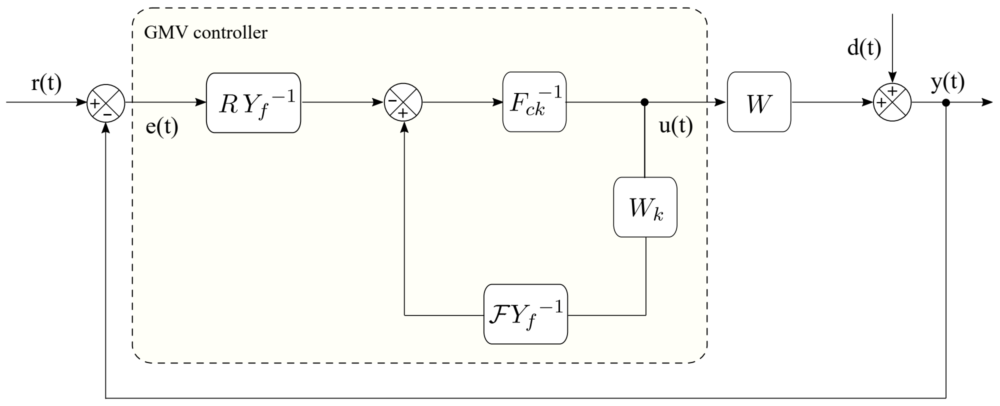


Figure 2.3: GMV controller implementation.

has been many years since the original approach, the GMV still attracts attention from the researchers. There have been rather recent attempts such as the GMV control approach in [29] which offers state-space solutions using a Kalman filter for the output predictions or the data-driven MV approach in [30].

2.3 NGMV Control

The initial versions of the NGMV used the same closed-loop feedback control system structure as that of the GMV in [9] described in the previous section. The NGMV controllers have the ability to revert to the GMV if the plant is linear.

The systems and the control law in the initial version were represented by polynomial expressions as well. However, main difference was that the plant model was allowed to be non-linear. The non-linear plant model is given by the statement below in which k denotes the delays,

$$Wu(t) = z^{-k}W_k u(t). \tag{2.11}$$

The objective of the NGMV algorithm is to minimize the cost function $J = E\{\phi^2(t+k)\}$ that is the variance of the signal $\phi(t) = P_c e(t) + F_c u(t)$, same as the GMV but there are a couple more differences.

Firstly, the control weighting term F_c could be non-linear to compensate for plant or actuator non-linearities,

$$F_c u(t) = z^{-k} F_{ck} u(t). \tag{2.12}$$

Second, the non-linear operator $\big(P_c W_k - F_{ck}\big)$ is restricted to be stably invertible to ensure the closed-loop stability. The solution procedure is also quite similar to the GMV. The variance term $\phi(t)$ is split into statistically independent terms like in (2.7) and (2.8) . Then the control signal that cancels out the terms it can affect is chosen as the NGMV controller,

$$u(t) = \frac{-R}{F_{ck} - P_c W_k} \phi(t), \tag{2.13}$$

which can be implemented using the same loop structure of Fig. 2.3 with,

$$u(t) = -\frac{R Y_f^{-1}}{F_{ck} - F Y_f^{-1} W_k} e(t). \tag{2.14}$$

An interesting implementation issue with the type of the diagram used, is that the closed-loop feedback control system faces the algebraic loop problem because of the inner-loop. Algebraic loops can become computational burdens and may prove difficult to solve but there are remedies to fix them. The control solution can be implemented in a different structure or the problem may be avoided much easily by introducing a unit-step delay before the inner feedback loop. In some cases, The Matlab/Simulink can solve it without an intervention. For the implementations in this thesis, the unit-step delay has worked.

This section summarizes the control philosophy of the classical NGMV solution. The modern NGMV technique will be revisited in much more detail.

2.4 LPV and State-Dependent Systems

Most physical systems are non-linear in nature thus the design of controllers that can handle complex models is important. The traditional approach to deal with non-linearities is applying linearisation procedures at designated operating points. Depending on the application, there can be various operating points and corresponding controllers designed for each of them. The controllers are then scheduled with respect to operating conditions. This technique is called the gain scheduled control [31] and has been popular especially in the aerospace industry.

Although the method has shown success in many applications, there are certain drawbacks to using it. For example, scheduling multi-variable feedback controllers can be daunting and time-consuming. Stability also can not be guaranteed other than at the design points. These facts had set the stage to address LPV techniques as a remedy [32, 33]. According to [34], the need for a framework that can handle both non-linearities and time-varying dynamical aspects mixed with gain-scheduling ideas has led actually to the rise of LPV systems.

Over the years, LPV systems have proven useful in many application areas including wind turbine control, automotive and aerospace industries [35 – 37]. A survey study [38] provides a wide overview of LPV controllers grouped by application areas. The results have been validated by experiments and simulations. The study claims that LPV control has evolved into an effective tool to address non-linear control problems and has been attracting an increasing attention from researchers.

However, LPV control has its own limitations in dealing with non-linear systems. It does not come with a standardized methodological approach for assuring stability. The derivation of LPV controllers do not go through sophisticated stability analysis tools like the Lyapunov theory, for example. Therefore, stability cannot always be guaranteed and mostly remain local or application specific.

Mathematically, LPV systems are defined as linear dynamical systems that consist of parameters whose values change over time. The following examples are used to explain the notion.

The state-space representation of a Linear Time-Invariant (LTI) system is given by the model below,

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t),\\ y(t) &= Cx(t) + Du(t),\end{aligned}$$

with state matrices A,B,C and D that are constant. The $x(t)$ is the state vector and $u(t)$ is the control input. On the other hand, a Linear Time-Varying (LTV) system differ from the LTI by depending on time. The state-space representation of a LTV system is as,

$$\begin{aligned}\dot{x}(t) &= A(t)x(t) + B(t)u(t),\\ y(t) &= C(t)x(t) + D(t)u(t).\end{aligned}$$

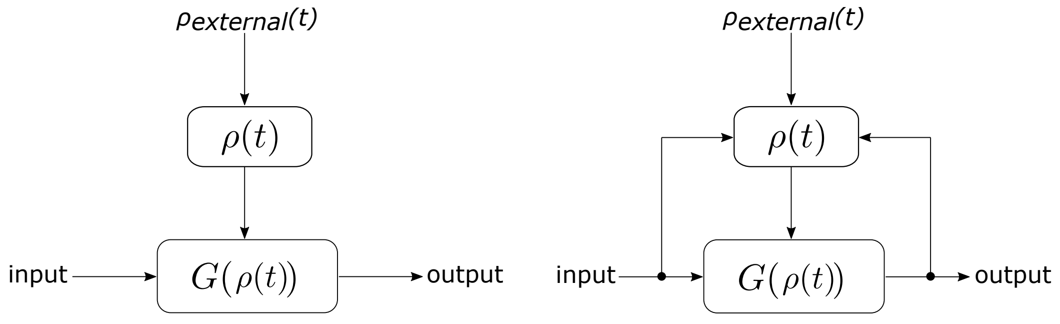
The formulation of LPV systems is distinguished from LTI and LTV systems as,

$$\begin{aligned}\dot{x}(t) &= A\big(\rho(t)\big)x(t) + B\big(\rho(t)\big)u(t),\\ y(t) &= C\big(\rho(t)\big)x(t) + D\big(\rho(t)\big)u(t),\end{aligned}$$

where $\rho(t)$ is the time-varying parameter vector that is external and unknown a-priori but can either be measured or estimated. It is also referred to as the scheduling variable vector in some cases.

The varying parameters are categorized formally as exogenous and endogenous. The parameter $\rho(t)$ is called exogenous if it is an external variable to the system (like in LPV case) and called endogenous if it is a function of the states.

The latter is a special case and it is referred to as a Quasi Linear Parameter-Varying (qLPV) system. It is a useful technique to approximate non-linear sys-



(a) LPV system (b) qLPV or SD system

Figure 2.4: LPV, qLPV and SD system structures.

tems. A well-known academic example from [39] is provided below to explain the qLPV systems:

$$\dot{x}(t) = -x(t)^2, \tag{2.15}$$

is a non-linear system and can be re-arranged in this fashion,

$$\dot{x}(t) = -\rho(t)x(t), \tag{2.16}$$

where $\rho(t) = x(t) \in \mathcal{R}$. This equation describes the original non-linear system indeed but has a linear form matching the general LPV system representation. While the statement is true it is not entirely precise because a qLPV system may also be function of inputs $u(t)$ and scheduling parameters $p(t)$ that are chosen by the designer. If the parameter $\rho(t)$ only depends on the states, it might be more accurate to classify it as state-dependent.

Before going any further, let us also clarify that unless a specific system modelling is being mentioned, this thesis treats both the LPV and qLPV under the joint term LPV since the control solution applies to both and also to be in accord with the LPV research literature terminology.

Fig. 2.4 demonstrates the LPV, qLPV and SD systems graphically. It is illustrated that the LPV system includes only the external variables $\big(\rho(t) = \rho_{external}(t)\big)$. Whereas in the qLPV case, $\rho(t)$ may also include a set of selected system states, inputs and the time-varying known parameter vector $p(t)$. For the SD case, its the states in other words, $\rho(t) = x(t)$. It is worth mentioning that because they overlap on the states, qLPV systems are sometimes expressed as SD in the literature. To clarify let us present their formulation as well. A qLPV system is represented using,

$$\dot{x}(t) = A\big(x(t), u(t), p(t)\big)x(t) + B\big(x(t), u(t), p(t)\big)u(t), \tag{2.17}$$

$$y(t) = C\big(x(t), u(t), p(t)\big)x(t) + D\big(x(t), u(t), p(t)\big)u(t). \tag{2.18}$$

and a SD system is presented with the model,

$$\dot{x}(t) = A\big(x(t)\big)x(t) + B\big(x(t)\big)u(t), \tag{2.19}$$

$$y(t) = C\big(x(t)\big)x(t) + D\big(x(t)\big)u(t). \tag{2.20}$$

LPV system representations mentioned so far are in their most generalized form. In fact, LPV system parametrisation is not unique and there are several modelling approaches such as polytope, affine parameter-dependent, polynomial parameter-dependent, rational parameter-dependent or Linear-Fractional Transformation (LFT). It is also common to extract LPV system models via Jacobian linearisation as in [40]. However, concerning the scope of this work the information provided here is sufficient.

2.5 NGMV Control for State-Space Systems

Having summarized the core concepts of MV strategies, the LPV and SD systems, it is time to introduce the modern NGMV control approach which is a descendant of the original NGMV controller described previously. The method diversifies to provide the optimal NGMV control solution for the multi-variable state-space systems including LPV systems. It also serves as the backbone of the novel RS-

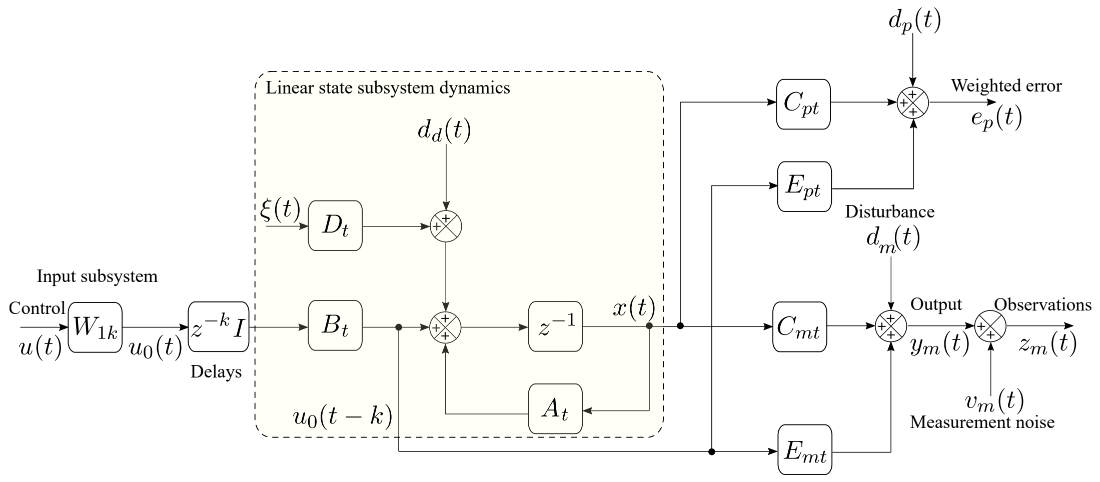


Figure 2.5: The LPV plant and signal models.

NGMV approach, thus will be studied in detail under this section. From here on, the term NGMV will be used to represent this approach only instead of the original polynomial version.

2.5.1 State-Space Plant Models and Signals

Consider the system in Fig. 2.5, it is composed of a set of individual subsystems that are described as plant, input, error weighting and disturbance models using matrices and signals. Note that the plant model that is mentioned is of the linear form as also highlighted in the figure. The model-based design of the NGMV algorithm does not only allow access to a wider range of non-linear systems but also serves well for general design processes by being modification-friendly. The linear plant subsystem is denoted by $W_0(z^{-1})$ and may be LPV, qLPV, SD, LTV or even LTI. The NGMV control law will remain the same except the matrices.

For notational simplicity, the index t will be used to denote the state-space matrices A_t,B_t,C_t or similarly D_t. To reference the corresponding subsystem, following subscripts are used.

- r: Reference subsystem,

- 0: Linear or LPV/SD subsystem,
- u: Input subsystem,
- d: Disturbance subsystem or deterministic disturbance signal index,
- p: Error weighting P_c subsystem,
- m: Measurement index,
- w: Deterministic component index of some signals.

The vectors that represent the signals used in the overall system are defined as,

- $x(t)$: Vector of augmented system states,
- $u_0(t)$: Vector of control signals applied to the linear state subsystem,
- $u(t)$: Vector of control signals applied to the input subsystem,
- $d_m(t)$: Vector of known output-disturbance signals to be measured,
- $d_p(t)$: Vector of known output-disturbance signals to be controlled,
- $d_d(t)$: Vector of known input-disturbance signals,
- $\xi(t)$: Zero-mean white noise as stochastic piece of the input-disturbance driven by the process noise signal $\zeta(t)$,
- $r_w(t)$: Vector of deterministic reference signals,
- $r(t)$: Vector of deterministic reference model output.

The individual subsystems mentioned here will be presented in detail in 2.5.2. These subsystems make up for the entire NGMV system and they can be incorporated into the augmented state-space model,

$$x(t+1) = A_t x(t) + B_t u_0(t-k) + D_t \xi(t) + d_d(t). \tag{2.21}$$

The system's controlled output $y(t)$ is defined as in the equation,

$$y(t) = C_t x(t) + E_t u_0(t-k) + d(t). \tag{2.22}$$

The augmented system's measured output $y_m(t)$ is expressed by the equation,

$$y_m(t) = C_{mt} x(t) + E_{mt} u_0(t-k) + d_m(t), \tag{2.23}$$

The terms controlled and measured separate the outputs as measurable and not measurable but controlled. In some cases, they can be the same $y(t) = y_m(t)$.

The measured outputs $y_m(t)$ are combined with the sensor or measurement noise $v_m(t)$ to obtain the observation signal $z_m(t)$,

$$z_m(t) = C_{mt} x(t) + E_{mt} u_0(t-k) + d_m(t) + v_m(t). \tag{2.24}$$

The reference signal $r(t)$ is assumed to be deterministic and defined as,

$$r(t) = W_w r_w(t), \tag{2.25}$$

where W_w is a linear ideal response model and is also a function of the unit-delay operator z^{-1} but from this point on, the models W will be simply used without this notation.

The system's output tracking error $e(t)$ is given by the definition,

$$e(t) = r(t) - y(t). \tag{2.26}$$

The weighted error signal $e_p(t) = P_c e(t)$ is denoted as in the model below,

$$e_p(t) = C_{pt} x(t) + E_{pt} u_0(t-k) + d_p(t). \tag{2.27}$$

The terms C_{pt} and E_{pt} are derived from the model of error weight P_c and in section 2.5.2, it will be explained how to. The input subsystem may be non-linear, and formulated by,

$$u_0(t) = W_1 u(t) = z^{-k} W_{1k} u(t). \tag{2.28}$$

which contains the unstructured plant model W_{1k}. This is an important feature also shared with the previous NGMV algorithms. The fact that the input subsystem model is generalized as unstructured provides a major advantage to the NGMV algorithms as it gives them the capability to deal with black box terms. The term black box could include any system from devices to algorithms, even biological systems like human brain whose inputs and outputs are observable but inner characteristics are completely unknown.

For example, a system model could be considered as black box if it is defined off-limits to the designer (closed source like firmware, due to copyrights). In that case, a controller that can deal with the unknown physical structure without having a need to go through major modifications, might be useful. All that is needed for the controller is the input and output information. This is a great benefit of the NGMV, since most industrial applications enforce legal restrictions, and modelling of a system from scratch may prove difficult or time-consuming for the control design engineers.

2.5.2 Augmented State-Space System Model

The total state-space model used by the Kalman filter and the NGMV control solutions is presented in this section. The RS-NGMV that will be introduced in chapter 3 also uses it as its state-space model. It is called the augmented state-space model because the information from the different subsystems are stacked in it. It uses the set of equations in $(2.21 - 2.24)$ and (2.27) which will be expanded in detail now starting with the state vector $x(t)$ in,

$$x(t) = (x_0(t), x_d(t), x_p(t), x_u(t))^T, \tag{2.29}$$

combining the LPV plant states $x_0(t)$, disturbance states $x_d(t)$, error states $x_p(t)$ and control input weighting states $x_u(t)$, respectively. Using $x_0(t)$, the LPV plant model, the measure output and observation signals are represented by the set of equations,

$$x_0(t+1) = A_{0t}x_0(t) + B_{0t}u_0(t-k) + D_{0t}\zeta_0(t) + G_{0t}d_{0d}(t), \tag{2.30}$$

$$y(t) = C_{0t}x_0(t) + E_{0t}u_0(t-k) + d_0(t), \tag{2.31}$$

$$y_m(t) = C_{0mt}x_0(t) + E_{0mt}u_0(t-k) + d_{0m}(t), \tag{2.32}$$

$$z_m(t) = y_{mt} + v_m(t), \tag{2.33}$$

where the controlled and the measured output disturbances are,

$$d_0(t) = d(t) + y_d(t), \tag{2.34}$$

$$d_{0m}(t) = d_m(t) + y_{dm}(t), \tag{2.35}$$

respectively. Both disturbance signals are composed of deterministic $(d(t), d_m(t))$, and stochastic $(y_d(t), y_{dm}(t))$ components.

The deterministic output disturbance component $d(t)$ is obtained by using the zero-mean white noise signal $w(t)$,

$$d(t) = W_d w(t), \tag{2.36}$$

where the component W_d is defined as the output disturbance model,

$$W_d = C_{dt}(zI - A_{dt})^{-1}D_{dt}. \tag{2.37}$$

Having described the LPV plant model, the disturbance state $x_d(t)$ driven by the stochastic signal $w(t)$, is used to define the disturbance model,

$$x_d(t+1) = A_{dt}x_d(t) + D_{dt}w(t), \tag{2.38}$$

$$y_d(t) = C_{dt}x_d(t), \tag{2.39}$$

$$y_{dm}(t) = C_{dmt}x_d(t). \tag{2.40}$$

Then, the weighted error state model is implemented as in,

$$x_p(t+1) = A_{pt}x_p(t) + B_{pt}(r(t) - y(t)), \tag{2.41}$$

$$y_p(t) = C_{pt}x_p(t) + E_{pt}(r(t) - y(t)), \tag{2.42}$$

whose matrices for example, can be obtained from the NGMV cost-function error weighting term $P_c(z^{-1})$ by defining it in discrete-time transfer function form and performing proper conversions (e.g. using Matlab's ssdata(sys) commands). The same procedure applies for the NGMV cost-function control weighting term F_c to derive control input weighting state-space model as in below,

$$x_u(t+1) = A_{ut}x_u(t) + B_{ut}u_0(t-k), \tag{2.43}$$

$$y_u(t) = C_{ut}x_u(t) + E_{ut}u_0(t-k), \tag{2.44}$$

on the condition that $(P_cW_k - F_{ck})$ is stably invertible [16]. If desired, an extra weighting can also be introduced on some state variables using,

$$y_x(t) = C_{p0t}x_0(t) + E_{pdt}x_d(t). \tag{2.45}$$

However, this leads to the modification of NGMV cost function by re-defining it as the SD-NGMV cost function which is not needed for the work in this thesis (for details see [14]). Similar to the state-weighting, $x_u(t)$ related control input state-space models can be avoided from using in the augmented state-space model especially when the NGMV solutions do not consider the unstructured non-linear subsystems. Revisiting (2.41) and substitute the $y(t)$ with (2.31),

$$x_p(t+1) = A_{pt}x_p(t) + B_{pt}\Big(r(t) - \big(C_{0t}x_0(t) + E_{0t}u_0(t-k) + d_0(t)\big)\Big), \quad (2.46)$$

and then using $d_0(t) = d(t) + y_d(t)$ along with (2.39) for (2.46), the result will return as (2.47). Then, similarly re-arranging the (2.42) to the output in (2.48)is obtained. The total weighted error model will be updated as,

$$x_p(t+1) = A_{pt}x_p(t) + B_{pt}\big(r(t) - C_{0t}x_0(t) - E_{0t}u_0(t-k) - C_{dt}x_d(t) - d(t)\big), \quad (2.47)$$

$$y_p(t) = C_{pt}x_p(t) + E_{pt}\big(r(t) - C_{0t}x_0(t) - E_{0t}u_0(t-k) - C_{dt}x_d(t) - d(t)\big), \quad (2.48)$$

Now, it can be shown how models above combine into the augmented state $x(t)$ and subsequently into the augmented state-space model,

$$\begin{pmatrix} x_0(t+1) \\ x_d(t+1) \\ x_p(t+1) \\ x_u(t+1) \end{pmatrix} = \begin{pmatrix} A_{0t} & 0 & 0 & 0 \\ 0 & A_{dt} & 0 & 0 \\ -B_{pt}C_{0t} & -B_{pt}C_{0t} & A_{pt} & 0 \\ 0 & 0 & 0 & A_{ut} \end{pmatrix} \begin{pmatrix} x_0(t) \\ x_d(t) \\ x_p(t) \\ x_u(t) \end{pmatrix} + \begin{pmatrix} B_{0t} \\ 0 \\ -B_{pt}E_{0t} \\ B_{ut} \end{pmatrix} u_0(t-k)$$
$$+ \begin{pmatrix} D_{0t} & 0 \\ 0 & D_{dt} \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \zeta(t) \\ \omega(t) \end{pmatrix} + \begin{pmatrix} G_{0t} & 0 \\ 0 & 0 \\ 0 & B_{pt} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} d_{0d}(t) \\ (r(t) - d(t)) \end{pmatrix}. \quad (2.49)$$

With the information from the SS disturbance models, the controlled and measured output signals defined in (2.31) and (2.32) of the LPV plant are expanded,

$$y(t) = C_{0t}x_0(t) + E_{0t}u_0(t-k) + C_{dt}x_d(t) + d(t), \tag{2.50}$$

$$y_m(t) = C_{0mt}x_0(t) + E_{0mt}u_0(t-k) + C_{dmt}x_d(t) + d_m(t). \tag{2.51}$$

Gathering the output equations for the weighted error signals we have,

$$\begin{pmatrix} y_p(t) \\ y_u(t) \\ y_x(t) \end{pmatrix} = \begin{pmatrix} -E_{pt}C_{0t} & E_{pt}C_{dt} & C_{pt} & 0 \\ 0 & 0 & 0 & C_{ut} \\ C_{p0t} & C_{pdt} & 0 & 0 \end{pmatrix} \begin{pmatrix} x_0(t) \\ x_d(t) \\ x_p(t) \\ x_u(t) \end{pmatrix} + \begin{pmatrix} -E_{pt}E_{0t} \\ E_{ut} \\ 0 \end{pmatrix} u_0(t-k) + \begin{pmatrix} E_{pt}(r(t)-d(t)) \\ 0 \\ 0 \end{pmatrix} \tag{2.52}$$

Finally, by denoting the matrices through (2.49−2.52) as below, the derivation of the augmented SS system model in (2.21−2.23) and (2.27) is revealed completely.

$$A_t = \begin{pmatrix} A_{0t} & 0 & 0 & 0 \\ 0 & A_{dt} & 0 & 0 \\ -B_{pt}C_{0t} & -B_{pt}C_{0t} & A_{pt} & 0 \\ 0 & 0 & 0 & A_{ut} \end{pmatrix}, B_t = \begin{pmatrix} B_{0t} \\ 0 \\ -B_{pt}E_{0t} \\ B_{ut} \end{pmatrix},$$

$$D_t = \begin{pmatrix} D_{0t} & 0 \\ 0 & D_{dt} \\ 0 & 0 \\ 0 & 0 \end{pmatrix}, G_t = \begin{pmatrix} G_{0t} & 0 \\ 0 & 0 \\ 0 & B_{pt} \\ 0 & 0 \end{pmatrix},$$

$$C_t = \begin{pmatrix} C_{0t} & C_{dt} & 0 & 0 \end{pmatrix}, E_t = E_{0t},$$

$$C_{mt} = \begin{pmatrix} C_{0mt} & C_{dmt} & 0 & 0 \end{pmatrix}, E_{mt} = E_{0mt}$$

$$C_{pt} = \begin{pmatrix} -E_{pt}C_{0t} & E_{pt}C_{dt} & C_{pt} & 0 \\ 0 & 0 & 0 & C_{ut} \\ C_{p0t} & C_{pdt} & 0 & 0 \end{pmatrix}, E_{pt} = \begin{pmatrix} -E_{pt}E_{0t} \\ E_{ut} \\ 0 \end{pmatrix},$$

$$d_d(t) = G_t \begin{pmatrix} d_{0d}(t) \\ (r(t) - d(t)) \end{pmatrix}, d_p(t) = \begin{pmatrix} E_{pt}(r(t) - d(t)) \\ 0 \\ 0 \end{pmatrix}, \xi(t) = \begin{pmatrix} \zeta(t) \\ w(t) \end{pmatrix}.$$

2.5.3 Prediction Model and LPV Kalman Filter

The NGMV controller utilizes a LPV Kalman filter for both $k-$steps ahead predictions of the output signals (characteristics of MV based optimization) and the estimation of states that cannot be measured.

The Kalman filter aforementioned is of the predictor-corrector form which was originally proposed in [41]. In previous NGMV related research such as the [25], the Extended Kalman Filter (EKF) was utilized for a NPGMV wind turbine control problem. In this case, the LPV model was extracted from the Jacobian linearisation of the non-linear plant thus it suited the use of EKF which involved Jacobian linearisation. In our case, the LPV Kalman filter is used because the models are assumed to be in LPV or SD form. Moreover, the use of the EKF that requires more intense computations is unnecessary if the model is in LPV or SD form. The Kalman filter's algorithm is presented in the Fig. 2.6 where the state prediction equation is expressed by,

$$\hat{x}(t+1|t) = A_t\hat{x}(t|t) + B_t u_0(t-k) + d_d(t). \tag{2.53}$$

The notation $\hat{x}(t|j)$ refers to the time t value of $\hat{x}(t)$ (the estimate of $x(t)$) that contains the information up to and including the time j. The corrector is given

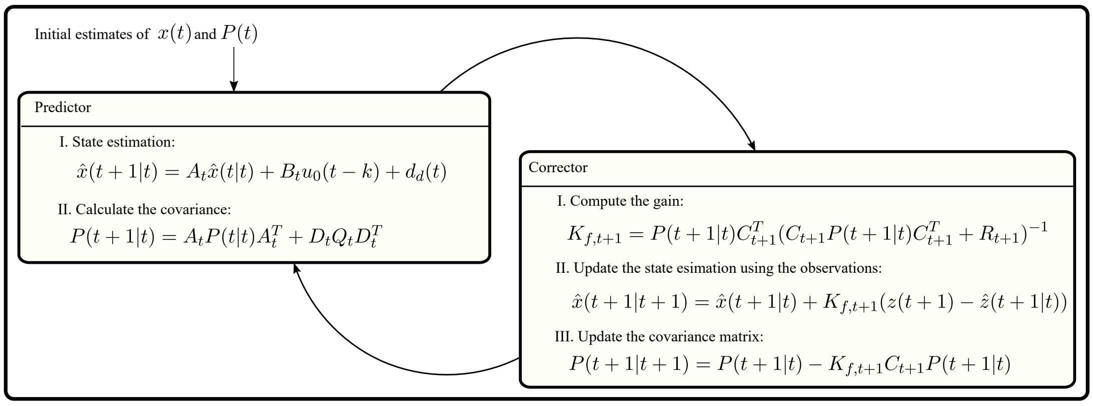

Initial estimates of $x(t)$ and $P(t)$
Predictor
I. State estimation:
$\hat{x}(t+1|t) = A_t\hat{x}(t|t) + B_t u_0(t-k) + d_d(t)$
II. Calculate the covariance:
$P(t+1|t) = A_t P(t|t) A_t^T + D_t Q_t D_t^T$
Corrector
I. Compute the gain:
$K_{f,t+1} = P(t+1|t)C_{t+1}^T(C_{t+1}P(t+1|t)C_{t+1}^T + R_{t+1})^{-1}$
II. Update the state esimation using the observations:
$\hat{x}(t+1|t+1) = \hat{x}(t+1|t) + K_{f,t+1}(z(t+1) - \hat{z}(t+1|t))$
III. Update the covariance matrix:
$P(t+1|t+1) = P(t+1|t) - K_{f,t+1}C_{t+1}P(t+1|t)$

Figure 2.6: The Kalman filter algorithm.

by the following equation,

$$\hat{x}(t+1|t+1) = \hat{x}(t+1|t) + K_{f,t+1}(z(t+1) - \hat{z}(t+1|t)), \tag{2.54}$$

where $\hat{z}(t+1|t) = C_{t+1}\hat{x}(t+1|t) + E_{t+1}u_0(t+1-k) + d_d(t+1)$. The Kalman filter corrector term $K_{f,t+1}$ is referred to as the Kalman filter gain and it is expanded in,

$$K_{f,t+1} = P(t+1|t)C_{t+1}^T(C_{t+1}P(t+1|t)C_{t+1}^T + R_{t+1})^{-1}. \tag{2.55}$$

The gain consists of process and measurement noises along with a-priori (between observations) and a-posteriori (post observations) covariance matrices detailed as:

- Process noise: $Q_t = E\{w_t w_t^T\}$ where w_t defines the noise term $d_d(t)$ of (2.53),
- Measurement noise: $R_t = E\{v_k v_k^T\}$,
- A-priori covariance: $P(t+1|t) = A_t P(t|t) A_t^T + D_t Q_t D_t^T$,
- A-posteriori covariance: $P(t+1|t+1) = P(t+1|t) - K_{f,t+1}C_{t+1}P(t+1|t)$,
- Initial covariance: $P(0|0) = E\{(x(0) - \hat{x}(0))(x(0) - \hat{x}(0))^T\}$.

The Kalman filtering procedure starts with the initial estimates of the states and covariances, then the algorithm is implemented in a recursive fashion.

The predictions of time t, are compared to the observations in the next time step $t+1$ and the state estimates are updated (i.e. they are corrected). Using the Kalman filter estimations, the $k-$steps ahead state prediction models can be constructed. First, the state model (2.21) is generalized for i-steps into the future,

$$\begin{aligned} x(t+i) = {} & A_{t+i-1}A_{t+i-2}\dots A_t x(t) + \sum_{j=1}^{i} A_{t+i-1}A_{t+i-2}\dots \\ & \dots A_{t+j}(B_{t+j-1}u_0(t+j-1-k) + D_{t+j-1}\xi(t+j-1) + d_d(t+j-1)), \end{aligned}$$

which is simplified in,

$$\begin{aligned} x(t+i) = {} & A_t^i x(t) + \sum_{j=1}^{i} A_{t+j}^{i-j}(B_{t+j-1}u_0(t+j-1-k) + D_{t+j-1}\xi(t+j-1)) \\ + \quad & d_{dd}(t+i-1), \end{aligned} \tag{2.56}$$

by defining the set of state-matrices as,

- $A_t^i = A_{t+i-1}A_{t+i-2}\dots A_t, \qquad$ if $i > 0$,
- $A_t^0 = I, \qquad$ if $i = 0$,
- $A_{t+m}^{i-m} = A_{t+i-1}A_{t+i-2}\dots A_{t+m}, \qquad$ if $i > m$,
- $A_{t+m}^0 = I, \qquad$ if $i = m$,

and known disturbances,

- $d_{dd}(t+i-1) = \sum_{j=1}^{i} A_{t+j}^{i-j} d_d(t+j-1), \qquad$ if $i > 0$,
- $d_{dd}(t-1) = 0, \qquad$ if $i = 0$.

Then, through (2.56) and definitions following it, the state predictor is introduced,

$$\hat{x}(t+k|t) = A_t^k \hat{x}(t|t) + \sum_{j=1}^{k} A_{t+j}^{k-j} B_{t+j-1} u_0(t+j-1-k) + d_{dd}(t+k-1). \qquad (2.57)$$

Secondly, the prediction error model is constructed starting with the weighted errors generalized in a similar fashion for (2.27),

$$e_p(t+i) = C_{pt+i} x(t+i) + E_{pt+i} u_0(t+i-k) + d_p(t+i). \qquad (2.58)$$

Substituting from (2.56) and the above equation becomes,

$$e_p(t+i) = C_{pt+i} A_t^i x(t) + C_{pt+i} \sum_{j=1}^{i} A_{t+j}^{i-j} B_{t+j-1} u_0(t+j-1-k) + \\ C_{pt+i} \sum_{j=1}^{i} A_{t+j}^{i-j} D_{t+j-1} \xi(t+j-1) + E_{pt+i} u_0(t+i-k) + d_{pd}(t+i), \qquad (2.59)$$

where the deterministic disturbance components of state and error models are combined,

$$d_{pd}(t+i) = d_p(t+i) + C_{pt+i} d_{dd}(t+i-1). \qquad (2.60)$$

The prediction error model is finalized in,

$$\hat{e}_p(t+k|t) = C_{pt+k} \hat{x}(t+k|t) + E_{pt+k} u_0(t) + d_p(t+k), \qquad (2.61)$$

which can be simplified by defining $d_{pd}^0(t+k) = d_{pd}(t+k) + C_{pt+k}\hat{x}(t+k|t)$,

$$\hat{e}_p(t+k|t) = E_{pt+k} u_0(t) + d_{pd}^0(t+k). \qquad (2.62)$$

The estimation of the weighted error at k-steps ahead has the relation,

$$e_p(t+k) = \hat{e}_p(t+k|t) + \tilde{e}_p(t+k), \qquad (2.63)$$

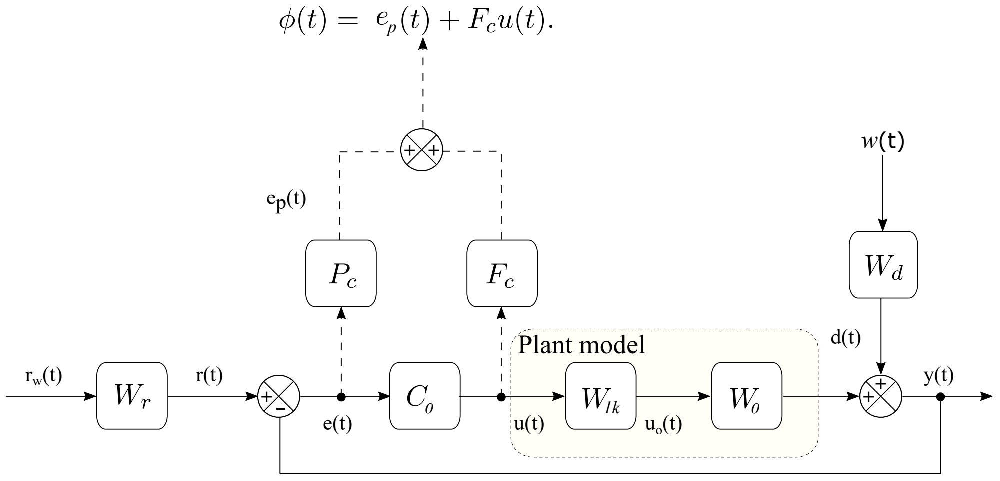


Figure 2.7: Basic NGMV control diagram.

where the estimation error satisfies.

$$\tilde{e}_p(t+k) = C_{pt+k}(x(t+k) - \hat{x}(t+k|t)) = C_{pt+k}(\tilde{x}(t+k)), \tag{2.64}$$

2.5.4 Optimization and the NGMV Control Law

With the LPVKF and prediction models set up, the NGMV cost function and its solution can now be discussed. Recall the GMV cost function of (2.5) which was also used as the NGMV cost function of the original example. The modern NGMV takes the same approach but in our example, the pseudo-output signal will be demonstrated with the simplified error terms thus becoming,

$$\phi(t) = e_p(t) + F_c u(t).$$

The control solution results from the minimization of this signal's variance,

$$J = E\{\phi^T(t+k)\phi(t+k)|t\}, \tag{2.65}$$

The basic control diagram of the NGMV shown in Fig. 2.7 contains the non-linear plant model W_k which may be separated into a delay-free linear plant subsystem W_0 and a non-linear input subsystem W_{1k} ($W_1 = z^{-k}W_{1k}$),

$$Wu(t) = z^{-k}W_{0k}W_{1k}u(t), \tag{2.66}$$

Thus, the control inputs $u(t)$ and $u_0(t)$ are related by the expression,

$$u_0(t) = W_{1k}u(t), \tag{2.67}$$

Substituting the weighted error term $e_p(t)$ with (2.27) the signal $\phi(t)$ becomes,

$$\phi(t) = C_{pt}x(t) + E_{pt}u_0(t-k) + d_p(t) + F_c u(t). \tag{2.68}$$

Recall that $F_c u(t) = z^{-k}F_{ck}u(t)$ and using (2.67), the signal is now,

$$\phi(t) = C_{pt}x(t) + (E_{pt}W_{1k} + F_{ck})u(t-k) + d_p(t), \tag{2.69}$$

whose future values are represented as,

$$\phi(t+k) = C_{pt+k}x(t) + (E_{pt+k}W_{1k} + F_{ck})u(t) + d_p(t+k). \tag{2.70}$$

The signal $\phi(t)$ is predicted by the model,

$$\hat{\phi}(t+k|t) = C_{pt+k}\hat{x}(t+k|t) + (E_{pt+k}W_{1k} + F_{ck})u(t) + d_p(t+k), \tag{2.71}$$

and the condition for optimality is $\hat{\phi}(t+k|t) = 0$ since the goal is to minimize the variances. Rewriting the cost function in (2.65) with regards to predictions and errors,

$$J = E\{\hat{\phi}(t+k)^T\hat{\phi}(t+k)|t\} + E\{\tilde{\phi}(t+k)^T\tilde{\phi}(t+k)|t\}, \tag{2.72}$$

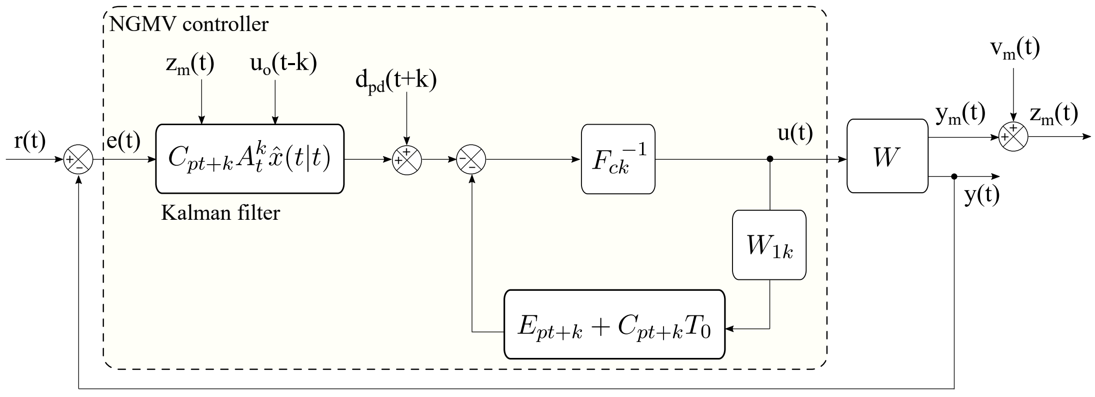


Figure 2.8: NGMV controller implementation.

The prediction errors $\tilde{\phi}(t+k)$ are independent of the control action, however, it is possible for the controller to influence the prediction values represented by $\hat{\phi}(t+k|t)$. Therefore, the minimum variance can be achieved by setting $\hat{\phi}(t+k|t) = 0$ whose solution returns the NGMV the controller in,

$$u(t) = (E_{pt+k}W_{1k} + F_{ck})^{-1}(-C_{pt+k}\hat{x}(t+k|t) - d_p(t+k)). \tag{2.73}$$

This is re-arranged below into a form more suitable for implementations,

$$u(t) = F_{ck}^{-1}\big(-C_{pt+k}\hat{x}(t+k|t) - d_p(t+k) - E_{pt+k}W_{1k}u(t)\big). \tag{2.74}$$

Alternative to (2.74), the control signal can be calculated in terms of the current state estimation value $\hat{x}(t|t)$ as in,

$$u(t) = F_{ck}^{-1}\big(-C_{pt+k}A_t^k\hat{x}(t|t) - d_{pd}(t+k) - (E_{pt+k} + C_{pt+k}T_0)W_{1k}u(t)\big), \tag{2.75}$$

where $d_{pd}(t+i) = d_p(t+i) + C_{pt+i}d_{dd}(t+i-1)$ and the operator T_0 of the LPVKF prediction equations is $T_0 = \sum_{j=1}^{i} A_{t+j}^{i-j}B_{t+j-1}z^{j-1-k}$. Finally, the NGMV controller in (2.75) is implemented as in the Fig. 2.8.

2.6 Closing Remarks

This sums up the NGMV method that is the basis of the RS-NGMV to be introduced. There are other implementation strategies for the approach with different benefits to each other [16]. The designer's background and understanding of the method can have a huge impact. Some NGMV control diagrams presented in the literature are only conceptual and are good for understanding the intuitions. However, the designer might have to resort to other structures when it comes to the implementation. Another important aspect is the choice of weightings. It is a highly crucial part of the design procedure, as it is related to the stability of the controller. More emphasis to design procedures will be given in the next chapter.

Chapter 3

RS-NGMV Control

"Nothing great is created suddenly, any more than a bunch of grapes or a fig. If you tell me that you desire a fig, I answer you that there must be time. Let it first blossom, then bear fruit, then ripen."

Epictetus, Discourses.

This chapter is organized as follows: In the first section the so-called RS controller is introduced by presenting the SISO and MIMO designs. Then, the parallel form of the RS controller is introduced as a special case by discussing the absolute and deviating controller gain concepts. The section 3.2 brings up the RS-NGMV controller for LPV systems followed by an investigation of the control solution for unstructured non-linear input subsystem model in section 3.3. The section 3.4 introduces the weighting strategy and stability discussion. The section 3.5 presents the design procedure for the RS-NGMV controllers and section 3.6 summarizes the chapter and mentions its connections with the other chapters.

3.1 Restricted Structure Controller

The very basic idea behind the RS approach is to characterize a controller with a pre-defined order and structure chosen by the designer. The order and structure of the RS controller are independent of the plant order which provides important advantages. Generally, RS controllers are designed to be of a lower order than the plant taking the form of phase-lead, phase-lag, phase lead-lag or industrial PID controllers as described in detail by Chapter 12 of [42]. They enable flexibility by being low-order approximations to high-order controllers. In fact, a great deal of optimal/predictive control designs are of high order which in return creates difficulties on the implementation side. Hence comes the motivation to restrict such advanced control strategies to structures of lower order with simpler designs and wider range of applications. As a result, there has been research [$43-57$] to offer optimal/predictive control methods that comes with restricted structures. Among these studies, [54] considered a multi-variable Predictive PID (PPID) algorithm having gains minimizing the GPC cost index. In [44], the LQG cost index was used for the RS-optimizations in the tuning of control gains. The [47] and [56] involve benchmarking of the multi-variable Restricted Structure Linear Quadratic Gaussian (RS-LQG) controllers for determining the best RS structure by defining a controller performance index evaluated for each loop.

RS-NGMV control algorithm has been presented in [58]. The algorithm considers the NGMV method within a novel reduced-order restricted structure of a general z-transfer function or PID. The cost function to be minimized is very similar to the NGMV cost function. The optimization procedure results in the optimal feedback controller gains. The approach enables the advantages of the NGMV with a low-order structure such as the PID, that engineers in the industry who have classical controls training tend to be familiar with. Based on the results of this chapter, a RS-NGMV controller has been designed for a qLPV two-link robotic manipulator system [59]. The results verify the RS-NGMV algorithm and are further discussed in Chapter 5. The RS approach utilized in the RS-NGMV has also been adapted for the so-called Restricted Structure Generalized Predictive Control (RS-GPC) in [60]. The RS algorithms of [$58-60$] have all 1-DOF structures suited for reference tracking problem. The work in [61] provides the 2-DOF and 3-DOF versions of the RS-GPC to use the feedforward reference and measured output disturbance signals.

3.1.1 SISO Case

The RS controller is obtained from the multiplication of user pre-specified functions by some optimized feedback gains. It can be formulated as in:

$$\begin{aligned} u(t) &= \sum_{j=1}^{N_e} f_j(z^{-1}, k_j(t))e(t) \qquad (3.1)\\ &= f_1(z^{-1}, k_1(t))e(t) + f_2(z^{-1}, k_2(t))e(t) + \cdots + f_{N_e}(z^{-1}, k_{N_e}(t))e(t), \end{aligned}$$

where $f_j(z^{-1}, k_j(t))$ denotes the pre-specified functions, $k_j(t)$ represents the optimized feedback gains and $e(t) = r(t) - z(t)$ the feedback errors. The above expression generalizes the SISO case. The notation j stands for the index number assigned to the restricted structure elements.

The scalar restricted structure could be designed for any low-order controller of the general z-transfer function form,

$$C_0(z^{-1}) = \frac{C_{0,num} + C_{1,num}z^{-1} + \cdots + C_{n,num}z^{-n}}{1 + C_{1,den}z^{-1} + \cdots + C_{m,den}z^{-m}}.$$

To demonstrate how the controller is utilized more clearly, PID will be incorporated in the RS fashion in the example below. The Restricted Structure PID (RS-PID) controller has $N_e = 3$ pre-specified functions that can be broken down into the terms,

$$f_1(z^{-1}) = 1, f_2(z^{-1}) = \frac{1}{1 - z^{-1}}, f_3(z^{-1}) = \frac{1 - z^{-1}}{1 - \alpha z^{-1}},$$

which represents the discretized proportional, integral and the filtered derivative

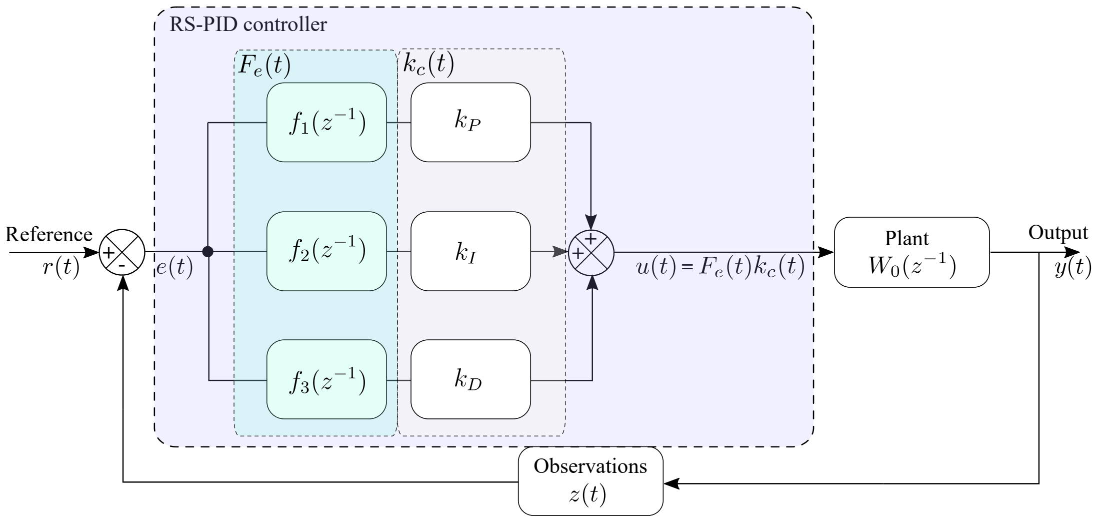


Figure 3.1: RS-PID controller.

operators respectively. Using the feedback gains k_{PID} and the feedback error $e(t)$ the RS-PID control signal is obtained,

$$u(t) = f_1(z^{-1})k_P e(t) + f_2(z^{-1})k_I T_s e(t) + f_3(z^{-1})k_D e(t), \tag{3.2}$$

which is illustrated in the Fig. 3.1. Beyond this point, it is up to the designer to decide how the PID gains are chosen and optimized. This thesis considers the NGMV optimization solution to tune the gains within the RS-PID structure (or RS-PI structure where the derivative term is not included). In an upcoming subsection, this is further explained under the parallel RS form where the designer can either chose to let RS optimizations to fully adjust the feedback gains $k_c(t)$ or define fixed gains first (for example PID) and then allow the RS optimizations to adjust these gains by deviations.

For software implementations, it is always useful to consider vectorizations. The scalar RS controller in (3.1) may be vectorized by,

$$u(t) = F_e(t)k_c(t). \tag{3.3}$$

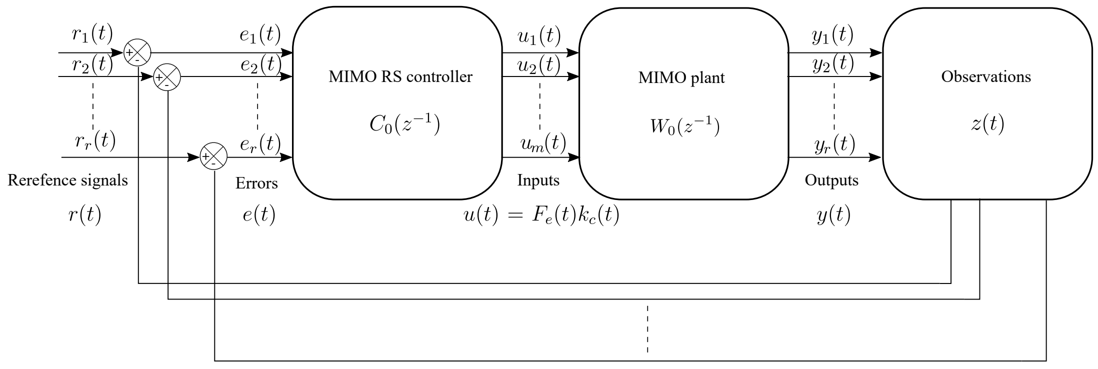


Figure 3.2: MIMO RS controller.

The term $F_e(t)$ is a vector of,

$$F_e(t) = \begin{pmatrix} e_{f1}(t) & e_{f2}(t) & \dots & e_{fN_e}(t) \end{pmatrix}_{1\times N_e}, \tag{3.4}$$

which are the products of the terms $f_j(z^{-1})$ extracted from the pre-specified functions and the feedback error $e(t)$, that is,

$$e_{fi}(t) = f_i(z^{-1})e(t), \tag{3.5}$$

for $i = \{1, 2, \dots, N_e\}$. The $k_c(t)$ is a $N_e \times 1$ size vector of gains,

$$k_c(t) = \begin{pmatrix} k_1 \\ k_2 \\ \vdots \\ k_{N_e} \end{pmatrix}_{N_e\times 1} \tag{3.6}$$

3.1.2 MIMO Case

It involves a little more tedious procedure to derive the RS controller for multi-variable systems. Consider a multi-variable system demonstrated in Fig. 3.2 with the dimension $m \times r$. We have r channels of error signals given by the vector below,

$$e(t) = \begin{pmatrix} e_1^T(t) & e_2^T(t) & \dots & e_r^T(t) \end{pmatrix}^T \tag{3.7}$$

The functions $f_j(z^{-1}, k_j(t))$ and gains $k_j(t)$ introduced in the SISO case now expands considering the size of the MIMO system, that is,

$$f_j(z^{-1}, k_j(t)) = \begin{pmatrix} f_{11}^j(z^{-1})k_{11}^j(t) & f_{12}^j(z^{-1})k_{12}^j(t) & \dots & f_{1r}^j(z^{-1})k_{1r}^j(t) \\ f_{21}^j(z^{-1})k_{21}^j(t) & f_{22}^j(z^{-1})k_{22}^j(t) & \dots & f_{2r}^j(z^{-1})k_{2r}^j(t) \\ \vdots & \vdots & \ddots & \vdots \\ f_{m1}^j(z^{-1})k_{m1}^j(t) & f_{m2}^j(z^{-1})k_{m2}^j(t) & \dots & f_{mr}^j(z^{-1})k_{mr}^j(t) \end{pmatrix}. \tag{3.8}$$

Multiplying the error signals in (3.7) with (3.8), the $f_j(z^{-1}, k_j(t))e(t)$ equals to,

$$\begin{pmatrix} f_{11}^j(z^{-1})k_{11}^j(t)e_1(t) + f_{12}^j(z^{-1})k_{12}^j(t)e_2(t) + \dots + f_{1r}^j(z^{-1})k_{1r}^j(t)e_r(t) \\ f_{21}^j(z^{-1})k_{21}^j(t)e_1(t) + f_{22}^j(z^{-1})k_{22}^j(t)e_2(t) + \dots + f_{2r}^j(z^{-1})k_{2r}^j(t)e_r(t) \\ \vdots \quad\quad \vdots \quad\quad \vdots \quad\quad \vdots \\ f_{m1}^j(z^{-1})k_{m1}^j(t)e_1(t) + f_{m2}^j(z^{-1})k_{m2}^j(t)e_2(t) + \dots + f_{mr}^j(z^{-1})k_{mr}^j(t)e_r(t) \end{pmatrix}. \tag{3.9}$$

The controller form in equation (3.1) is summarized into its MIMO form,

$$u(t) = \begin{pmatrix} \sum_{j=1}^{N_e} \sum_{l=1}^{r} \{f_{11}^j(z^{-1})k_{1l}^j e_l(t)\} \\ \sum_{j=1}^{N_e} \sum_{l=1}^{r} \{f_{21}^j(z^{-1})k_{2l}^j e_l(t)\} \\ \vdots \\ \sum_{j=1}^{N_e} \sum_{l=1}^{r} \{f_{m1}^j(z^{-1})k_{ml}^j e_l(t)\} \end{pmatrix}_{m\times 1}, \tag{3.10}$$

It is possible to parametrize the MIMO RS controller as $u(t) = F_e(t)k_c(t)$ too, by introducing the pre-specified functions and gains in the matrix form.

While it is possible to get away without parametrisation in the SISO case, the approach is undeniably advantageous when dealing with large multi-variable systems because it allows the software implementation of the RS controller to be much less complex. Firstly, the user pre-specified functions from (3.4) are expanded starting from the first row as in,

$$\begin{pmatrix} f_e^{11} \\ f_e^{12} \\ \vdots \\ f_e^{1r} \end{pmatrix} = \begin{pmatrix} f_{11}^{1}(z^{-1})e_1(t) & f_{11}^{2}(z^{-1})e_1(t) & \dots & f_{11}^{N_e}(z^{-1})e_1(t) \\ f_{12}^{1}(z^{-1})e_2(t) & f_{12}^{2}(z^{-1})e_2(t) & \dots & f_{12}^{N_e}(z^{-1})e_2(t) \\ \vdots & \vdots & \ddots & \vdots \\ f_{1r}^{1}(z^{-1})e_r(t) & f_{1r}^{2}(z^{-1})e_r(t) & \dots & f_{1r}^{N_e}(z^{-1})e_r(t) \end{pmatrix}_{r\times N_e}, \tag{3.11}$$

and then the second row expands,

$$\begin{pmatrix} f_e^{21} \\ f_e^{22} \\ \vdots \\ f_e^{2r} \end{pmatrix} = \begin{pmatrix} f_{21}^{1}(z^{-1})e_1(t) & f_{21}^{2}(z^{-1})e_1(t) & \dots & f_{21}^{N_e}(z^{-1})e_1(t) \\ f_{22}^{1}(z^{-1})e_2(t) & f_{22}^{2}(z^{-1})e_2(t) & \dots & f_{22}^{N_e}(z^{-1})e_2(t) \\ \vdots & \vdots & \ddots & \vdots \\ f_{2r}^{1}(z^{-1})e_r(t) & f_{2r}^{2}(z^{-1})e_r(t) & \dots & f_{2r}^{N_e}(z^{-1})e_r(t) \end{pmatrix}_{r\times N_e}, \tag{3.12}$$

and rest of the rows expand similarly, until the *mth* row,

$$\begin{pmatrix} f_e^{m1} \\ f_e^{m2} \\ \vdots \\ f_e^{mr} \end{pmatrix} = \begin{pmatrix} f_{m1}^{1}(z^{-1})e_1(t) & f_{m1}^{2}(z^{-1})e_1(t) & \dots & f_{m1}^{N_e}(z^{-1})e_1(t) \\ f_{m2}^{1}(z^{-1})e_2(t) & f_{m2}^{2}(z^{-1})e_2(t) & \dots & f_{m2}^{N_e}(z^{-1})e_2(t) \\ \vdots & \vdots & \ddots & \vdots \\ f_{mr}^{1}(z^{-1})e_r(t) & f_{mr}^{2}(z^{-1})e_r(t) & \dots & f_{mr}^{N_e}(z^{-1})e_r(t) \end{pmatrix}_{r\times N_e}. \tag{3.13}$$

These terms are gathered within the matrix below,

$$\begin{pmatrix} e_{f1}(t) \\ e_{f2}(t) \\ \vdots \\ e_{fm}(t) \end{pmatrix} = \begin{pmatrix} f_e^{11} & f_e^{12} & \dots & f_e^{1r} \\ f_e^{21} & f_e^{22} & \dots & f_e^{2r} \\ \vdots & \vdots & \ddots & \vdots \\ f_e^{m1} & f_e^{m2} & \dots & f_e^{mr} \end{pmatrix}_{m\times r}, \tag{3.14}$$

which is used to define the diagonal matrix $F_e(t)$ that contain the pre-specified functions (size $m \times (m \times r \times N_e)$ matrix when fully expanded),

$$F_e(t) = diag\{e_{f1}(t), e_{f2}(t), \cdots, e_{fm}(t)\}. \tag{3.15}$$

Now, the similar approach is taken for parametrising the feedback gains. First, define the $(m \times r \times N_e) \times 1$ size total gain vector $k_c(t)$,

$$k_c(t) = \begin{pmatrix} k_{c1}^T(t) & k_{c2}^T(t) & \cdots & k_{cm}^T(t) \end{pmatrix}^T, \tag{3.16}$$

where feedback gains belonging to each multi-variable loop are embedded in channels indexed by $i = \{1, 2, \ldots, m\}$,

$$k_{ci}(t) = \begin{pmatrix} k_c^{i1}(t) & k_c^{i2}(t) & \ldots & k_c^{ir}(t) \end{pmatrix}. \tag{3.17}$$

The individual elements of the first channel is expanded with the fashion,

$$\begin{aligned} k_{c1}(t) &= \begin{pmatrix} k_c^{11}(t) & k_c^{12}(t) & \ldots & k_c^{1r}(t) \end{pmatrix} \\ &= \begin{pmatrix} k_{11}^1(t) & k_{12}^1(t) & \ldots & k_{1r}^1(t) \\ k_{11}^2(t) & k_{12}^2(t) & \ldots & k_{1r}^2(t) \\ \vdots & \vdots & \ddots & \vdots \\ k_{11}^{N_e} & k_{12}^{N_e}(t) & \ldots & k_{1r}^{N_e}(t) \end{pmatrix}_{r \times N_e}, \end{aligned} \tag{3.18}$$

The second channel expands similarly,

$$\begin{aligned} k_{c2}(t) &= \begin{pmatrix} k_c^{21}(t) & k_c^{22}(t) & \ldots & k_c^{2r}(t) \end{pmatrix} \\ &= \begin{pmatrix} k_{21}^1(t) & k_{22}^1(t) & \ldots & k_{2r}^1(t) \\ k_{21}^2(t) & k_{22}^2(t) & \ldots & k_{2r}^2(t) \\ \vdots & \vdots & \ddots & \vdots \\ k_{21}^{N_e} & k_{22}^{N_e}(t) & \ldots & k_{2r}^{N_e}(t) \end{pmatrix}_{r \times N_e}, \end{aligned} \tag{3.19}$$

and so on until the *mth* channel,

$$k_{cm}(t) = \begin{pmatrix} k_c^{m1}(t) & k_c^{m2}(t) & \dots & k_c^{mr}(t) \end{pmatrix} = \begin{pmatrix} k_{m1}^1(t) & k_{m2}^1(t) & \dots & k_{mr}^1(t) \\ k_{m1}^2(t) & k_{m2}^2(t) & \dots & k_{mr}^2(t) \\ \vdots & \vdots & \ddots & \vdots \\ k_{m1}^{N_e} & k_{m2}^{N_e}(t) & \dots & k_{mr}^{N_e}(t) \end{pmatrix}_{r\times N_e}, \tag{3.20}$$

Finally, as a combination of the parametrised function terms and the gains, the RS control input for the multi-variable system is calculated,

$$u(t) = F_e(t)k_c(t) = \begin{pmatrix} e_{f1}(t)k_{c1}^T(t) \\ e_{f2}(t)k_{c2}^T(t) \\ \vdots \\ e_{fm}(t)k_{cm}^T(t) \end{pmatrix}_{m\times 1}. \tag{3.21}$$

3.1.3 Parallel Form of the Controller

Parallel form allows the expression of two special cases at the same time, by splitting the feedback gains of the RS controller.

The feedback gain term $k_c(t)$ may be expressed as the combination of a constant component $\overline{k}_c$ and a time-varying component $\widetilde{k}_c(t)$. In this case, the RS control input may be re-formulated by,

$$\begin{aligned} u(t) &= F_e(t)k_c(t) = F_e(t)\overline{k}_c + F_e(t)\widetilde{k}_c(t) \\ &= \sum_{j=1}^{N_e} f_j(z^{-1})\overline{k}_j e(t) + \sum_{j=1}^{N_e} f_j(z^{-1})\widetilde{k}_j(t)e(t), \end{aligned} \tag{3.22}$$

that is if $\overline{k}_c = 0$, it refers to the absolute gain case and $k_c(t) = \widetilde{k}_c(t)$. If $\overline{k}_c \neq 0$, then $k_c(t) = \overline{k}_c + \widetilde{k}_c(t)$ and this refers to the gain deviation case. To explain the concept of parallel form easier, Fig. 3.3 can be referred to. The gain deviation

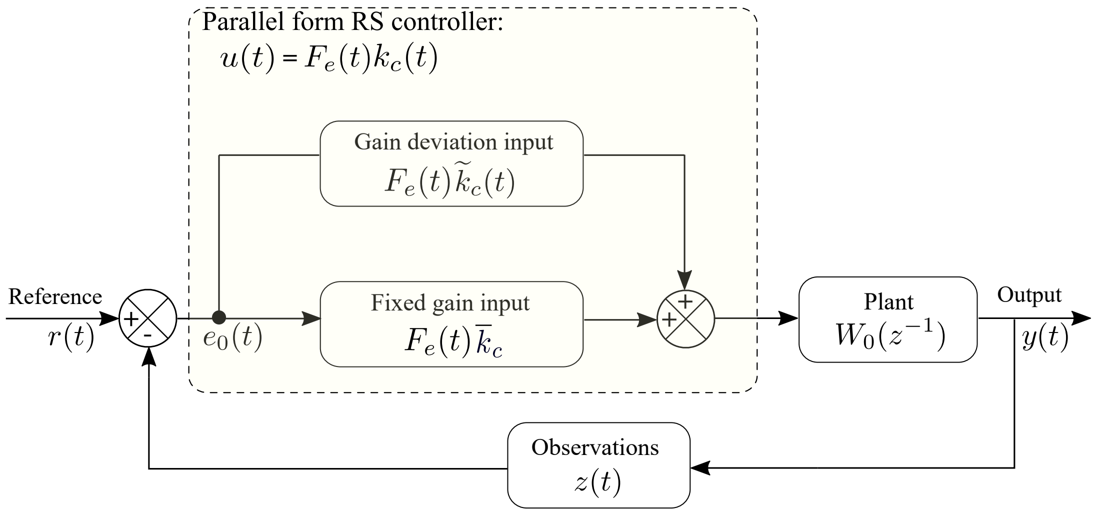


Figure 3.3: Parallel form of the RS controller.

case is similar to having two RS controllers in parallel with one having constant gains, the other time-varying gains that are combined for the final feedback gain.

Splitting the controller gains can be very practical if a fixed gain controller is already available (like PID for example). Then, the task is to compute the deviations from the constant gains through optimizations. This can be especially useful when considering processes with limited computing power (like small microcontrollers). In that case, the controller only computes the gain deviations component unlike absolute gain case where it deals with the minimization of the entire controller gains.

From the stability point of view, if used for fixed gains $\overline{k}_c$, the PID controller that is chosen must be stabilizing the system, already. The gain deviations $\widetilde{k}_c(t)$ that are calculated from the optimizations and will be combined with PID gains, determine the stability thus the RS-NGMV must be weighted properly. However, this does not guarantee stability. The weighting and stability concepts are discussed in 3.4 in more detail.

3.2 RS-NGMV Solution for LPV Systems

RS-NGMV control algorithm takes the same state-space modelling approach with the NGMV introduced in the section 2.5 of the second chapter. It is designed for the state-space system (represented by $(2.19-2.22)$ and (2.25)) with subsystems depicted in Fig. 2.5. The method also uses the prediction models (2.55) and (2.59) for its LPVKF (predictor-corrector type). Therefore, it is avoided to replicate the system and LPVKF definitions here. The objective of this section is to present how the NGMV optimization is utilized to calculate the feedback gains $k_c(t)$ of the RS controller $u(t) = F_e(t)k_c(t)$, thus becoming the so-called RS-NGMV controller.

Compared to the traditional versions, the optimization procedure recruits some additional weightings that need to be provided prior to the analysis,

- Constant weighting placed on the tracking error's dynamic weighting: $\Lambda_p^2 = diag\left(\lambda_{p1}^2,\ \ \lambda_{p2}^2,\ \ \cdots,\ \ \lambda_{pn_p}^2\right).$
- Constant weighting on the control signal u_0: $\Lambda_u^2 = diag\left(\lambda_{u1}^2,\ \ \lambda_{u2}^2,\ \ \cdots,\ \ \lambda_{un}^2\right).$
- Constant weighting to penalize the deviations in controller gains: $\Lambda_k^2 = diag\left(\lambda_{k1}^2,\ \ \lambda_{k2}^2,\ \ \cdots,\ \ \lambda_{kn}^2\right).$
- Constant weighting on increments on the gain deviations: $\Lambda_d^2 = diag\left(\lambda_{d1}^2,\ \ \lambda_{d2}^2,\ \ \cdots,\ \ \lambda_{dn}^2\right).$

These terms help tune the controller with higher precision than before. For example, depending on the control problem, large gain deviations might be undesired, the rate of change of controller gains might be too fast or too slow.

The optimization analysis in this section considers the plant to be purely of the LPV form thus the unstructured subsystem (NL input subsystem in Fig. 2.5) needs to be by-passed for now. In other words, it is set to $W_{1k} = I$ and the control signal $u_0(t) = u(t)$. The results will be further generalized for the unstructured subsystem in the optimization analysis of the section 3.3.

Define the RS-NGMV cost function J,

$$J = E\{{e_p}^T(t+k)\Lambda_p^2 e_p(t+k) + {u_0}^T(t)\Lambda_u^2 u_0(t) + \widetilde{k}_c^T(t)\Lambda_k^2\widetilde{k}_c(t) + \Delta\widetilde{k}_c^T(t)\Lambda_d^2\Delta\widetilde{k}_c(t)|t\}, \quad (3.23)$$

where $\Delta\widetilde{k}_c(t)$ represents the incremental gain change,

$$\Delta\widetilde{k}_c(t) = \widetilde{k}_c(t) - \widetilde{k}_c(t-1) = k_c(t) - k_c(t-1), \quad (3.24)$$

Then with the help of LPV Kalman filter estimations, it follows from (3.23) that,

$$\begin{aligned} J = E\{(\hat{e}_p(t+k) + \widetilde{e}_p(t+k))^T\Lambda_p^2(\hat{e}_p(t+k) + \widetilde{e}_p(t+k)) + {u_0}^T(t)\Lambda_u^2 u_0(t) + \widetilde{k}_c^T(t)\Lambda_k^2\widetilde{k}_c(t) \\ + \Delta\widetilde{k}_c^T(t)\Lambda_d^2\Delta\widetilde{k}_c(t)|t\}, \end{aligned}$$

which can be simplified further as in below due to the fact that state estimates and estimation errors are orthogonal and using (2.61) from chapter 2,

$$J = {e_p}^T(t+k)\Lambda_p^2 e_p(t+k) + {u_0}^T(t)\Lambda_u^2 u_0(t) + \widetilde{k}_c^T(t)\Lambda_k^2\widetilde{k}_c(t) + \Delta\widetilde{k}_c^T(t)\Lambda_d^2\Delta\widetilde{k}_c(t) + J_0(t),$$

where $J_0(t) = E\{\widetilde{e}_p^T(t+k)\Lambda_p^2\widetilde{e}_p(t+k)|t\}$. Using (2.60) from chapter 2 as well, the following is obtained,

$$\begin{aligned} J = (d_{pd}^0(t+k) + E_{pt+k}u_0(t))^T\Lambda_p^2(d_{pd}^0(t+k) + E_{pt+k}u_0(t)) + {u_0}^T(t)\Lambda_u^2 u_0(t) + \widetilde{k}_c^T(t)\Lambda_k^2\widetilde{k}_c(t) \\ + \Delta\widetilde{k}_c^T(t)\Lambda_d^2\Delta\widetilde{k}_c(t) + J_0(t), \end{aligned}$$

expanding the terms above, it becomes,

$$\begin{aligned} J = d_{pd}^0(t+k)\Lambda_p^2 d_{pd}^0(t+k) + d_{pd}^0(t+k)\Lambda_p^2 E_{pt+k}u_0(t) + {u_0}^T(t){E_{pt+k}}^T\Lambda_p^2 d_{pd}^0(t+k) \\ + {u_0}^T(t)({E_{pt+k}}^T\Lambda_p^2 E_{pt+k} + \Lambda_u^2)u_0(t) + \widetilde{k}_c^T(t)\Lambda_k^2\widetilde{k}_c(t) + \Delta\widetilde{k}_c^T(t)\Lambda_d^2\Delta\widetilde{k}_c(t) + J_0(t), \end{aligned}$$

recall that the gain $k_c(t)$ can be split into constant and deviating terms thus there is the relation $\widetilde{k}_c(t) = k_c(t) - \overline{k}_c$, and using (3.24) in addition to substitute the term $\Delta\widetilde{k}_c(t)$,

$$\begin{aligned}J &= d^0_{pd}(t+k)\Lambda_p^2 d^0_{pd}(t+k) + {d^0_{pd}}^T(t+k)\Lambda_p^2 E_{pt+k}u_0(t) + u_0(t)^T {E_{pt+k}}^T\Lambda_p^2 d^0_{pd}(t+k)\\ &+ u_0(t)^T({E_{pt+k}}^T\Lambda_p^2 E_{pt+k} + \Lambda_u^2)u_0(t) + (k_c(t) - \overline{k}_c)^T\Lambda_k^2(k_c(t) - \overline{k}_c) + J_0(t)\\ &+ (k_c(t) - k_c(t-1))^T\Lambda_d^2(k_c(t) - k_c(t-1)).\end{aligned}$$

The controller has the restricted structure defined as $u(t) = F_e(t)k_c(t)$ previously, and it was also stated that $u(t) = u_0(t)$ since the unstructured plant model was by-passed. Thus, expanding some more terms and substituting the $u_0(t)$ as in,

$$\begin{aligned}J &= {d^0_{pd}}^T(t+k)\Lambda_p^2 d^0_{pd}(t+k) + {d^0_{pd}}^T(t+k)\Lambda_p^2 E_{pt+k}F_e(t)k_c(t) + {k_c}^T(t){F_e}^T(t){E_{pt+k}}^T\Lambda_p^2 d^0_{pd}(t+k)\\ &+ {k_c}^T(t)({F_e}^T(t)({E_{pt+k}}^T\Lambda_p^2 E_{pt+k} + \Lambda_u^2)F_e(t) + \Lambda_k^2)k_c(t) - {k_c}^T(t)\Lambda_k^2\overline{k}_c - {\overline{k}_c}^T\Lambda_k^2 k_c(t) + J_0(t)\\ &+ {\overline{k}_c}^T\Lambda_k^2\overline{k}_c + {k_c}^T(t)\Lambda_d^2 k_c(t) - {k_c}^T(t)\Lambda_d^2 k_c(t-1) - {k_c}^T(t-1)\Lambda_d^2 k_c(t) + {k_c}^T(t-1)\Lambda_d^2 k_c(t-1).\end{aligned}$$

The expression above is crowded but might become more compact by defining the following matrices,

$$X_0(t) = {F_e}^T(t)({E_{pt+k}}^T\Lambda_p^2 E_{pt+k} + \Lambda_u^2)F_e(t) + \Lambda_k^2 + \Lambda_d^2, \tag{3.25}$$

$$P_p(t) = {F_e}^T(t){E_{pt+k}}^T\Lambda_p^2, \tag{3.26}$$

$$\psi_k(t) = -\Lambda_k^2\overline{k}_c - \Lambda_d^2 k_c(t-1), \tag{3.27}$$

$$\overline{J_0}(t) = ({\overline{k}_c}^T\Lambda_k^2\overline{k}_c + {k_c}^T(t-1)\Lambda_d^2 k_c(t-1) + J_0(t). \tag{3.28}$$

Substituting each term above the cost function J returns in the compact form,

$$\begin{aligned}J &= {d^0_{pd}}^T(t+k)\Lambda_p^2 d^0_{pd}(t+k) + ({\psi_k}^T(t) + {d^0_{pd}}^T(t+k){P_p}^T(t))k_c(t)\\ &+ {k_c}^T(P_p(t)d^0_{pd}(t+k) + \psi_k(t)) + {k_c}^T(t)X_0(t)k_c(t) + \overline{J_0}(t).\end{aligned}$$

The cost function J is minimized by calculating its gradient $\partial J/\partial k_c(t) = 0$ to obtain the optimal value of $k_c(t)$. Note that $\overline{J_0}(t)$ is independent of $k_c(t)$ and

$X_0(t)$ is symmetric. Taking the gradient of J and getting rid of the noise terms independent of control action results in the statement below,

$$({\psi_k}^T(t) + {d_{pd}^0}^T(t+k){P_p}^T(t)) + (P_p(t)d_{pd}^0(t+k) + \psi_k(t)) + 2X_0(t)k_c(t) \; = 0,$$

Since the first two terms are equal, the equation becomes,

$$2(P_p(t)d_{pd}^0(t+k) + \psi_k(t)) + 2X_0(t)k_c(t) = 0, \tag{3.29}$$

assuming that $X_0(t)$ is also invertible, the solution for the optimal gains is,

$$k_c(t) = -X_0(t)^{-1}(P_p(t)d_{pd}^0(t+k)) + \psi_k(t)). \tag{3.30}$$

The solution depends on the inverse of $X_0(t)$ clearly. This matrix is guaranteed to be full rank and symmetric with the choice of weightings $\Lambda_k^2 > 0$ and $\Lambda_d^2 \geq 0$. Recalling $d_{pd}^0(t+k) = d_{pd}(t+k) + C_{pt+k}\hat{x}(t+k|t)$ and re-arranging gains to be more suitable for software implementation,

$$k_c(t) = -X_0(t)^{-1}(P_p(t)(d_{pd}(t+k) + C_{pt+k}\hat{x}(t+k|t)) + \psi_k(t)). \tag{3.31}$$

The optimal gains $k_c(t)$ is continuously updated until the optimizations becoming constant in the steady-state.

The RS-NGMV controller derived, has the generalized 1-DOF feedback controller structure as shown in Fig. 3.4. Gains $k_c(t)$ are computed in the background using LPVKF observations and RS-NGMV optimizations. The feedback calculates the tracking errors which are utilized in the user specified functions $F_e(t)$. Gains and functions are updated in real-time and return the control signal $u(t)$ that is applied on the LPV plant with delay ($W_0 = W_{0k}z^{-k}$).

The RS-NGMV implementation may also encounter algebraic loop problem.

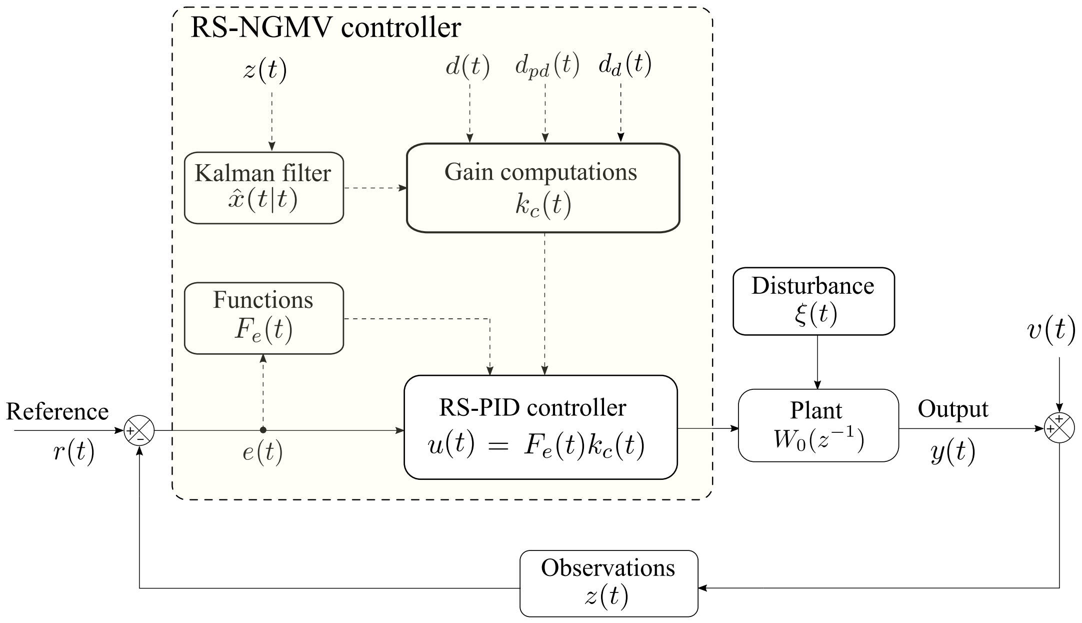


Figure 3.4: RS-NGMV controller implementation.

In the cases studies in this thesis, the problem was overcome easily by using a unit-step delay block.

3.3 RS-NGMV Solution with NL Subsystem

The results follow from the previous section's optimal RS-NGMV control law. This time the plant involves a non-linear input subsystem (the unstructured subsystem) thus, $u_0(t) = W_{1k}u(t)$ as was shown in Fig. 2.5. The control weighting operator F_{ck} is assumed to be of full rank (invertible) and is allowed to be non-linear ($F_c u(t) = z^{-k}F_{ck}u(t)$).

The optimization problem requires the minimization of a new cost function that is the variance of the signal $\phi_p(t+k)$,

$$J = E\{{\phi_p}^T(t+k)\phi_p(t+k)\}, \tag{3.32}$$

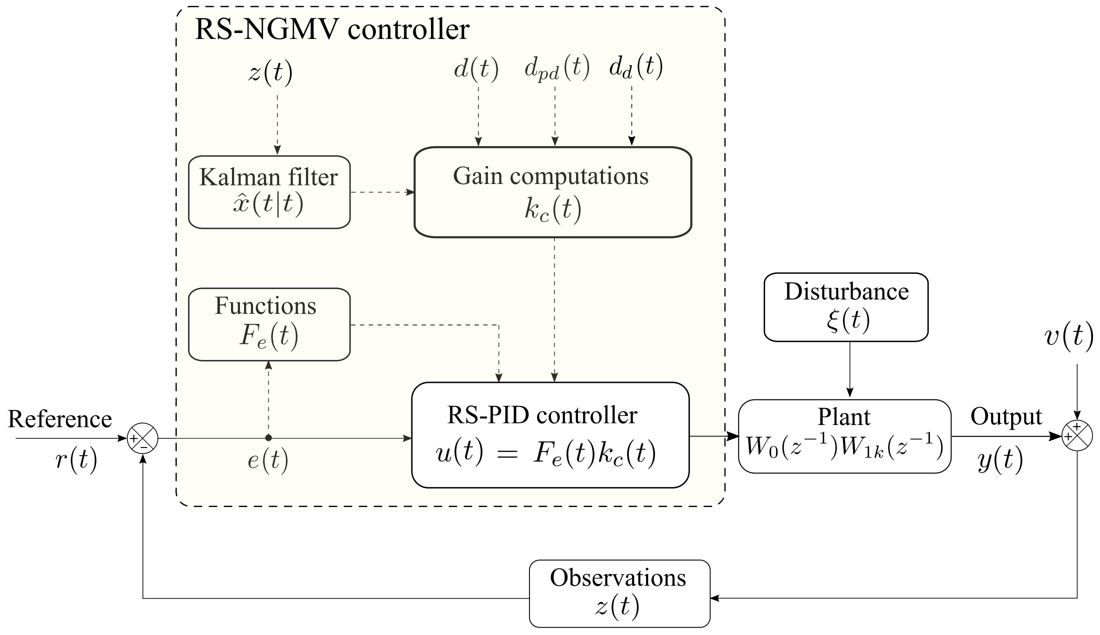

RS-NGMV controller
$z(t)$
$d(t)$ $d_{pd}(t)$ $d_d(t)$
Kalman filter $\hat{x}(t|t)$
Gain computations $k_c(t)$
Functions $F_e(t)$
Disturbance $\xi(t)$
$v(t)$
Reference $r(t)$
$e(t)$
RS-PID controller $u(t) = F_e(t)k_c(t)$
Plant $W_0(z^{-1})W_{1k}(z^{-1})$
Output $y(t)$
Observations $z(t)$

Figure 3.5: RS-NGMV controller implementation for plants with NL subsystem.

which is a re-arranged version of the pseudo-output signal presented earlier,

$$\phi_p(t+k) = P_p(t)e_p(t+k) + F_{c0}u_0(t) + F_{c1}\widetilde{k}_c(t) + F_{c2}\Delta\widetilde{k}_c(t) + {F_e}^T(t)F_{ck}u(t). \tag{3.33}$$

The signal $\phi_p(t+k)$ is composed of control and error weightings that are re-defined, $P_p(t) = {F_e}^T(t)E^T_{pt+k}{\Lambda_p}^2, F_{c0} = {F_e}^T(t){\Lambda_u}^2, F_{c1} = {\Lambda_k}^2, F_{c2} = {\Lambda_d}^2$, and F_{ck} in addition.

Using (2.61) from chapter 2 on (3.33), the signal $\phi_p(t+k)$ becomes,

$$\begin{aligned}\phi_p(t+k) = {} & P_p(t)\hat{e}_p(t+k) + F_{c0}u_0(t) + F_{c1}\widetilde{k}_c(t) + F_{c2}\Delta\widetilde{k}_c(t) + P_p(t)\widetilde{e}_p(t+k) \\ & + {F_e}^T(t)F_{ck}u(t).\end{aligned}$$

Similar to the weighted error $e_p(t+k)$, the signal $\phi_p(t+k)$ is also separated into prediction and estimation error terms, $\phi_p(t+k) = \hat{\phi}_p(t+k) + \widetilde{\phi}_p(t+k)$ where,

- $\hat{\phi}_p(t+k) = P_p(t)\hat{e}_p(t+k) + F_{c0}u_0(t) + F_{c1}\widetilde{k}_c(t) + F_{c2}\Delta\widetilde{k}_c(t) + {F_e}^T(t)F_{ck}u(t)$,
- $\widetilde{\phi}_p(t+k) = P_p(t)\widetilde{e}_p(t+k)$.

Re-writing the cost function (3.32) after the separation of $\phi_p(t+k)$,

$$\widetilde{J} = E\{(\hat{\phi}_p(t+k) + \widetilde{\phi}_p(t+k))^T(\hat{\phi}_p(t+k) + \widetilde{\phi}_p(t+k))|t\}, \tag{3.34}$$

which can be simplified due to the fact that estimation error and estimation terms are orthogonal,

$$\widetilde{J} = E\{\hat{\phi}_p^T(t+k)\hat{\phi}_p(t+k) + \widetilde{\phi}_p^T(t+k)\widetilde{\phi}_p(t+k)|t\}, \tag{3.35}$$

The equation is further simplified by,

$$\widetilde{J} = \hat{\phi}_p^T(t+k)\hat{\phi}_p(t+k) + \widetilde{J}_1. \tag{3.36}$$

where $\widetilde{J}_1$ is defined as $\widetilde{J} = E\{\widetilde{\phi}_p^T(t+k)\widetilde{\phi}_p(t+k)|t\}$.

Substituting $\hat{e}_p(t+k)$ from (2.61) in $\hat{\phi}_p(t+k)$, it gives,

$$\begin{aligned}\hat{\phi}_p(t+k) = {} & P_p(t)d^0_{pd}(t+k) + P_p(t)E_{pt+k}u_0(t) + F_{c0}u_0(t) + F_{c1}\widetilde{k}_c(t) + F_{c2}\Delta\widetilde{k}_c(t)\\ & + {F_e}^TF_{ck}u(t).\end{aligned}$$

Then substituting the $P_p(t)$ term associated with the control input $u_0(t)$, the terms $\widetilde{k}_c(t)$,$\Delta\widetilde{k}_c(t)$, F_{c0}, F_{c1} and F_{c2},

$$\begin{aligned}\hat{\phi}_p(t+k) = {} & P_p(t)d^0_{pd}(t+k) + {F_e}^T(t)E^T_{pt+k}{\Lambda_p}^2E_{pt+k}u_0(t) + {F_e}^T(t){\Lambda_u}^2u_0(t) + {\Lambda_p}^2(k_c(t) - \overline{k}_c)\\ & + {\Lambda_d}^2(k_c(t) - k_c(t-1)) + {F_e}^TF_{ck}u(t).\end{aligned}$$

Since $u_0(t) = (W_{1k}u)(t)$ and $u(t) = F_e(t)k_c(t)$, the signal can be re-arranged as,

$$\begin{aligned}\hat{\phi}_p(t+k) = {} & P_p(t)d^0_{pd}(t+k) + {\Lambda_k}^2k_c(t) + {\Lambda_d}^2k_c(t) + {F_e}^T(t)(E^T_{pt+k}{\Lambda_p}^2E_{pt+k} + {\Lambda_u}^2)W_{1k}F_e(t)k_c(t)\\ & + {F_e}^T(t)F_{ck}F_e(t)k_c(t) - {\Lambda_k}^2\overline{k}_c(t) - {\Lambda_d}^2(k_c(t-1).\end{aligned}$$

Then substituting the term $\phi_k(t)$ from (3.27), the term $\hat{\phi}_p(t+k)$ above is updated as in,

$$\hat{\phi}_p(t+k) = P_p(t)d^0_{pd}(t+k) + {\Lambda_k}^2 k_c(t) + {\Lambda_d}^2 k_c(t) + {F_e}^T(t)(E^T_{pt+k}{\Lambda_p}^2 E_{pt+k} + {\Lambda_u}^2)W_{1k}F_e(t)k_c(t) + \\ + {F_e}^T(t)F_{ck}F_e(t)k_c(t) - \psi_k(t).$$

The solution for the optimal gain $k_c(t)$ results from the minimization of the cost function $\widetilde{J}$ in (3.36). Due to the cost-function term $\widetilde{J}_1$ being independent of the control action, the condition for optimality becomes $\hat{\phi}_p(t+k) = 0$ (recall that for minimum variance controllers the variance term must be minimized). Therefore, setting $\hat{\phi}_p(t+k)$ of above to zero and solving for $k_c(t)$ yields the definition below,

$$k_c(t) = ({\Lambda_k}^2 + {\Lambda_d}^2 + {F_e}^T(t)F_{ck}F_e(t))^{-1}(-P_p(t)d^0_{pd}(t+k) - \psi_k(t) \\ - {F_e}^T(t)(E^T_{pt+k}{\Lambda_p}^2 E_{pt+k} + {\Lambda_u}^2)W_{1k}F_e(t)k_c(t)).$$

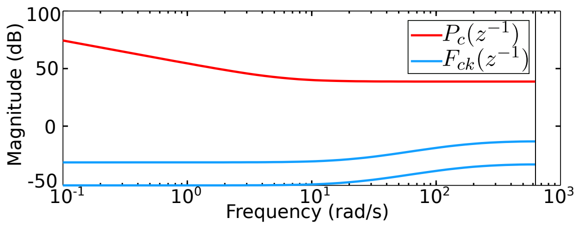
For implementations the solution is better organized as,

$$k_c(t) = -({\Lambda_k}^2 + {\Lambda_d}^2)^{-1}(P_p(t)(d_{pd}(t+k) + C_{pt+k}\hat{x}(t+k|t)) + \psi_k(t) \\ + {F_e}^T(t)(F_{ck} + E^T_{pt+k}{\Lambda_p}^2 E_{pt+k} + {\Lambda_u}^2)W_{1k}F_e(t)k_c(t)).$$

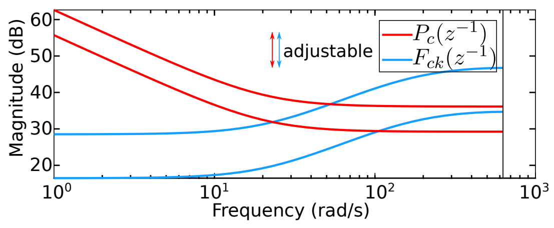
Then the optimal RS-NGMV controller can be implemented as in Fig. 3.5. The feedback control structure is similar to Fig. 3.4 except that the plant model is now W_0W_{1k} as the case involves the non-linear input subsystem. It is important to mention that there is a corollary that follows the new result, that is, if $F_{ck} \to 0$ & $W_{1k} = I$ then, the condition for optimality becomes $\hat{\phi}_p(t+k) = 0$,

$$\hat{\phi}_p(t+k) = P_p(t)d^0_{pd}(t+k) + {\Lambda_k}^2 k_c(t) + {\Lambda_d}^2 k_c(t) + {F_e}^T(t)(E^T_{pt+k}{\Lambda_p}^2 E_{pt+k} + {\Lambda_u}^2)F_e(t)k_c(t) \\ - \psi_k(t) = 0,$$

the solution of which is non-other than (3.31), the optimal gain $k_c(t)$ from the initial analysis in section 3.3.



(a) Selected weights



(b) Selected weights re-adjusted

Figure 3.6: Control and error terms weighting.

3.4 Weighting Strategy and Stability Discussion

To describe the notion of weighting, first consider the frequency responses for two different sets of dynamic control and error weights as displayed in Fig. 3.6. Part (a) results from the weightings chosen for the robotic manipulator case study simulations of the chapter 5. Part (b) demonstrates the response when the weights in (a) are re-adjusted only to highlight their frequency characteristics. The control weight $F_{ck}(z^{-1})$ has high-pass dynamics and is designed as a lead-compensator term to make sure the controller rolls off at high frequencies. The error weight $P_c(z^{-1})$ demonstrates low-pass filter characteristics and is in the lag-compensator form for the realistic roll-off of error dynamics. Both weightings could be adjusted to work within desired frequency ranges depending on the type of applications. For instance, the intersection between the control and error weights could be used to define the bandwidth of the system.

To consider the stability of the RS-NGMV solution first recall that the NGMV full order controller can ensure stability if the weightings are chosen based on an existing stabilizing controller and the following assumptions are made:

- $F_{ck}(z^{-1})$ must be full rank and invertible,
- The non-linear subsystem W_{1k} is finite gain stable (has stable inverse),
- The linear subsystem W_0 is allowed to contain unstable models,

- The operator $(P_cW_k - F_{ck})$ has a finite gain stable causal inverse.

With a suitable choice of weightings it can be ensured that the $(P_cW_k - F_{ck})$ is minimum-phase and the closed-loop stability is achievable.

In more detail, consider a linear and negative $F_{ck} = -F_k$. Then the operator can be expressed by $F_k(F_k^{-1}P_cW_k + I)$. The term $(I + (F_k^{-1}P_cW_k)$ is of the form $(1 + KGH)$ known as the return-difference operator [62] where the terms K,G and H represent the control, plant and feedback models. The return-difference operator is related to the closed-loop stability of feedback control systems. There is a link between the operators as the term $(I + (F_k^{-1}P_cW_k)$ can be used to represent a system with the plant model W_k and the feedback controller $K_c = F_k^{-1}P_c$. The connection thus hints a strategy for cost-function weighting selection. As an example, suppose a PID controller K_c which already stabilizes the closed-loop system exists. In this case, $K_c = F_k^{-1}P_c$ can inspire the initial choice of control and error weightings. Assume that $F_k = -I$, then the $P_c = -K_c$ which will be minimum phase if the PID parameters are non-negative.

The example approach is referred to as the PID motivated NGMV weighting technique [63]. It can be used as a simple and effective starting point to design the controller.

Now, going back to the RS-NGMV case the stability can be established in a special case that is nevertheless useful. The argument is as follows:

- Recall that the NGMV weightings can be selected equal to the stabilizing classical controller to guarantee stability,
- Assume a full-order observer (Kalman filter) is used in the RS-NGMV state-space controller,
- If the two cases above apply, then the optimal RS-NGMV solution must be the same as for the full order case (i.e. reduced order RS not used) that can be guaranteed stable.

Clearly, moving away from this case by using a progressively lower order observer will result in sub-optimality and increase cost and reduce stability margins even leading to instability. However, this is a problem when any parametrised controller is used where the coefficients are found by a meta-heuristic or global optimization method.

3.5 Design Procedure

A suggested design procedure for the RS-NGMV control algorithm blended with software implementations tips can be summarized in the steps below:

- **Step 1**: Parametrise the system including the weights in a Matlab script, Assign the variables to a struct object to be called by the functions and scripts as this approach makes it easier to manage huge numbers of parameters.

  The weights can be assigned arbitrarily or intuitively at this stage, what matters at this step is to allocate memory space for them as vectors and matrices. The step 1 is encountered as the "main.m" in the Matlab/Simulink documentation.

- **Step 2**: Design and implement the linear plant model,

$$\begin{aligned} x_0(t+1) &= A_{0t}x_0(t) + B_{0t}u_0(t-k) + D_{0t}\zeta_0(t) + G_{0t}d_{0d}(t), \\ y(t) &= C_{0t}x_0(t) + E_{0t}u_0(t-k) + d_0(t), \\ y_m(t) &= C_{0mt}x_0(t) + E_{0mt}u_0(t-k) + d_{0m}(t), \end{aligned}$$

  that represents the LPV models. The physical equations of the plant are entered and discretized in an internal Matlab script named as the "sys.m", or similarly. This is not strictly the plant to be controlled. It is the model of the plant used to calculate the control signals and estimate states. They could be slightly different. However, caution must be paid for potential model mismatch issues.

- **Step 3**: At this stage model the physical system to be controlled by using Simulink or if desired use a Matlab Fcn or S-function. Consider open-loop control to investigate stability of the system.

- **Step 4**: Separate the linear plant model W_{0k} and the non-linear subsystem W_{1k} ($W_{1k} = I$ if the system does not include the unstructured NL subsystem).

  For example, the NL subsystems may be at the input level and act as hard limits on the controller applied by the saturation blocks.

- **Step 5**: Design a PID controller that stabilizes the closed-loop system (keep the gains $\overline{k}_c$ for parallel form approach or PID motivated weighting),

- **Step 6**: If the results are satisfying, design the error terms and combine them with the tracking errors to obtain functions $F_e(t)$,

- **Step 7**: Calculate the RS-PID input as $F_e(t)\overline{k}_c$ either with Matlab s-functions or Simulink blocks (like the implementations in this thesis),

- **Step 8**: Use LPVKF to observe the system states. Tune the covariance matrices if necessary.

  It is good practice to make sure the LPVKF operates well before proceeding with implementing the RS-NGMV controller as the success of the optimization depends on utilizing the estimations.

  The LPVKF is implemented by a level-1 s-function in which the LPV plant model script "sys.m" is called and the augmented system model is constructed,

$$\begin{aligned} x(t+1) &= A_t x(t) + B_t u_0(t-k) + D_t \xi(t) + d_d(t), \\ y(t) &= C_t x(t) + E_t u_0(t-k) + d(t), \\ y_m(t) &= C_{mt} x(t) + E_{mt} u_0(t-k) + d_m(t), \\ z_m(t) &= C_{mt} x(t) + E_{mt} u_0(t-k) + d_m(t) + v_m(t). \end{aligned}$$

Then the state and error predictor models are used following the predictor-corrector logic (shown in Fig. 2.6),

$$\begin{aligned} \hat{x}(t+k|t) &= A_t^k \hat{x}(t|t) + \sum_{j=1}^{k} A_{t+j}^{k-j} B_{t+j-1} u_0(t+j-1-k) + d_{dd}(t+k-1), \\ e_p(t+i) &= C_{pt+i} x(t+i) + E_{pt+i} u_0(t+i-k) + d_p(t+i). \end{aligned}$$

- **Step 9**: Implement a state-space block for the dynamic control weighting F_{ck}. The dynamic control weighting matrices should already be extracted in step 1's “main.m”, using ssdata(sys) approach explained in chapter 2,
- **Step 10**: The $F_e(t)$ terms, dynamic F_{ck} and LPVKF observations $z(t)$ carrying the estimations $\hat{x}(t)$, are fed to another level-1 s-function to calculate the total gains $k_c(t)$. Then the RS-NGMV optimization is performed and the $k_c(t)$ is calculated (as shown in Figures Fig. 3.4 and Fig. 3.5),
- if $W_1(k) = I$, use:
  $k_c(t) = -X_0(t)^{-1}(P_p(t)(d_{pd}(t+k) + C_{pt+k}\hat{x}(t+k|t)) + \psi_k(t))$.
- Otherwise:

$$\begin{aligned} k_c(t) = &-({\Lambda_k}^2 + {\Lambda_d}^2)^{-1}(P_p(t)(d_{pd}(t+k) + C_{pt+k}\hat{x}(t+k|t)) + \psi_k(t) \\ &+ {F_e}^T(t)(F_{ck} + E_{pt+k}^T {\Lambda_p}^2 E_{pt+k} + {\Lambda_u}^2) W_{1k}) F_e(t) k_c(t)). \end{aligned}$$

- **Step 11**: If parallel form of the RS controller is used, separate the gain $k_c(t) = \overline{k}_c + \widetilde{k}_c(t)$. Let RS-NGMV calculate the deviations and combine with the fixed PID gains (Fig. 3.3 style),
- **Step 12**: Extract the final $F_e(t)k_c(t)$ signal and apply to the physical model of the plant,
- **Step 13**: Follow the weighting guidelines recommended in section 3.4 or other techniques and choose/tune the weights.
- **Step 14**: The previous step completes the implementation of the RS-NGMV. For simulations, the "main.m" script must be run first to initialize the parameter sets. The script calls the main Simulink file when it is finished. Then the Simulink model is run and results can be viewed. The rest is debugging if needed or fine-tuning the controller weightings.

For details on how the implementation is performed, refer to Matlab/Simulink codes and diagrams in Appendix B-F.

3.6 Closing Remarks

In this possibly the most important chapter, we have derived the RS-NGMV controller for a more general state-space structure and novel solution. The SISO and MIMO RS controllers have been introduced whose gains are optimized using NGMV algorithm.

In chapter 5, the RS-NGMV algorithm is verified with a two-link robotic manipulator case study. The controller uses PID as the restricted structure and the reference tracking performance is investigated. Through simulations some important features of the RS-NGMV are shown and discussed.

An EV case study is presented in the Chapter 7. The RS-NGMV is used for achieving longitudinal speed tracking within well-known driving cycles under road grade disturbances. The disturbance rejection of RS-NGMV is observed.

Chapter 8 concludes the thesis while looking at future work. Some initial results from an attempt to use RS-NGMV in a method of storing controller gains to be used later will be shown. This is the so-called Scheduled RS-NGMV.

RS-NGMV has a lot of potential for future work and it has features yet to explore. For example, a more detailed stability analysis would surely be interesting and useful. Additional degrees of freedom structures or polynomial versions could also be considered.

Chapter 4

Preview Control

"We can only see a short distance ahead, but we can see plenty there that needs to be done."

Alan Turing, Computing Machinery and Intelligence.

The traditional definition of the preview control technique may be described as the method of deriving optimal control solutions by using future reference information (or future disturbance in some cases). The preview control is different than the predictive control mainly in the method of solution. The preview control solution is related to LQ or LQG type solutions whereas the predictive control solution is more related to the minimum variance type solution of Åström's [1].

The method emerged in 1966 with the work of Sheridan [64] who introduced the idea in three models. The motivation was to characterize the ability of a human (drivers, pilots and etc.) or artificially intelligent operator to look-ahead or preview an input course to adjust their actions. The third model could be considered as the first optimal control strategy that considers the previewing feature. In 1968, Bender proposed an optimum linear preview control strategy [65] applied to a vehicle suspension system. Among the early works, perhaps the biggest impact was made by Tomizuka and Whitney presenting a discrete-time optimal preview controller for tracking problems by solving an LQ type cost function [66]. To date, several preview control approaches have been proposed and a significant number of them still use the LQ framework. As an example, the preview controller of [67] takes the H_∞ approach on a servomechanism. The controller in [68], is an adaptation of Tomizuka's standard LQ-Preview controller for optimal vehicle steering. There are also rather different preview control solutions like the fuzzy logic approach in [69] and the use of Artificial Neural Network (ANN) techniques as in [70]. The preview control subject has been trending among the control researchers more than before since the 2000s. The development of the technique over the years including an analysis of the notable approaches, applications and the directions are covered in a broad literature survey [71].

Automotive industry has shown a keen interest on the preview control since it was introduced. Many of the research papers and real-world application examples appear to be in this area. For example, Nissan and Toyota are among the big companies investing in the preview control research. They have made studies on path tracking and active suspension systems using preview control [72, 73]. The automotive applications mostly focus on suspension systems and vehicle guidance. The two branches include providing comfortable driving, improving security by increased vehicle stability, trajectory following, lane keeping or changing since they are all closely related to utilizing the information of the road ahead. Another automotive focused preview research interest is on the Advanced Driver Assistance Systems (ADAS) [74]. However, the research is not constrained to the automotive industry because any control application where there is access to the future reference information is open to preview control solutions. In [75] researchers used preview control with Zero-Moment Point approach for a humanoid robot problem in which the walking pattern was known. The work in [76] considers an application for a robotic manipulator which is an example where future reference trajectories are often pre-determined. Another example is on the wind turbines where wind profile is estimated [77].

The amount of work done is substantial but the topic has not matured yet. There is a lack of straightforwardness to offer the preview feature meaning there is a need to generalize the preview control solutions under a theoretical framework. The development of simulation tools like the Preview Control Toolbox in [78] is also needed. The toolbox provides a H_2/H_∞ solution to tackle the computational burden of solving Algebraic Riccati Equation (ARE) or shortly Riccati Equation (RE) [79]. However, the results are only for LTI systems like the majority of other preview controllers in the literature.

There exists only a very small number of papers with LPV modelling focus. The work in [72] is an example of such where a robust preview controller has been derived using H_∞ and Linear Matrix Inequality (LMI) techniques. To our knowledge, it was the only study with LPV modelling for the preview control of autonomous vehicle steering problem before the results from the case study in Chapter 6 of this thesis were published [19]. The SDRE control approach to preview control has been proposed motivated by these findings with the aim to provide a straightforward LPV or SD adaptations for the standard LQ-Preview controller.

4.1 Preview Control Concept

The preview control can be classified as an optimization problem as the controller is derived from the solution of some cost functions. The preview action is often added as an extra degree of freedom to the design, in the format of feed-forward control conceptualized as in Fig. 4.1. The control input that refers to this scheme can be generalized by the following equation,

$$u(t) = K_{fb}x(t) + \sum_{i=0}^{N_p} K_{ff}p(t+i), \tag{4.1}$$

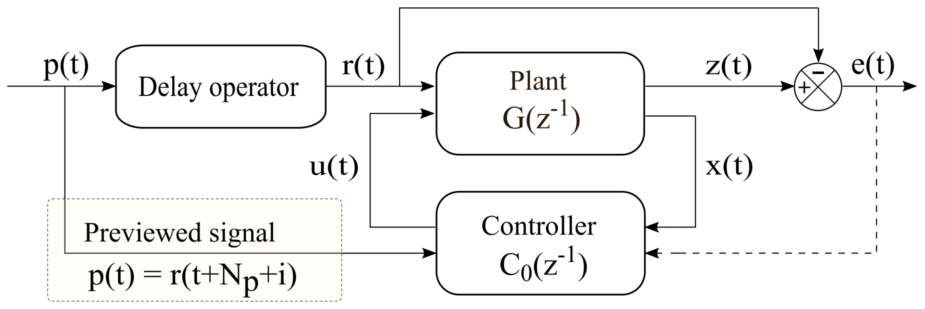


Figure 4.1: Conceptual preview control scheme.

where K_{fb}, $x(t)$, N_p, $p(t)$, K_{ff} refers to the feedback gain, states, preview length, previewed signal and preview gain (feedforward) respectively. Information of the reference signal $r(t)$ is available for N_p steps, and provided to the controller in a shift-register like mechanism. The objective is then to calculate the feedback gains control gains and feedforward preview gains to derive the control input of (4.1) for the tracking of desired reference trajectories. Note that, the preview controller of (4.1) is assumed to have state feedback but different preview control structures may include error feedback terms as well [67].

As one would expect, the preview control has its similarities with the predictive control as both methods look into the future when generating the control signals. Despite the similarities, they differ from each other particularly for the cost functions. For example, the standard preview control is based on the LQ framework and demands the solutions of algebraic Riccati equations. Therefore, their control solutions might seem more computationally expensive. However, they do not require real-time prediction at each time step like the predictive controllers. Being based on the well-structured LQ theory, the preview controllers are also expected to hold good robustness characteristics. There are studies that compare these methods in detail. The study in [80] presents theDynamic Performance Preview-Predictive Control (DPPC) index which provides links between the LQ-Preview and the GPC controllers. Depending on the choice, the index provides either of the controllers or their combination. Another study with interesting results show that for long preview lengths and long control horizons, predictive and LQ based preview approaches give identical results [81].

4.2 LQ Preview Control

In this entire section, we proceed with deriving the LQ-Preview controller [80] or the so-called standard preview controller since it is the most commonly used one.

The results follow from the preview control principles of section 4.1 but the LQ-Preview control is implemented differently. Its gain vector $K = \left(K_{fb}, K_{ff}\right)$ also includes both the state feedback and preview gains but it has the control structure of a LQ state feedback controller with future information utilized in its solution. The first task is to formulate the state-space system to incorporate the previewed reference and the second task is to calculate the preview control gains through LQ optimization solutions and thus derive the control input.

4.2.1 Augmented State-Space System

Consider the discrete-time LTI state-space system to be controlled,

$$x(t+1) = Ax(t) + Bu(t), \tag{4.2}$$

$$y(t) = Cx(t), \tag{4.3}$$

where $x(t)$ is an $n \times 1$ state vector and $u(t) \in \mathcal{R}^m$, $y(t) \in \mathcal{R}^r$.

The reference signal y_r is deterministic within the preview horizon N_p, beyond which there is no more information (for example limited sensor vision),

$$y_r(t + N_p + i) = y_r(t + N_p), \tag{4.4}$$

where $i > 0$, and the reference model becomes uncertain i.e. a random process.

The state-space reference model is given by the expressions below,

$$x_r(t+1) = A_r x_r(t) + B_r w_r(t), \tag{4.5}$$

$$y_r(t) = C_r x_r(t), \tag{4.6}$$

where $w_r(t)$ is a stochastic white noise signal, and A_r, B_r and C_r are state matrices. The reference models (4.5) and (4.6) are synthesized in the fashion below,

$$\begin{pmatrix} y_r(t+1) \\ y_r(t+2) \\ \vdots \\ y_r(t+N_p) \\ x_r(t+N_p+1) \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & \dots & 0 & 0 & C_r \\ 0 & \dots & 0 & 0 & A_r \end{pmatrix} \begin{pmatrix} y_r(t) \\ y_r(t+1) \\ \vdots \\ y_r(t+N_p-1) \\ x_r(t+N_p) \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ B_r \end{pmatrix} w_r(t+N_p), \tag{4.7}$$

and then (4.7) is compressed (or redefined) within a single equation as,

$$Y_r(t+1) = \overline{A}_r Y_r(t) + \overline{B}_r \overline{w}_r(t). \tag{4.8}$$

The new reference model of (4.8) above is now combined with the plant model (4.2) and become the augmented system model,

$$\begin{pmatrix} x(t+1) \\ Y_r(t+1) \end{pmatrix} = \begin{pmatrix} A & 0 \\ 0 & \overline{A}_r \end{pmatrix} \begin{pmatrix} x(t) \\ Y_r(t) \end{pmatrix} + \begin{pmatrix} B \\ 0 \end{pmatrix} u(t) + \begin{pmatrix} 0 \\ \overline{B}_r \end{pmatrix} \overline{w}_r(t). \tag{4.9}$$

The (4.9) can be expressed as the summarized model below,

$$\mathcal{X}(t+1) = \mathcal{A}\mathcal{X}(t) + \mathcal{B}u(t) + \mathcal{B}_r \overline{w}_r(t). \tag{4.10}$$

meaning (4.9) and (4.10) are the same. The output of the state-space system in (4.10) is defined by $Y(t)$,

$$Y(t) = y(t+i) - y_r(t+i), \tag{4.11}$$

which by using (4.3) and (4.8) becomes,

$$Y(t) = Cx(t+i) - \begin{pmatrix} 1, & 0, & \ldots, & 0 \end{pmatrix} Y_r(t+i), \tag{4.12}$$

and is finalized following the augmented system formulation,

$$\begin{align} Y(t) &= \begin{pmatrix} C, & -1, & 0, & \ldots, & 0 \end{pmatrix} \mathcal{X}(t+i) \tag{4.13} \\ &= \mathcal{C}\mathcal{X}(t+i). \tag{4.14} \end{align}$$

The models of (4.10) and (4.14) represent the augmented the state-space system which will be used for the control solutions from here on.

4.2.2 Control Solution

Define the LQ cost function at time t,

$$\begin{align} J_t = \frac{1}{N+1} E\{Y(t+N)^T S Y(t+N) + \sum_{i=1}^{N-1} [Y(t+i)^T Q Y(t+i) \tag{4.15} \\ + u(t+i)^T R u(t+i)]\}, \end{align}$$

with the expectations taken with respect to $w_r(t+N_p)$, $w_r(t+N_p+1)$, $\ldots$,$w_r(t+N-1)$ where N is the control horizon.

For the augmented system with previewed reference in (4.10), the cost function above is re-defined in,

$$\begin{align} J_t = \frac{1}{N+1} E\{\mathcal{X}(t+N)^T \mathcal{C}^T S \mathcal{C} \mathcal{X}(t+N) \tag{4.16} \\ + \sum_{i=1}^{N-1} [\mathcal{X}(t+i)^T \mathcal{C}^T Q \mathcal{C} \mathcal{X}(t+i) + u(t+i)^T R u(t+i)]\}. \end{align}$$

The solution of (4.16) is equivalent to the general LQ control solution and returns the preview controller in,

$$u(t) = -K\mathcal{X}(t) = -(\mathcal{B}^T P(t+1)\mathcal{B} + R)^{-1}\mathcal{B}^T P(t+1)\mathcal{A}\mathcal{X}(t), \tag{4.17}$$

where P matrix satisfies the solution of the Riccati Equation,

$$P(t) = \mathcal{A}^T P(t+1)\mathcal{B}(\mathcal{B}^T P(t+1)\mathcal{B} + R)^{-1}\mathcal{B}^T P(t+1)\mathcal{A} + \mathcal{A}^T P(t+1)\mathcal{A} + \mathcal{C}^T Q\mathcal{C}, \tag{4.18}$$

summarizing the LQ-Preview controller.

This thesis considers the preview control for the autonomous vehicle steering problems. A noteworthy approach that uses the LQ-Preview control technique of this section for such problem is Sharp's Linear Quadratic Regulator (LQR) Preview controller [68]. However, Sharp's approach tailors the LQ-Preview controller for the car steering problem with the road and vehicle dynamics taken into account. Therefore, it is slightly different. Firstly, the solution is derived over infinite horizon and secondly the augmented output matrix $\mathcal{C}$ includes vehicle dynamic transformation elements as the output matrix of LQ-Preview approach will not work straightforward for the problem. However, Sharp's approach is for LTI systems, thus the preview controller of section 4.3.3 of this chapter aims to extend his solution to a larger class of systems including LPV.

4.3 SDRE Approach to Preview Control

The SDRE control approach was proposed in [17] for the non-linear dynamic systems encapsulated in the linear structures called as the State-Dependent Coefficient (SDC) forms. It is an optimal control technique derived by using similar procedures to LQ based algorithms.

The SDRE control technique has performed well for many applications including robotic manipulators [84]. It has received a growing interest from researchers and is well-established. In fact, a comprehensive survey study [18] clearly shows that there have been so many successful applications of the method even surpassing the theoretical results.

4.3.1 SDRE Control

This section will revise the results of [17] and derive the standard SDRE controller. The SDRE control [17] uses the LQR cost function re-arranged as in,

$$J(x(t), u(t)) = \frac{1}{2}\int_{t_0}^{\infty} \big(x(t)^T Q(x) x(t) + u(t)^T R(x) u(t)\big) dt, \tag{4.19}$$

where weightings $Q(x)$ and $R(x)$ are allowed to be functions of states. To indicate that a system element is state-dependent the simplified notation x is used instead of $x(t)$ to reduce crowded terms due to the RE. A non-linear system can be represented by the general formula,

$$\dot{x}(t) = f(x) + g(x)u(t), \tag{4.20}$$

which can also be expressed in linear structure using the SDC form,

$$\dot{x}(t) = A(x)x(t) + B(x)u(t). \tag{4.21}$$

The SDRE equation for the (4.22) is defined in,

$$A^T(x)P(x) + P(x)A(x) - P(x)B(x)R^{-1}(x)B^T(x)P(x) + Q(x) = 0.$$

that is solved for $P(x) \geq 0$. Using the solution, the SDRE control signal is calculated,

$$u(t) = -K(x)x(t) = -R^{-1}(x)B^T(x)P(x)x(t). \tag{4.22}$$

The SDRE algorithm is demonstrated by the flowchart in Fig. 4.2, The measure-

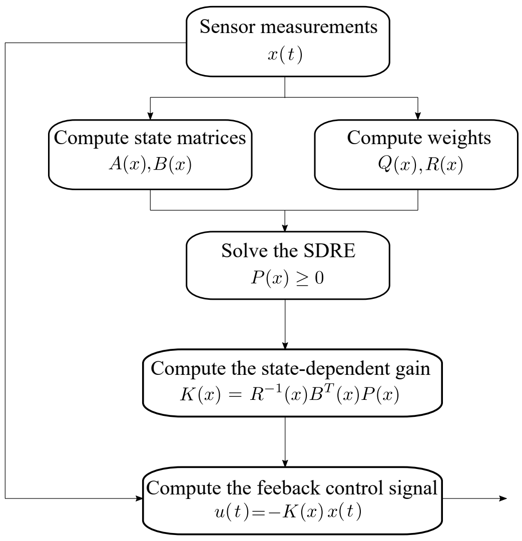


Figure 4.2: SDRE flowchart.

ments $x(t)$ convey the system information and the state matrices along with the weightings are calculated. These are used in the SDRE equation which is solved for $P(x) \geq 0$ and compute the controller gain $K(x)$. Lastly, the control signal $u(t)$ is calculated and sent to the plant. The methods used in this thesis are in discrete-time.

While having introduced the SDRE technique in its original way, it is important to note that the solution for the discretized (4.21) will be,

$$u(t) = -K(x)x(t) = -(R(x) + B^T(x)P(x)B(x))^{-1}B^T(x)P(x)A(x)x(t), \quad (4.23)$$

where $P(x) \geq 0$ satisfies the discrete SDRE solution,

$$P(x) = A(x)^T(P(x)-P(x)B(x)^T(R(x)+B(x)^TP(x)B(x))^{-1}B(x)P(x))A(x)+Q(x).$$

This summarizes the background information for the SDRE Preview and LPV-RE Preview controllers that follow.

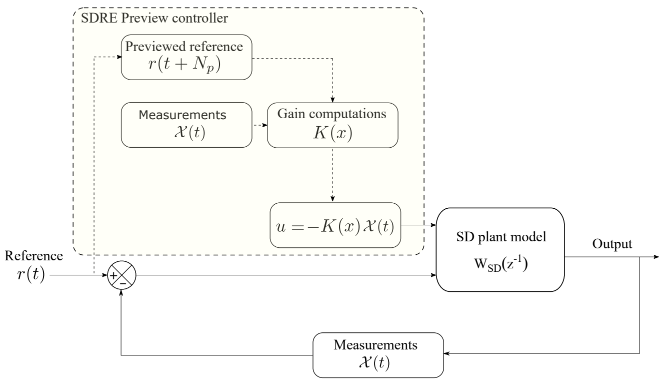


Figure 4.3: SDRE Preview control diagram.

4.3.2 SDRE Preview Control Solution

Transforming the LTI augmented system model with preview information introduced in (4.10) by using the linear SDC form,

$$\mathcal{X}(t+1) = \mathcal{A}(x)\mathcal{X}(t) + \mathcal{B}(x)u(t) + \mathcal{B}_r\overline{w}_r(t), \tag{4.24}$$

with the output given by,

$$Y = \mathcal{C}(x)\mathcal{X}(t). \tag{4.25}$$

Define the discrete SDRE cost function for the SD system in (4.25),

$$J(\mathcal{X}(t), x, u(t)) = \lim_{N\to\infty} \sum_{t=0}^{N} \big(\mathcal{X}(t)^T \mathcal{Q}(x)\mathcal{X}(t) + u^T(t)R(x)u(t)\big), \tag{4.26}$$

where $\mathcal{Q}(x) = \mathcal{C}(x)^T Q(x)\mathcal{C}(x)$. As the control horizon $N \to \infty$, the minimization of the cost function J in (4.26) returns the SDRE Preview control law,

$$u(t) = -K(x)\mathcal{X}(t) = -(R(x) + \mathcal{B}(x)^T P(x)\mathcal{B}(x))^{-1}\mathcal{B}(x)^T P(x)\mathcal{A}(x)\mathcal{X}(t), \tag{4.27}$$

where P satisfies the discrete Riccati Equation,

$$P(x) = \mathcal{A}(x)^T(P(x) - P(x)\mathcal{B}(x)^T(R(x) + \mathcal{B}(x)^T P(x)\mathcal{B}(x))^{-1}\mathcal{B}(x)P(x))\mathcal{A}(x) + \mathcal{Q}(x).$$

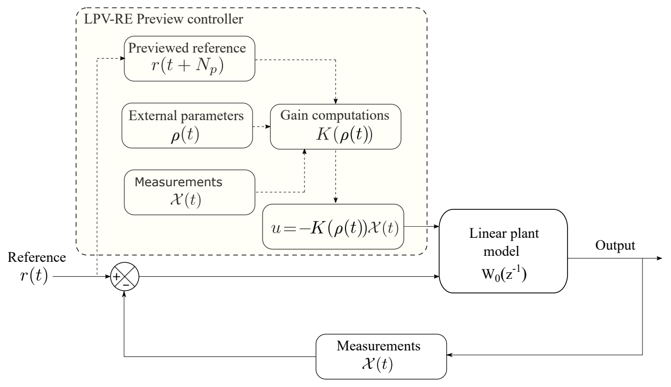

LPV-RE Preview controller
Previewed reference $r(t+N_p)$
External parameters $\rho(t)$
Gain computations $K(\rho(t))$
Measurements $\mathcal{X}(t)$
$u = -K(\rho(t))\mathcal{X}(t)$
Linear plant model $W_0(z^{-1})$
Output
Reference $r(t)$
Measurements $\mathcal{X}(t)$

Figure 4.4: LPV-RE Preview control diagram.

The basic SDRE Preview control diagram is shown in Fig. 4.3. The plant $W_{SD}(z^{-1})$ presents the model (4.24) with its output (4.25). The controller gain $K(x)$, is a combination of state feedback and preview gains such that $K(x) = \big(K_{fb}(x), K_{ff}(x)\big)$.

4.3.3 LPV-RE Preview Control Solution

The basic LPV-RE Preview control diagram is shown in Fig. 4.4. The linear plant model $W_0(z^{-1})$ is similar to the linear plant subsystem of the RS-NGMV formulations and is allowed to be LPV, LTI or LTV. With this, the algorithm can also accommodate the external parameters $\rho(t)$ within the augmented system model and the solution will remain the same. Similar to the SDRE preview control notation, we use ρ instead of $\rho(t)$. For a LPV system we can re-consider (4.24) within the LPV formulation such as,

$$\mathcal{X}(t+1) = \mathcal{A}(\rho)\mathcal{X}(t) + \mathcal{B}(\rho)u(t) + \mathcal{B}_r\overline{w}_r(t), \tag{4.28}$$

and the output would then be given by,

$$Y = \mathcal{C}(\rho)\mathcal{X}(t). \tag{4.29}$$

Define the LPV-RE cost function for the LPV system in (4.28),

$$J(\mathcal{X}(t), \rho, u(t)) = \lim_{N\to\infty} \sum_{t=0}^{N} \big(\mathcal{X}(t)^T \mathcal{Q}(\rho)\mathcal{X}(t) + u^T(t)R(\rho)u(t)\big), \tag{4.30}$$

where $\mathcal{Q}(\rho) = \mathcal{C}(\rho)^T Q(\rho)\mathcal{C}(\rho)$. When control horizon N approaches infinity, the minimization of (4.30) returns the LPV-RE Preview control law $u(t) = -K(\rho)\mathcal{X}(t)$ expanded as in,

$$u(t) = -(R(\rho) + \mathcal{B}(\rho)^T P(\rho)\mathcal{B}(\rho))^{-1}\mathcal{B}(\rho))^T P(\rho)\mathcal{A}(\rho)\mathcal{X}(t), \tag{4.31}$$

where P satisfies the LPV Riccati equation,

$$P(\rho) = \mathcal{A}(\rho)^T (P(\rho) - P(\rho)\mathcal{B}(\rho)^T (R(\rho) + \mathcal{B}(\rho)^T P(\rho)\mathcal{B}(\rho))^{-1}\mathcal{B}(\rho)P(\rho))\mathcal{A}(\rho) + \mathcal{Q}(\rho).$$

The controller gain $K(\rho)$, is also combination of state feedback and preview gains such that $K(\rho) = \big(K_{fb}(\rho), K_{ff}(\rho)\big)$.

4.4 Design Procedure

A suggested design procedure for the preview control algorithm including software implementations tips are summarized in the steps of this part:

- **Step 1**: System parametrisation including the assignment of weight matrices Q,R and the preview steps N_p with trial and error approach. The step 1 is encountered as the "main.m" in the codes as usual.
- **Step 2**: Design and implement the plant model. If SD use,

$$\begin{aligned}\mathcal{X}(t+1) &= \mathcal{A}(x)\mathcal{X}(t) + \mathcal{B}(x)u(t) + \mathcal{B}_r\overline{w}_r(t),\\ Y &= \mathcal{C}(x)\mathcal{X}(t).\end{aligned}$$

If the plant is LPV use,

$$\begin{aligned}\mathcal{X}(t+1) &= \mathcal{A}(\rho)\mathcal{X}(t) + \mathcal{B}(\rho)u(t) + \mathcal{B}_r\overline{w}_r(t),\\ Y &= \mathcal{C}(\rho)\mathcal{X}(t).\end{aligned}$$

- **Step 3**: At this stage model the physical system to be controlled by using Simulink or if desired use a Matlab Fcn or S-function. Consider open-loop control to investigate stability of the system.

- **Step 4**: Implement the preview controller following the SDRE flowchart of Fig.4.2. If the plant is SD the solution is the SDRE Preview controller,

  $$u(t) = -K(x)\mathcal{X}(t) = -(R(x) + \mathcal{B}(x)^T P(x)\mathcal{B}(x))^{-1}\mathcal{B}(x)^T P(x)\mathcal{A}(x)\mathcal{X}(t).$$

  If the plant is LPV the solution is the LPV-RE Preview controller,

  $$u(t) = -K(\rho)\mathcal{X}(t) = -(R(\rho) + \mathcal{B}(\rho)^T P(\rho)\mathcal{B}(\rho))^{-1}\mathcal{B}(\rho))^T P(\rho)\mathcal{A}(\rho)\mathcal{X}(t),$$

- **Step 5**: Both systems are implemented using the dlsim command of Matlab and the Riccati Equations are solved using dlqr function. As long the systems matrices are entered properly the dlqr command solves for the P matrix and calculates the preview gains.

4.5 Closing Remarks

This chapter has introduced the SDRE approach to the preview control for the first time in the literature and derived the SDRE and LPV-RE Preview controllers. Our motivation in using the SDRE technique for the preview problem is that the technique has some theoretical stability theory available and numerous useful results as a practical method for controlling non-linear systems. There are not many simple and straightforward methods available but the SDRE.

In chapter 6, the LPV-RE Preview controller will be implemented on an LPV autonomous vehicle model with the varying parameter as the velocity. The initial results from this analysis was presented in [19]. The paper was titled as the "SDRE Preview Control for a LPV Modelled Autonomous Vehicle" to point out the SDRE approach was used for the preview controller. To be more precise the controller in that paper is re-named for this thesis as the LPV-RE Preview controller.

The main conclusion of this chapter is that the preview control is valuable and more work can be done including the preview action for Restricted Structure controllers. For example, since the cost-functions are closely related to a Restricted-Structure version of the Non-linear LQG controller may be considered. The work in the thesis has already considered the use of preview action for the RS-NGMV and as a result proposes the RS-NPGMV controller for the solution. However, this is an entirely different controller and outside the scope of this PhD project. See Appendix A for further details.

The stability of analysis of the SDRE/LPV-RE Preview controllers can be studied using methods like the satisficing [110]. As an example, the theorems in [111] prove that using a satisficing set, the state-dependent feedback gain $K(x)$ is stabilizable for parameters that are bounded on a convex plane which looks like the case for the varying parameter V_x in our example. It is possible to use the method for proving a stabilizable $K(\rho)$ gain as well. These could be considered for future works.

Chapter 5

Two-Link Robotic Manipulator Control

"...We were born to work together like feet, hands and eyes, like the two rows of teeth, upper and lower..."

Marcus Aurelius, Meditations.

Robot manipulators are devices that are used to perform tasks like manipulating or moving materials, tools, parts without direct human contact. They are widely used in the industry for welding, assembly, painting, packaging and testing applications. Their range of applications also extend to space and surgical operation systems. Due to these factors, they have emerged as a popular subject of research. The control of robotic arms can be a difficult job considering that their dynamics are highly non-linear by nature. The challenge of non-linearity can be addressed by different modelling approaches. There has been an increasing demand for LPV modelling solutions in the non-linear control area including robotic applications. The qLPV representation of robotic manipulators have been verified with research results like [85 – 88].

The qLPV approach enables the expression of the non-linear robotic system in the so-called LPV framework. As expressed earlier, the idea is to simply contain the non-linear characteristics inside a structure that appears to be a linear model. In this section, the qLPV approach has been employed to represent the dynamics of a 2-link robotic arm. A multi-variable RS-NGMV controller will be designed for the model to complete the task of link position control via reference tracking. The RS takes the form of a classical digital PID controller with filtered derivative term. Transient and stochastic performances of the controller are investigated considering set-point changes.

5.1 Robotic Manipulator Dynamics

The robot manipulators are basically composed of joints, links and end-effectors. Links are rigid or flexible body elements that are connected via joints which can be revolute, cylindrical, prismatic, spherical or planar depending on the design of the manipulator system. End-effectors are mounted on the tip of the links for handling of the objects, carrying tools, interacting with surfaces and etc. Grippers are the simplest type of end-effectors designed only for the actions of opening and closing to grasp objects [89].

The manipulator in this study is an elbow-type, two-link, revolute-joint robot. Each joint has a single degree-of-freedom thus the manipulator has two-degrees-of-freedom. As illustrated in the Fig. 5.1, the robot operates horizontally on a two dimensional coordinate system for which the base of robot is considered the origin.

Note that, the mechanical design and kinematics are not within the scope of this example. The control problem deals with the forces producing the desired motion. Therefore, the emphasis is placed upon the aspects of dynamic equations which relates the forces and motion.

Euler-Lagrange method have been used for modelling of the robot dynamics

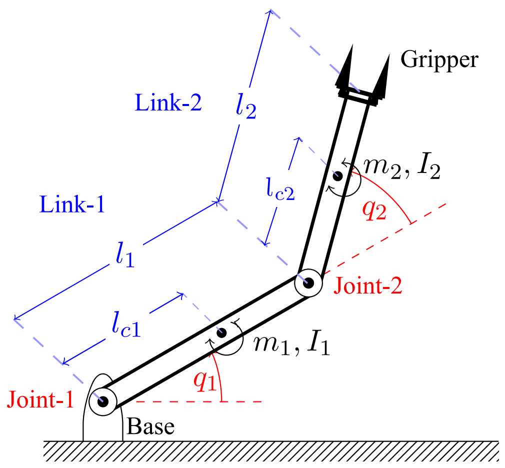


Figure 5.1: Two-link robot arm.

as shown in detail in [90 − 93]. The Lagrangian equations are derived from the difference between the system's kinetic and potential energies. They are represented by the matrix form,

$$M(q)\ddot{q} + H(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) = \tau, \tag{5.1}$$

where $M(q)$ represents the inertia matrix and $H(q,\dot{q})$, $F(\dot{q})$, $G(q)$ represent the Coriolis matrix, friction and gravity vectors, respectively. Torque is given by the vector τ while q denotes the joint angle vector.

The forces of friction are neglected and since the robot is assumed to operate horizontally, the affect of gravitational forces can be omitted as well. Hence, expanding the dynamic equations given in the matrix form in (5.1) returns the expression below,

$$\begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix} \begin{pmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{pmatrix} + \begin{pmatrix} h\dot{q}_2 & h\dot{q}_1 + h\dot{q}_2 \\ h\dot{q}_1 & 0 \end{pmatrix} \begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \end{pmatrix} = \begin{pmatrix} \tau_1 \\ \tau_2 \end{pmatrix}. \tag{5.2}$$

The equation indicates the non-linear characteristics of the robot manipulator. The non-linearity becomes even clearer when the elements of the inertia matrix

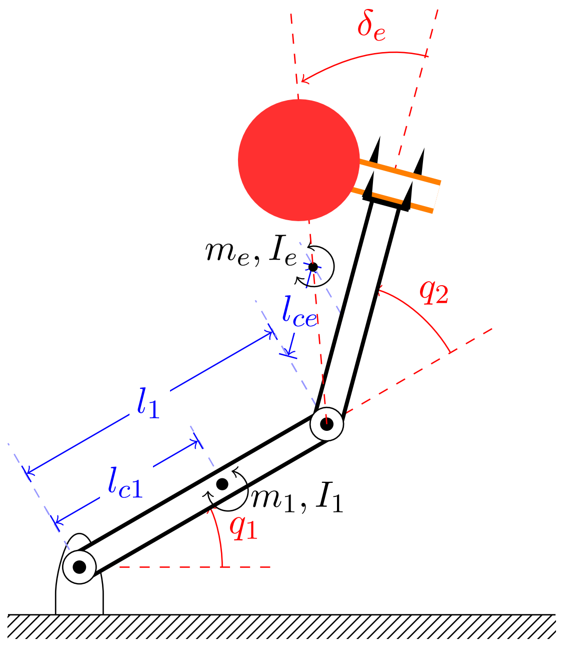


Figure 5.2: Two-link robot arm with unknown load.

M and the Coriolis matrix H are broken down,

$$\begin{aligned}
M_{11} &= I_1 + m_1 {l_{c1}}^2 + m_2({l_1}^2 + {l_{c2}}^2 + 2l_1 l_{c2} cosq_2) + I_2,\\
M_{12} &= M_{21} = m_2 l_1 l_{c2} cosq_2 + m_2 {l_{c2}}^2 + I_2,\\
M_{22} &= m_2 {l_{c2}}^2 + I_2,\\
h &= m_2 l_1 l_{c2} sinq_2.
\end{aligned}$$

The simulation studies of this chapter adopts the robot model and parameters from Example 9.1 in [94]. The example considers the same model above with an unknown load held by the robot's gripper as seen in the Fig. 5.2 for which a table tennis racket is used merely for illustrative purposes. The referenced study introduces the unknown load for estimation of unknown parameters and develop an adaptive control strategy. In our example, we consider these system parameters to be known and keep the rest of the approach nevertheless, since it represents a realistic application scenario with the manipulator holding an object.

As it can be expected the load impacts the manipulator dynamics thus the inertia and Coriolis elements need to be re-arranged in the following fashion,

$$\begin{aligned}
&M_{11} = p_1 + 2p_3 cos q_2 + 2p_4 sin q_2,\\
&M_{12} = M_{21} = p_2 + p_3 cos q_2 + p_4 sin q_2,\\
&M_{22} = p_2,
\end{aligned}$$

with the set of parameters, $p = (p_1, p_2, p_3, p_4)$, chosen to be,

$$\begin{aligned}
&p_1 = I_1 + m_1 {l_{c1}}^2 + I_e + m_e {l_{ce}}^2 + m_e l_1^2,\\
&p_2 = I_e + m_e {l_{ce}}^2,\\
&p_3 = m_e l_1 l_{ce} cos\delta_e,\\
&p_4 = m_e l_1 l_{ce} sin\delta_e,
\end{aligned}$$

and finally $h = p_3 sin q_2 - p_4 cos q_2$. The inertia matrix M is invertible, thus through re-arranging of the equations it is possible to represent the system in the form below,

$$\dot{x} = \begin{pmatrix} \dot{q} \\ \ddot{q} \end{pmatrix} = \begin{pmatrix} 0_{2\times2} & I_{2\times2} \\ 0_{2\times2} & M^{-1}(q)H(\dot{q}) \end{pmatrix} \begin{pmatrix} q \\ \dot{q} \end{pmatrix} + \begin{pmatrix} 0_{2\times2} \\ M^{-1}(q) \end{pmatrix} \tau. \tag{5.3}$$

$$y = q = \begin{pmatrix} I_{2\times2} & 0_{2\times2} \end{pmatrix} \begin{pmatrix} q \\ \dot{q} \end{pmatrix}. \tag{5.4}$$

where $q = (q_1, q_2)^T$ and $\tau = (\tau_1, \tau_2)^T$. The robotic system in (5.3) and (5.4) is of the form,

$$\begin{aligned}
\dot{x}(t) &= A\big(x(t), p(t)\big)x(t) + B\big(x(t), p(t)\big)u(t),\\
y(t) &= C\big(x(t), p(t)\big)x(t) + D\big(x(t), p(t)\big)u(t).
\end{aligned}$$

and is a qLPV system indeed (depending on the physical model, matrices could also be functions of the control input).

For the controller design and Matlab/Simulink implementations the system is discretized by taking the Euler approximation technique. The procedure is realized by $A_d = 1+T_sA$, $B_d = T_sB$, $C_d = C$, $D_d = D$, with the sample time, T_s.

5.2 Simulation Studies

System parameters used within the simulation studies are given by, $m_1 = 1$, $I_1 = 0.12$, $l_1 = 1$, $l_{c1} = 0.5$, $m_e = 2$, $I_e = 0.25$, $l_{ce} = 0.6$, $\delta_e = 30°$. The control objective is to have the robot follow a desired position trajectory $q_d = (q_{1d}, q_{2d})^T$. The trajectory consists of set-point changes which are specified with detail in,

$$q_d = \begin{cases} (60°, 90°)^T, & \text{for } t \leq 50 \\ (45°, 45°)^T, & \text{for } 50 < t \leq 100 \\ (60°, 90°)^T. & \text{for } t > 100 \end{cases}$$

Control inputs are transmitted as torque signals to the joints. A saturation limit of $\pm 2Nm$ is forced on the torque inputs. Torque inputs are subjected to a torque load of $T_d = 0.4Nm$. The sampling time for the simulation studies is set to $T_s = 0.005s$.

For the RS-NGMV control strategy, the parallel form expressed in (3.22) is a suitable design approach because a PID controller with gains k_{PID} already exists,

$$k_{PID} = \begin{pmatrix} k_p \\ k_I \\ k_D \end{pmatrix} = \begin{pmatrix} 3 \\ 0.5 \\ 7 \end{pmatrix}.$$

Likewise, the PD control is also a quite common for robotic manipulator control. However, to demonstrate the properties of RS-NGMV more clearly the full PID

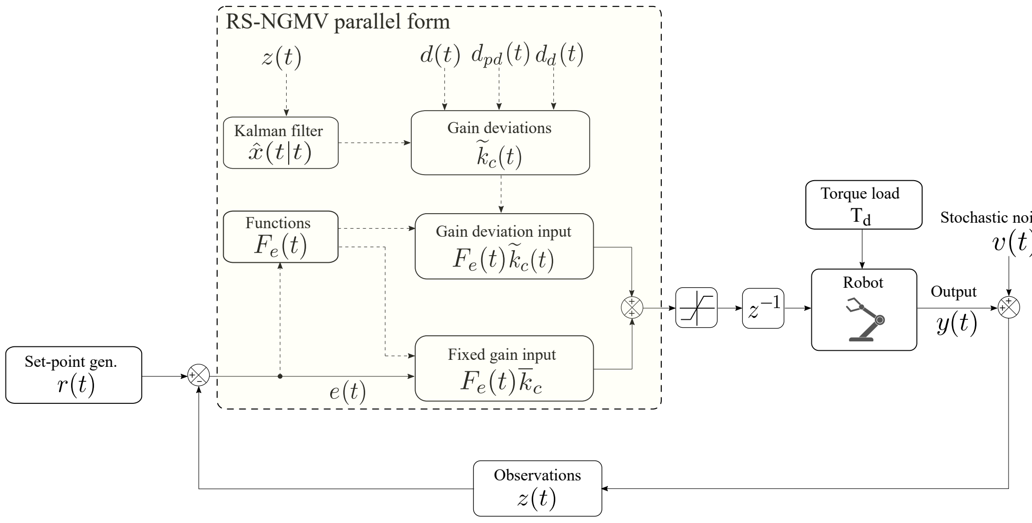

RS-NGMV parallel form
$z(t)$
$d(t)$ $d_{pd}(t)$ $d_d(t)$
Kalman filter $\hat{x}(t|t)$
Gain deviations $\widetilde{k}_c(t)$
Functions $F_e(t)$
Gain deviation input $F_e(t)\widetilde{k}_c(t)$
Fixed gain input $F_e(t)\overline{k}_c$
Torque load T_d
Stochastic noise $v(t)$
Robot
z^{-1}
Output $y(t)$
Set-point gen. $r(t)$
$e(t)$
Observations $z(t)$

Figure 5.3: RS-NGMV control of the robot.

has been preferred for its RS in this case study. During the simulation runs, the time-varying gain deviations $\widetilde{k}_c(t)$ will be calculated and added to the constant gains $\overline{k}_c(t)$, defining the final controller gain $k_c(t)$. See Fig. 5.3 for the implementation of the controller.

$$\overline{k}_c = \begin{pmatrix} k_{PID} \\ 0_{6\times 1} \\ k_{PID} \end{pmatrix}_{12\times 1} .$$

Next, the error and dynamic control cost weightings are specified by the following,

$$P_c(z^{-1}) = \begin{pmatrix} 87.5\dfrac{1-0.97z^{-1}}{1-z^{-1}} & 0 \\ 0 & 87.5\dfrac{1-0.97z^{-1}}{1-z^{-1}} \end{pmatrix},$$

$$F_{ck}(z^{-1}) = \begin{pmatrix} 0.016\dfrac{1-0.9z^{-1}}{1-0.4z^{-1}} & 0 \\ 0 & 0.16\dfrac{1-0.9z^{-1}}{1-0.4z^{-1}} \end{pmatrix},$$

using the weighting strategies described in chapter 3. The compensator transfer functions were preferred for realistic roll-off of signals.

The weighting to penalize the deviations in controller gains is assigned as,

$$\lambda_k = \begin{pmatrix} 3\times10^{-6} & 0 & 0 \\ 0 & 0.5\times10^{-6} & 0 \\ 0 & 0 & 7\times10^{-6} \end{pmatrix},$$

which forms the diagonal weight matrix, $\Lambda_k^2 = diag\left(\lambda_k, \quad \lambda_k, \quad \lambda_k, \quad \lambda_k\right) \in \mathcal{R}^{12\times12}$.

The weighting on increments on the deviations in control gains is chosen by,

$$\lambda_d = \begin{pmatrix} 40 & 0 & 0 \\ 0 & 40 & 0 \\ 0 & 0 & 40 \end{pmatrix},$$

that makes up the weight matrix, $\Lambda_d^2 = diag\left(\lambda_d, \quad \lambda_d, \quad \lambda_d, \quad \lambda_d\right) \in \mathcal{R}^{12\times12}$.

The constant weighting on the control inputs and weighting on tracking errors are expressed by the matrices below, respectively,

$$\Lambda_p^2 = \begin{pmatrix} 10^{-4} & 0 \\ 0 & 10^{-4} \end{pmatrix}, \Lambda_u^2 = \begin{pmatrix} 0.05 & 0 \\ 0 & 0.05 \end{pmatrix}.$$

5.2.1 Transient Performance

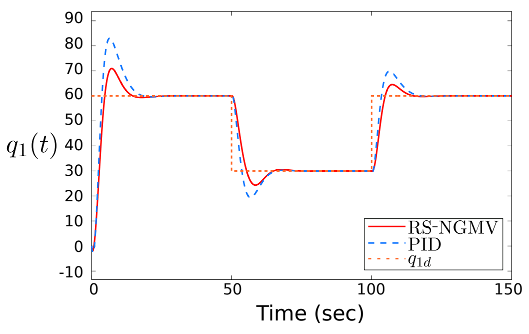
The first set of results consider the reference angle tracking problem along with set-point changes. The results are grouped in Fig. 5.4 starting with the positions of robot links. It is shown in Fig. 5.4(a) and 5.4(b) that the robot link angles follow the desired position trajectory q_d.

The PID controller has been chosen as the baseline control method. Firstly, a set of constant PID gains denoted by $\overline{k}_c$ has been selected by testing through trial and error. Using the parallel form, the RS-NGMV controller gains are produced as a combination of the fixed PID gains and the gain deviation terms,

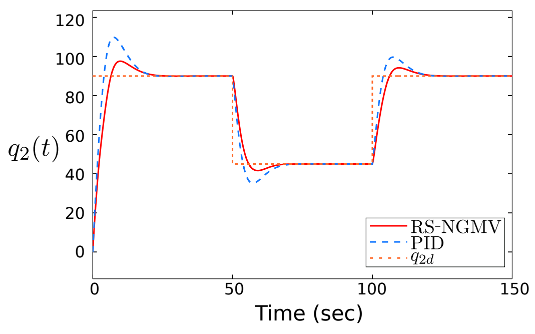
$\widetilde{k}_c(t)$, which result from the feedback optimization and Kalman filter observations. The optimized gain terms are demonstrated Fig. 5.4(c) and 5.4(d). It is shown that the deviation terms are constantly updated in the background. It becomes more noticeable when the reference signal is generated at $t = 0$s for the first time and set-point changes occur at $t = 50$s and $t = 100$s. Calculation efforts increase during the transitions. The RS-NGMV controller attempts to adapt to the changes by re-adjusting its gains. The gain deviations rise and then slow down as the link angles $q_1(t)$ and $q_2(t)$ approach the steady-state and once it is reached they remain constant unless demanded otherwise. This is an important feature of the RS-NGMV control method. The controller is able to adapt to the changes within the system. From this perspective, although it may not be theoretically classified as one, the RS-NGMV philosophy is very close to the adaptive controllers. For instance, PID based Model Reference Adaptive Control uses optimization and learning rates to modify its PID control input [95].

5.2.2 Baseline PID Controller

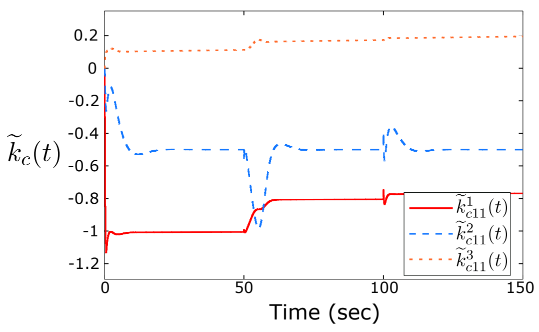
The Fig. 5.4 also compares the position tracking performances of RS-NGMV and PID. As illustrated in parts (a) and (b), both controllers are successful in realizing the control objective. The RS-NGMV inherits the advantages of the NGMV optimal control thus it shows better performance compared to that of the PID. It shows smaller overshoots and the control signals shown in 5.4 (e) and (f) are smoother. However, it is important to clarify that they are both different controllers and RS-NGMV does not lay a claim to being a PID alternative or a better controller in general sense. It utilizes the PID as its controller structure (e.g. Restricted Structure) and uses the fixed PID gains to calculate its controller gains. As expressed earlier, the motivation of RS-NGMV is to provide optimal NGMV solutions through the PID structure for classically trained staff who are more familiar with it.



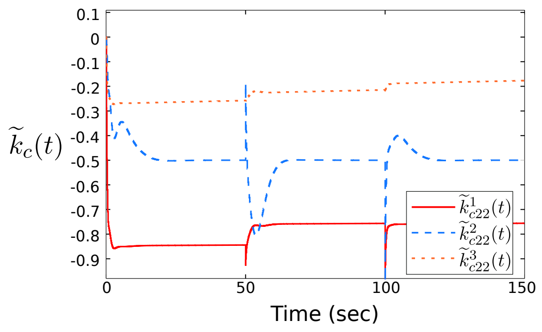
(a) Link-1 positions



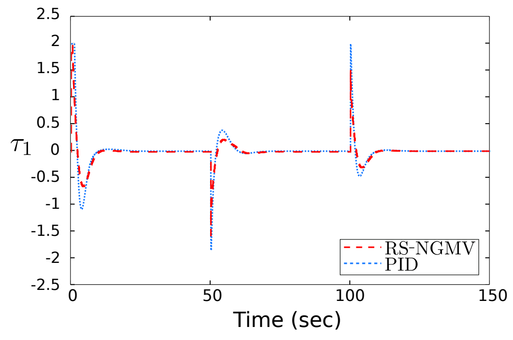
(b) Link-2 positions



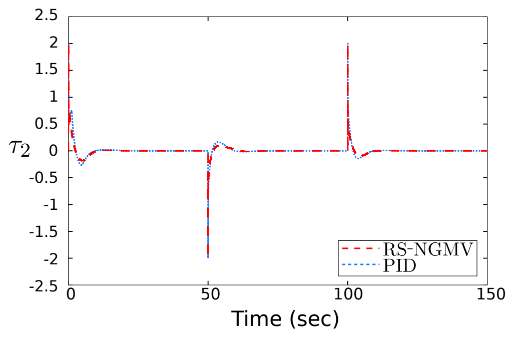
(c) RS-NGMV gains $\widetilde{k}^1_c(t)$



(d) RS-NGMV gains $\widetilde{k}^2_c(t)$

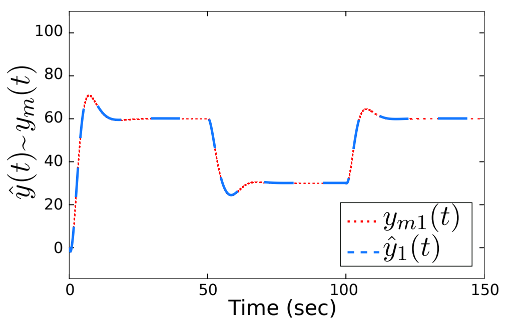


(e) Control input $u_1(t)$

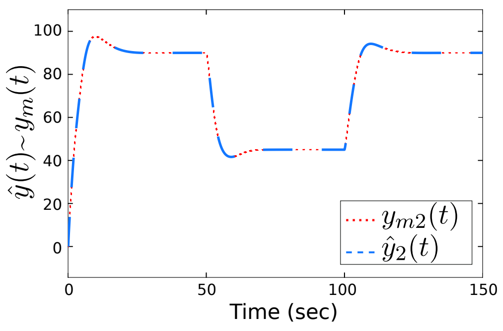


(f) Control input $u_2(t)$

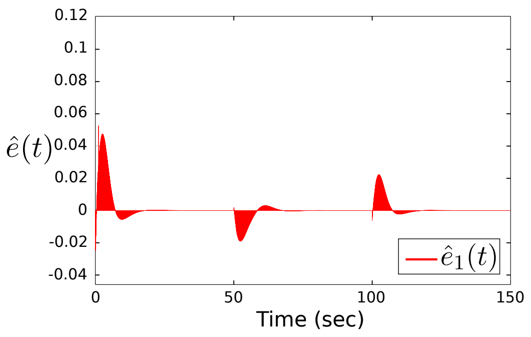
Figure 5.4: Reference angle tracking with set-point changes.



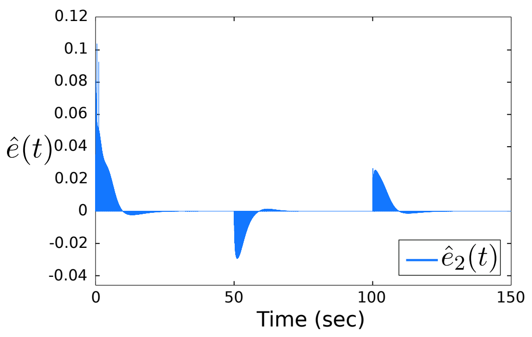
(a) Output and estimates 1



(b) Output and estimates 2



(c) Estimate errors 1



(d) Estimate errors 2

Figure 5.5: Output signals and estimates.

5.2.3 Kalman Filter Estimates

As the RS-NGMV uses state estimates to calculate $k_c(t)$, it is essential that the filter works properly. The estimates are shown in Fig. 5.5, where part (a) and (b) compare the measured values of robot link angles to their estimated values. Estimate errors are presented in part (c) and (d) where it is seen that they are minimal. It is shown that the LPVKF converges to steady-state. The diagonal covariance matrices P, Q and R are tuned manually through trial and error. Initial conditions of the state estimate vector $\hat{x}(0)$ are set to zero.

5.2.4 Stochastic Performance

The simulations so far have considered only the transient performance of the controller. The results in Fig. 5.6 analyse the stochastic behaviour when a fictitious output disturbance signal has been introduced to the system. The output disturbance signal for each link consists of an arbitrarily chosen vector of step signals combined with Gaussian white noise. In addition, the stochastic white noise has been low-pass filtered to attenuate high-frequency content. The total noise signal is given in part (g) of Fig. 5.6 which is scaled down by a factor of 10 for illustrative purposes. The results in part (a) through part (f) show that, the RS-NGMV maintains its response by adapting to the changes and achieves the control objective. In addition, gain deviations and the control input characteristics have remained similar to the non-disturbance case. RS-NGMV has again shown better performance compared to the PID.

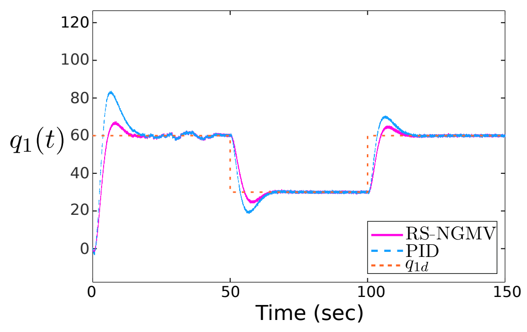
Note that for the stochastic case, the constant weighting on increments on the deviations in control gains has been re-adjusted to,

$$\lambda_d = \begin{pmatrix} 80 & 0 & 0 \\ 0 & 80 & 0 \\ 0 & 0 & 80 \end{pmatrix},$$

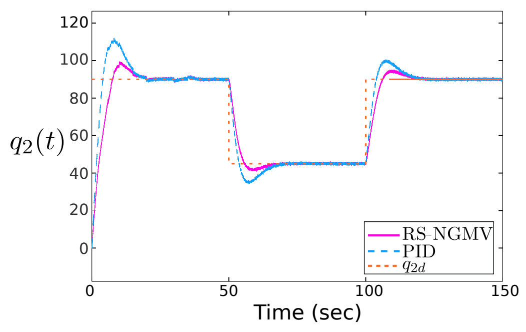
for the diagonal weight matrix, $\Lambda_d^2 = diag\left(\lambda_d, \;\; \lambda_d, \;\; \lambda_d, \;\; \lambda_d\right) \in \mathcal{R}^{12\times12}$. The output results with the previous value of λ_d were identical to part (a) and (b). However, the RS-NGMV control inputs demonstrated slightly sharper responses than expected. By investigating the gain plots, it was understood that an increased cost on their increments could result in control inputs that are less aggressive. This shows that the additional weightings work to the advantage of the RS-NGMV algorithm.

5.3 Closing Remarks

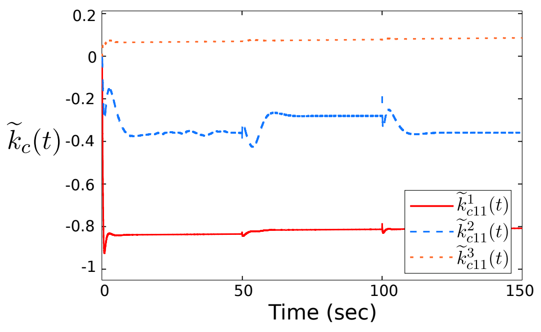
The robotic manipulator example in this chapter is the first implementation of the RS-NGMV control method in the literature. The most important output from the



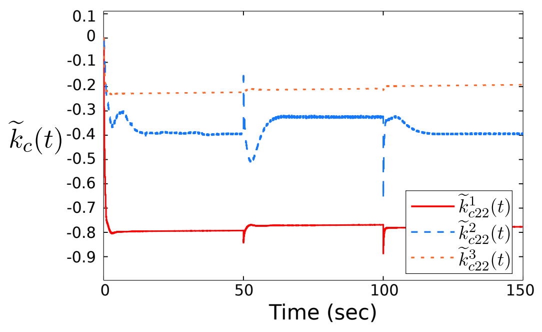
(a) Link-1 positions



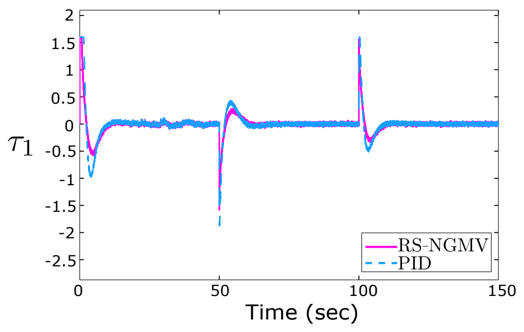
(b) Link-2 positions



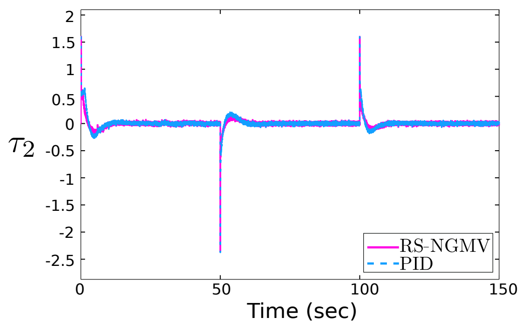
(c) RS-NGMV gains $\widetilde{k}^1_c(t)$



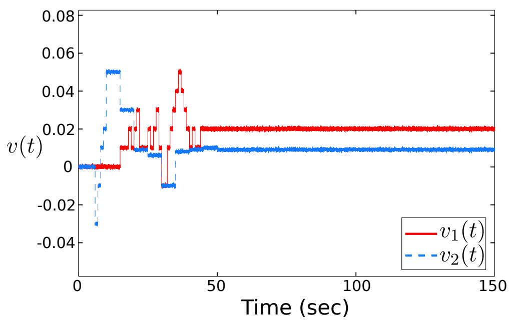
(d) RS-NGMV gains $\widetilde{k}^2_c(t)$



(e) Input $u_1(t)$



(f) Input $u_2(t)$



(g) Stochastic signal

Figure 5.6: Reference angle tracking under disturbance.

results of this case study is the adaptability of the controller. With the LPVKF providing information from the system to the controller, the optimizations can update the control gains to respond to the parameter changes. This makes RS-NGMV highly suitable for the control of LPV systems.

In this chapter, the models and the RS-NGMV design do not consider the input non-linearity. Therefore, it has been assumed that $W_{1k} = 1$ and the control solution in (3.31) has been implemented. The input non-linearity is only applied at the simulation layer. It is introduced by the saturation block imposing the hard limits $(u_{min}, u_{max}) = \pm 2Nm$ on the control signal. The model mismatch did not have an effect on the performance. The solution for the RS-NGMV with input non-linearity has already been provided should an alternative be needed for another application case.

This case study has considered disturbances for a more realistic approach. Torque load with constant value of $T_d = 0.4Nm$, has been applied on the inputs. It is easily handled by the controller and has not impacted the performance at all. The stochastic performance has also been investigated with the noise enforced on the output. For different applications, however, the disturbance may be a function of system parameters or states. As a result, the disturbance modelling becomes more crucial. This is encountered in the EV application example in chapter 7.

Chapter 6

Autonomous Vehicle Steering Control

“*The vehicles will be fully self-driving. So you have your own personal space where you can sit back and relax.*”

John Krafcik, CEO of Waymo.

This chapter brings on the autonomous vehicle steering control as a case study for the LPV-RE Preview control of chapter 4. The vehicle dynamics will be modelled with the LPV approach first, followed by the LPV-RE Preview control design. The simulation studies will investigate the controller performance under parameter variations and use LQ-Preview control as the baseline technique [68]. The remainder of this introduction represents my understanding of the current autonomous vehicles scene by following the online course [121].

The current era of automobiles is considered to be the dawn of the driverless car by many enthusiasts, researchers and pioneers in this field.

Darpa Urban Challenge in 2005 was surely a very important milestone for the self-driving car. University of Stanford’s autonomous car Stanley [96] came the winner of a 132 miles autonomous off-road race by completing the track the fastest. The control algorithm used is named Stanley controller in the literature and is based on the kinematic car model. Although the method works very well, Stanley does not reflect many characteristics of a fully self-driving car. It neglected some aspects of the vehicle dynamics and forces. Noisy environments, traffic and severe weather conditions were also not considered.

A fully self-driving car takes on the full-time driving task under all road and environmental conditions equally to what a human driver handles. Such system is classified as level 5 according to the Society of Automotive Engineers (SAE)'s taxonomy for self-driving cars which is introduced under the standard J3016 [97].

Pioneers from many universities and companies have made impressive progress recently but the field faces challenges coming from all directions especially from safety considerations or the substantial amount of time needed for the testing processes. It can be understood that the transition to the driverless cars does and will demand solutions for many control problems [121].

6.1 Lateral Autonomous Vehicle Model

The lane changing is one of the most interesting questions to tackle for the autonomous vehicles. Researchers have applied various control methods in case studies and real-world implementations to address the issues [98, 99]. According to the report [100] of National Highway Traffic Safety Administration in the US, transportation researchers estimate that up to 10% of all crashes are lane-changing related. The introduction of the autonomous cars could impact the statistics so effective controllers are crucial to mitigate the risks.

From the controls perspective, to perform lane-changing manoeuvres successfully the algorithm must deal with the lateral vehicle dynamics. An important element of the lane changing is to handle the effect of external parameters like longitudinal velocity on the lateral vehicle dynamics. For this reason, the LPV modelling approach has been taken for the case study here.

Consider the illustrated lateral vehicle model in Fig. 6.1 known as the bicycle model in the literature [101], derived by transforming the front and rear wheels into singles. In the bicycle model, the V_x stands for the longitudinal velocity that moves the vehicle, the ψ being the yaw angle that decides the heading of the vehicle while the δ acts as the steering angle (control signal) and β is the side slip angle. The center of gravity is denoted by the cg, and the distances from the cg to the tires are given by l_r and l_f for the front and rear, respectively. The state-space model of the lateral vehicle dynamics is summarized by,

$$\begin{aligned} \dot{x}(t) &= A_t x(t) + B_t u(t), \\ y(t) &= C_t x(t) + D_t u(t). \end{aligned} \tag{6.1}$$

Expanding the model, the state matrices A_t and B_t are given by,

$$A_t = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{C_f+C_r}{mV_x(t)} & \frac{C_f+C_r}{m} & -\frac{C_f l_f - C_r l_r}{mV_x(t)} \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{C_f l_f - C_r l_r}{I_z V_x(t)} & \frac{C_f l_f - C_r l_r}{I_z} & \frac{C_f l_f}{I_z G} \end{pmatrix}, B_t = \begin{pmatrix} 0 \\ \frac{C_f}{mG} \\ 0 \\ \frac{C_f l_f}{I_z G} \end{pmatrix}, \tag{6.2}$$

where $V_x(t)$ is an external parameter and may be time-varying, i.e. it is the scheduling parameter $\rho(t) = V_x(t)$ allowing the state-space model classify as LPV. The remaining parameters G is ratio of steering wheel to road wheel angle with, m is the vehicle mass, I_z is the yaw moment of inertia and the C_f and C_r express the front and rear cornering stiffness coefficients. The state vector $x(t)$ contains the lateral position $y(t)$, the $\psi(t)$ and their derivatives,

$$x(t) = \begin{pmatrix} y(t) & \dot{y}(t) & \psi(t) & \dot{\psi}(t) \end{pmatrix} \tag{6.3}$$

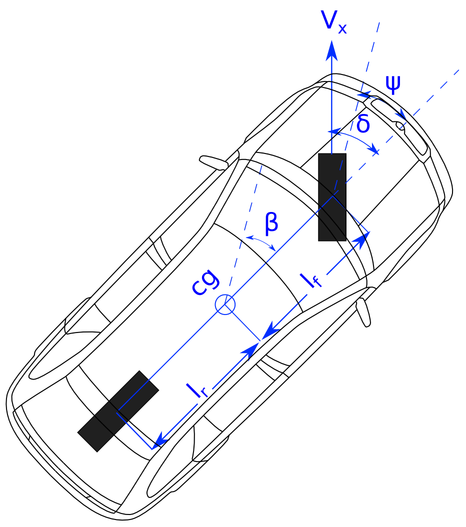


Figure 6.1: Bicycle model (CAD drawing [108] modified).

and the control input $u(t) = \delta(t)$. The output model expands below,

$$y(t) = \begin{pmatrix} y(t) \\ \psi(t) \end{pmatrix}, C_t = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, D_t = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \tag{6.4}$$

The models are discretized explicitly for the controller using forward Euler approximation approach which leaves $A_{dt} = (1 + T_s A_t)$, $B_{dt} = T_s B_t$, $C_{dt} = C_t$ and $D_{dt} = D_t$ with T_s being the sample time.

In section 6.2, the LPV-RE Preview controller is designed whose solution has been introduced in chapter 4 earlier. The original idea is that the controller has access to the road information ahead which will be incorporated in the final augmented state-space model before deriving the control solutions like Sharp introduced with his LQ-Preview controller [68]. However, if the plant dynamics are facing changes or i.e. parameter variations, it could prove a challenge for these controllers. The intuition is that the LPV-RE Preview controller can be a suitable control solution for the problem due to the fact that it uses optimized feedback gains which can respond to the changes imposed on the lateral dynamics by the varying external parameter $\rho(t) = V_x(t)$.

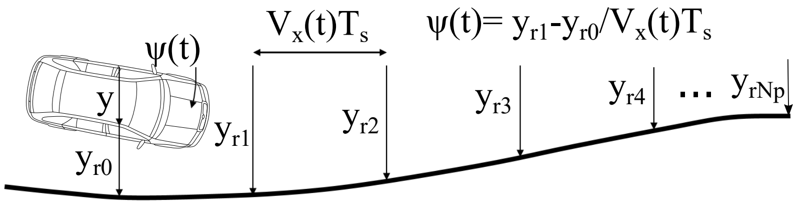


Figure 6.2: The previewed road [68].

6.2 LPV-RE Preview Control Design

In this section, the state-space representation of the reference road model is given and then used in constructing the augmented state-space model for the controller. The augmented state-space model is the LPV adaptation of the [68] and the controller will be the LPV-RE Preview algorithm introduced in this thesis. After the mentioned state-space models are presented, the LPV-RE Preview control solution will be derived and control parameters will be given.

6.2.1 Previewed Road Model

The road is previewed for N_p steps to be used as future reference information by the controller as shown in Fig. 6.2. The state-space road model is given below,

$$y_r(t+1) = A_r y_r(t) + B_r y_{rN_p}(t). \tag{6.5}$$

The vector $y_r(t)$ with size $N_p \times 1$ denotes the previewed road states while the y_{rN_p} represents the scalar input vector. The state matrices A_r and B_t are,

$$A_r = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ 0 & 0 & 0 & \dots & 0 \end{pmatrix}, B_r = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix} \tag{6.6}$$

6.2.2 Augmented System Model

Next, the previewed road model in (6.5) is combined with the LPV vehicle plant model in (6.1) that has been discretized,

$$\begin{pmatrix} x(t+1) \\ y_r(t+1) \end{pmatrix} = \begin{pmatrix} A_{dt} & 0 \\ 0 & A_r \end{pmatrix} \begin{pmatrix} x(t) \\ y_r(t) \end{pmatrix} + \begin{pmatrix} B_{dt} \\ 0 \end{pmatrix} \delta(t) + \begin{pmatrix} 0 \\ B_r \end{pmatrix} y_{rN_p}(t), \tag{6.7}$$

which is summarized as the model below,

$$\mathcal{X}(t+1) = \mathcal{A}_t \mathcal{X}(t) + \mathcal{B}_t \delta(t) + \mathcal{B}_r y_{rN_p}(t). \tag{6.8}$$

To complete the augmented system model, the dynamics approach in [68] was followed which has already been illustrated in the Fig. 6.2. The approach considers the distance between two preview points as $V_x(t)T_s$ and using this information to calculate the heading angle $\psi(t)$ which interprets the road information to the vehicle dynamics model through the new measurements matrix $\mathcal{C}$,

$$\mathcal{Y}(t+1) = \mathcal{C}_t \mathcal{X}(t) = \begin{pmatrix} 1 & 0 & 0 & 0 & -1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & \dfrac{1}{V_x(t)T_s} & \dfrac{-1}{V_x(t)T_s} & 0 & \dots & 0 \end{pmatrix} \mathcal{X}(t). \tag{6.9}$$

6.2.3 LPV-RE Preview Controller

Let the LPV-RE Preview control cost function $J(\mathcal{X}(t), \delta(t))$ be as in,

$$J(\mathcal{X}(t), \rho, \delta(t)) = \lim_{N\to\infty} \sum_{t=0}^{N} \big(\mathcal{X}(t)^T \mathcal{Q}(\rho) \mathcal{X}(t) + \delta^T(t) R \delta(t)\big), \tag{6.10}$$

for the augmented system in (6.8) and define the weighting on the states,

$$\mathcal{Q}(\rho) = \mathcal{C}_t^T Q \mathcal{C}_t. \tag{6.11}$$

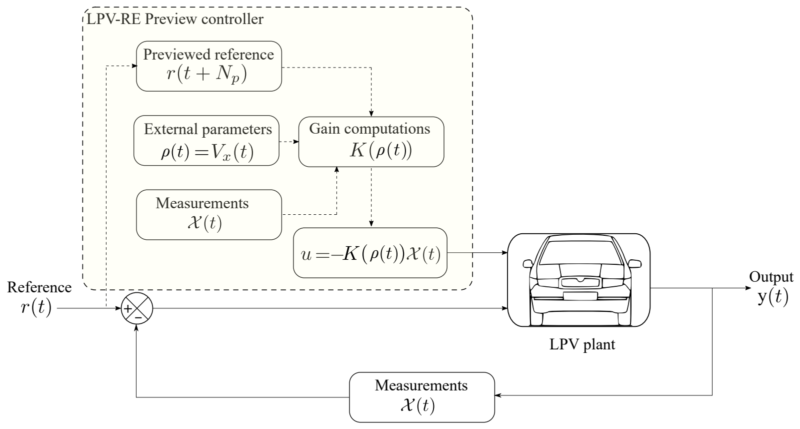


Figure 6.3: LPV-RE Preview control diagram.

where Q matrix's values are chosen as,

$$Q = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix},$$

and the control weighting term $R = 1$. Note that for notational simplicity, the algorithm's elements are not indicated as state-dependent this time. The solution of (6.10) returns the LPV-RE Preview control law,

$$\delta(t) = -K(\rho)\mathcal{X}(t), \tag{6.12}$$

implemented as in control diagram of Fig. 6.3. The gain vector K has the structure of $K(\rho) = \big(K_{fb}(\rho), K_{ff}(\rho)\big)$ including the state feedback and preview gains which are feed-forward. The gain vector K is,

$$K(\rho) = (R + \mathcal{B}_t^T P(\rho)\mathcal{B}_t)^{-1}\mathcal{B}_t^T P(\rho)\mathcal{A}_t\mathcal{X}(t), \tag{6.13}$$

with the $P(\rho)$ matrix from the solution of the discrete LPV-RE,

$$P(\rho) = \mathcal{A}_t^T P(\rho)\mathcal{A}_t + \mathcal{A}_t^T P(\rho)\mathcal{B}_t\Big(R + \mathcal{B}_t^T P(\rho)\mathcal{B}_t\Big)^{-1}\mathcal{B}_t^T P(\rho)\mathcal{A}_t + \mathcal{Q}(\rho). \tag{6.14}$$

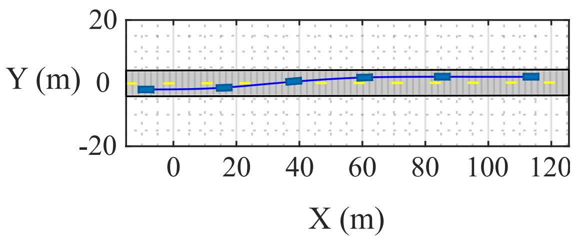


Figure 6.4: Lane changing manoeuvre.

6.3 Simulation Studies

The reference trajectory for the autonomous vehicle is demonstrated in Fig. 6.4 using Matlab's Driving Scenario toolbox. The lane-changing path is generated by having a series of adjacent way points connected. This is implemented using the linspace function and visualized using the toolbox. A common and a simpler way to generate reference paths for the autonomous vehicles, is to use line segments and join them or advanced mathematical techniques such as the clothoids also exist. However, the approach taken has been sufficient for the purposes of this study. The parameters chosen for the autonomous vehicle lateral model are given by the Table 6.1. The simulation results of this chapter are demonstrated in

Table 6.1: Autonomous vehicle lateral model parameters

Parameter	**Symbol**	**Value**
Gravity	g	$9.81m/s^2$
Vehicle Mass	m	$1573kgs$
Yaw Moment of Inertia	I_z	$2550kgm^2$
Cornering Stiffness Coefficient (front)	C_f	$2 * 88310N/rad$
Cornering Stiffness Coefficient (rear)	C_r	$2 * 64076N/rad$
Tire Distance to CG (front)	l_f	$0.913m$
Tire Distance to CG (rear)	l_r	$1.73m$
Steering Wheel to Tire Ratio	G	16
Sampling Time	T_s	$0.02s$
Preview Steps	N_p	250

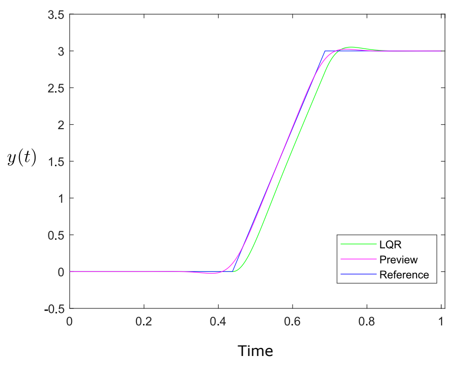
the Fig. 6.5 where a single lane-change is performed under 10 simulation seconds which are scaled in the graphs. There are two longitudinal speed profiles for the given set of results which are the constant velocity $V_x(t) = 20m/s$ and the varying $V_x(t)$ which initially starts at $20m/s$ with an acceleration rate of $2m/s^2$ and ending up at $40m/s$, shown in parts (e) and (f), respectively. When the longitudinal speed is constant, it is seen in part (a) that both the LQR and LPV-RE Preview controllers are successful in tracking the reference signal. However, the LPV-RE Preview controller is faster due to the use of future reference information. In real traffic conditions, considering a take-over might be needed or on-coming traffic is present, the task of lane-changing might require acceleration thus the parameter variations for the controller. The longitudinal velocity would vary in such case. The results in part (b) shows that while LPV-RE Preview approach can adapt to the changes, the LQR struggles to keep up as would expected since it is expected to need re-tuning. This is also observed in the state-feedback portion of the controller gains which takes the initial values given below,

$$K_{fb}(0) \approx \begin{pmatrix} 0.3095, & 0.076, & 4.602, & 0.4120 \end{pmatrix}.$$

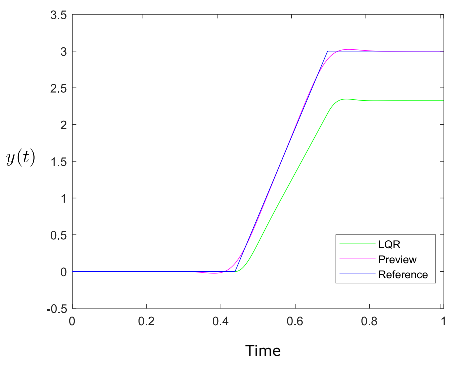
The gains remain constant for the LQR while they are updated continuously when the LPV-RE approach is taken. Because the variation is not very noticeable on the plots, steady-state values are presented to show that the gains are updated,

$$K_{fb}(\infty) \approx \begin{pmatrix} 0.3056, & 0.1128, & 4.4504, & 0.6497 \end{pmatrix}.$$

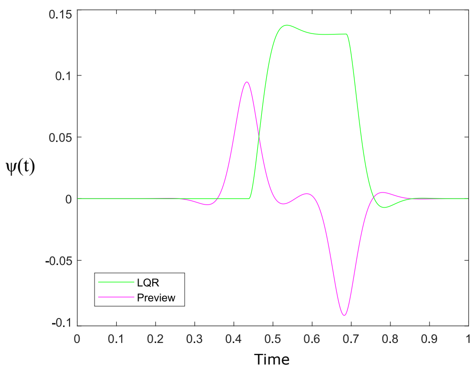
The results in part (c) and (d) compare the steering angle signals used by both controllers for the two velocity cases. The LPV-RE Preview requests the steering action earlier compared to the LQR due to the future reference information which helps the fast transient response for the lateral position of the vehicle. The control signals might seem different at first, but the steering principle is the same. In other words, both controllers perform sort of an S-shape steering action,



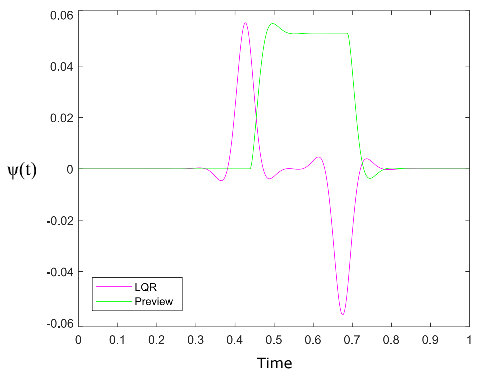
(a) Tracking with constant $V_x(t)$



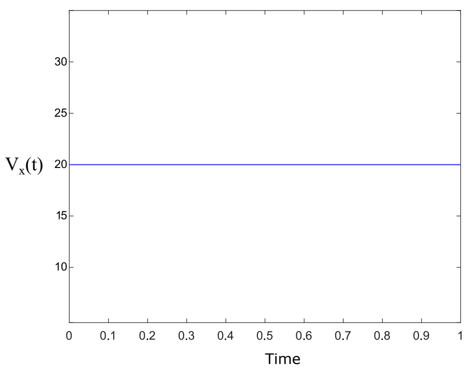
(b) Tracking with varying $V_x(t)$



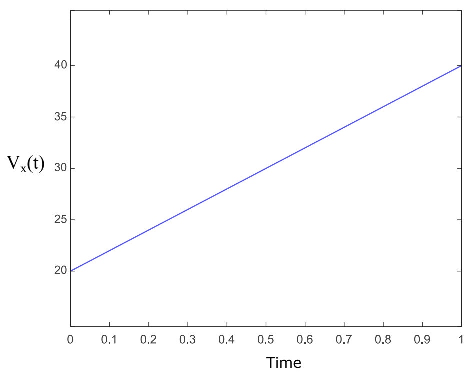
(c) Yaw angle with constant $V_x(t)$



(d) Yaw angle with varying $V_x(t)$



(e) Constant velocity profile



(f) Varying velocity profile

Figure 6.5: Reference lateral position tracking for lane-changing.

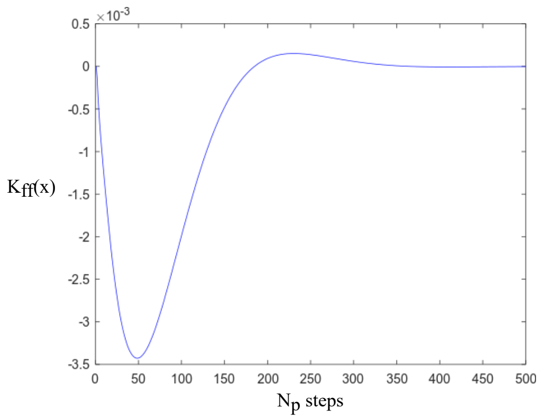


Figure 6.6: Preview gains vs. preview steps

only tweaked very differently. The preview controller gives quicker responses while waiting less in between the two main steering actions of the lane change manoeuvre.

6.4 Closing Remarks

For this study, the preview steps are chosen $N_p = 250$. The value was chosen with intuition, trial and error mixed together. Researchers state that the preview steps would reach a threshold beyond which no further benefits are possible [83]. There is not an algorithm or any studies in literature so far that deals with the problem of calculating an optimal preview length yet. This is also confirmed in the largest preview control literature review paper [71]. However, using the approach of this thesis which is demonstrated in Fig. 6.6, it may be possible to at least guess which numbers would be sufficient. In this figure, the feedforward preview gains are plot vs. the preview length N_p. We observe that much variations occur for the small number of steps which seem to saturate after some point. This approach remains intuitive though and without a formulation cannot be methodical. It is surely an interesting research question for the preview control area.

The case study of this chapter is the first implementation of the SDRE motived preview controllers in chapter 4. It is also the second time in the literature, a LPV system was considered for the preview control. The results have verified the advantage and effectiveness of the preview controller for the autonomous vehicle lane change problem for both constant and varying speed profiles.

Chapter 7

Electric Vehicle Control

"*I have actually made a prediction that within 30 years a majority of new cars made in the United States will be electric. And I don't mean hybrid. I mean fully electric.*"

Elon Musk, PBS Interview, 2008.

This case study takes the RS-NGMV approach to the longitudinal speed tracking control problem for a battery electric vehicle also known as pure/full electric vehicle, for which the sole source of energy are batteries. For convenience though, the notation EV has been used throughout the thesis to refer to this type of vehicle. The standard EV driving cycle scenarios and constant speed cruise control have been considered as the reference speed profiles. The EV has been modelled as a scalar qLPV system and the RS-NGMV algorithm with PI structure has been used as the control approach. The results illustrate the tracking behaviour under the impact of road and environmental disturbances.

The increasing concerns over the climate change combined with the success of the electric vehicles in the recent years are encouraging many countries into taking further environmental measures. Last year, the UK government announced the ban on the sales of new petrol and diesel engines by 2030 [102]. Automotive companies like GM has also announced their commitments to phase-out of fossil fuel vehicles [103]. Several other mega brands are committing to full EV models. It has been thus a motivating factor for this thesis as well, to consider the RS-NGMV performance for an EV control problem.

The longitudinal speed tracking problem is not a new topic for conventional vehicles where PI controllers have been the dominant solution for several applications such as [104]. The PI control approach appears to be the consensus for the EV implementations as well. However, with the new advances in electronics, external information from the vehicle environment such as the weather, road conditions and etc. could be utilized much more for performance or fuel economy improvements. Therefore, this research addresses the question of how an advanced control method, e.g. RS-NGMV, can contribute to the problem when considering the impact of external parameters.

The remainder of this chapter is organized as follows: Section 1 shows the longitudinal vehicle dynamics equations and derives the qLPV model for the controller and also presents the EV battery equations. Section 2 explains the derivation of the RS-NGMV controller and Section 3 demonstrates the analysis of the simulation results. Finally, some further comments are discussed under the closing remarks section.

7.1 EV Model

This section introduces the equations used to derive the EV model. First the longitudinal dynamics and second the battery equations are presented. The dynamics are represented within a qLPV model.

7.1.1 Longitudinal Dynamics and qLPV Model

Consider a Forward Wheel Drive (FWD) car driven on a road with inclination θ depicted in Fig. 7.1. The vehicle's traction, forward in our example, depends on

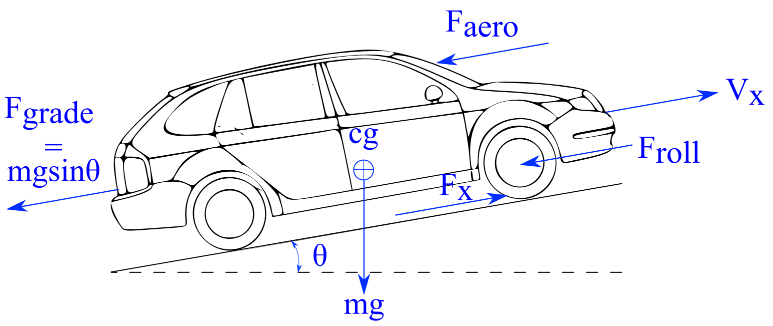


Figure 7.1: Longitudinal car dynamics (CAD drawing [108] modified).

the combination of several forces. The forces mentioned include: Aerodynamic drag, gravitational, tractive and rolling resistance forces.

The longitudinal dynamics express the relation of these forces. The mathematical principles of the models derived in this chapter are adapted from [105, 106]. The total force on longitudinal motion is summarized as in,

$$F_{total} = ma = F_x - F_{aero} - F_{grade} - F_{roll}, \tag{7.1}$$

where m denotes the mass of the vehicle and a denotes the acceleration, following Newton's second law of motion. The F_x represents the tractive force acting on the tires to move the vehicle in the desired trajectory. The opposing forces are external and act as disturbances. The first is the aerodynamic drag force,

$$F_{aero} = \frac{\rho A C_d}{2}(V_x + V_{wind})^2. \tag{7.2}$$

It consists of the pieces: ρ, the mass density of air, A the vehicle frontal area, C_d aerodynamic drag coefficient, V_x the longitudinal velocity and V_{wind} the wind speed. The study in this chapter assumes $V_{wind} = 0$, and thus $F_{aero} = (1/2)\rho A C_d V_x^2$.

The gravitational force, F_{grade}, results from the product of m and the acceleration of gravity g. When the road grade (inclination) factor is considered, it becomes,

$$F_{grade} = mgsin\theta. \tag{7.3}$$

The rolling resistance or rolling friction force is expressed by,

$$F_{roll} = mgC_r, \tag{7.4}$$

where C_r stands for the rolling resistance coefficient, a factor depending on tire pressure and other conditions.

The longitudinal dynamics can now be expressed by the state-space model,

$$\begin{aligned} \dot{V}_x &= -\frac{\rho A C_d V_x}{2m} V_x + \frac{1}{m} F_x - g\theta - gC_r, & (7.5) \\ y &= V_x + W_d, & (7.6) \end{aligned}$$

where $\dot{V}_x = a$. The longitudinal speed V_x is the controlled output subjected to W_d, the output disturbance which also may contain stochastic noise. Note that since the roads are constructed to have relatively small inclinations, the road grade force can been linearised at $\theta = 0$ and thus $mgsin\theta = mg\theta$. The non-linear system has been derived the way in (7.5) and (7.6) deliberately, to provide a qLPV model for the RS-NGMV control solution. It matches the representation,

$$\begin{aligned} \dot{x} &= A\big(x(t), p(t)\big)x + B\big(x(t), p(t)\big)u, \\ y &= C\big(x(t), p(t)\big)x + D\big(x(t), p(t)\big)u, \end{aligned}$$

with $x = V_x$ and selected scheduling parameters $p = (p_1, p_2)$ defined by,

$$\begin{aligned} p_1 &= -\frac{\rho A C_d}{2m}, \\ p_2 &= \frac{1}{m}, \end{aligned}$$

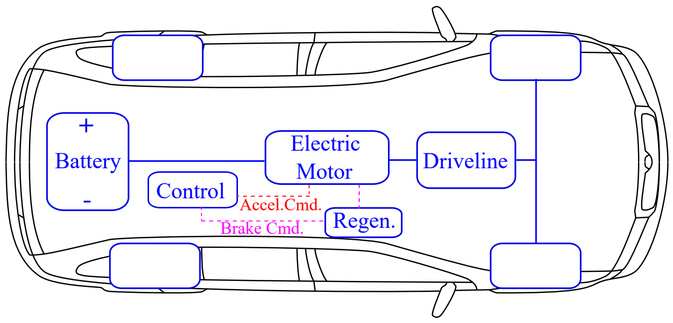


Figure 7.2: EV powertrain components (CAD drawing [108] modified)

As stated in chapter 2 section 4, a qLPV system may be function of states $x(t)$, inputs $u(t)$ and scheduling parameters $p(t)$. The $p(t)$ is a known parameter vector that may be time-varying and is chosen by the designer. For this example, it is a combination of constant parameters of the longitudinal vehicle dynamics.

To further clarify, the EV components are not individually represented in the state-space model. However, all EV components impact and determine the tractive force F_x.

The model is discretized by the explicit forward Euler approximations in the Matlab implementations at sampling time, $T_s = 1$s. The position of the vehicle is related to the velocity by $q = \int V_x dt$ discretized as well. This is later used for EV range calculations.

7.1.2 EV Powertrain

The main components of the EV powertrain are the battery, electric motor and the driveline as shown in Fig. 7.2. The battery provides the energy needed for the tractive force of the vehicle. The control signals send the acceleration and deceleration commands to the electric motor. The deceleration signals are received by the brakes with energy recovery (regeneration) property and transferred to the electric motor as torque requests. The electric motor receives all torque commands and provide the demand. Through the driveline the wheels receive the

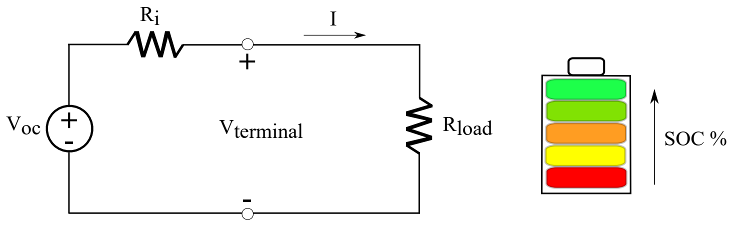


Figure 7.3: ECM and SOC%

traction force F_x.

Battery

The battery model used in this study is rather simplified. It is assumed that the battery is always able to supply the power demanded by the load unless it is completely discharged of course. The assumption means that parameters of the battery remain constant which is not the case for long term vehicle usage. These factors are considered by Battery Management Systems (BMS) using sophisticated methods like Kalman filtering, Least Squares and etc. [109]. However, BMS strategies are not within the scope of this analysis.

To define the battery model, the Equivalent Circuit Model (ECM) shown in Fig. 7.3 is used. The ECM represents the operation of a Li-ion cell model. The battery current of the cell is calculated below,

$$I = \frac{V_{oc} - \sqrt{V_{oc}^2 - 4R_i P_{motor}}}{2R_i}, \tag{7.7}$$

where V_{oc} is the Open-Circuit Voltage (OCV) and R_i is the internal resistance of the battery. P_{motor} is the motor power combined with the accessory load (AC, multi-media, etc.). The battery power is calculated with losses taken into account,

$$P_{battery} = IV_{oc} - I^2 R_i. \tag{7.8}$$

In more complicated ECM models, the open-circuit voltage is a function of the

Battery State-of-Charge (SOC) denoted by $z(t)$. SOC is the concept of indicating the available amount of charge level in the battery in percentage at a time instant (see Fig. 7.3).

A very crude analogy would be the fuel gauge in the conventional vehicles. However, unlike fossil fuels it is unfortunately not possible to measure the battery SOC. Therefore, estimation methods are used to tackle the problem. Once again, this is a topic for the BMS and not a concern of this thesis but studies as future work is already in progress.

In this chapter, it is assumed that the battery SOC at the start is always known and thus there is only the task of updating it which is given by,

$$z(t) = z(t-1) - \frac{IT_s}{Q}, \tag{7.9}$$

at time t, with Q being the energy capacity.

Electric Motor

The output power provided by the electric motor [107] is given by,

$$P_{motor} = P_i - P_{loss}, \tag{7.10}$$

where the input power P_i is subjected to the motor losses P_{loss}. To break down these terms, we first present,

$$P_{motor} = \tau_{motor}\omega_{motor}, \tag{7.11}$$

showing that the motor power output is the product of net motor torque τ_{motor} and the rotational speed ω_{motor} of the shaft. The input power is given by,

$$P_i = \tau\omega, \tag{7.12}$$

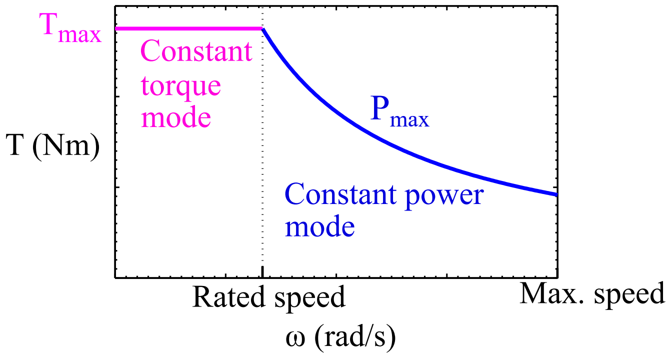


Figure 7.4: Motor torque vs. speed characteristics

and the motor loss model is expanded by,

$$P_{loss} = C + k_c\tau^2 + k_i\omega + k_w\omega^3, \tag{7.13}$$

with the parameters: Motor loss constants C, k_c, k_i, k_w, maximum torque τ and rated speed ω, related to the design of the motor whose characteristics are shown in Fig. 7.4.

By defining the motor input-output efficiency factor $\eta = P_{motor}/P_i$ as below, a dual operation mode is enabled, that is, the traction and the regeneration, respectively.

To represent the motor operation modes, define the factors,

$$If \quad \dot{V}_x \geq 0 \rightarrow \eta_{traction} = \frac{P_{motor}}{P_{motor} + P_{loss}}, \tag{7.14}$$

where the motor acts as a propeller, and,

$$If \quad \dot{V}_x < 0 \rightarrow \eta_{regen.} = \frac{P_{motor} + P_{loss}}{P_{motor}}, \tag{7.15}$$

where the motor acts like a generator during braking. The torque generated is delivered to the driveline which is the final component before the wheels.

Driveline

The driveline provides the tractive force F_x to move the tires while providing motor speed as feedback to the electric motor,

$$F_x = (\tau_{motor} - \tau_{loss})\frac{G}{r_w} - F_{br}. \tag{7.16}$$

The torque is subjected to losses τ_{loss} associated with the driveline and the net torque goes through the gear ratio G while being divided by the tire radius r_w. If there is any braking, this is subtracted as the brake force F_{br}.

7.2 RS-NGMV Control Design

This section starts with the introduction of the PI controller and then provides the RS-NGMV controller design.

7.2.1 Baseline PI Controller

The PI controller is used as the baseline control method for this chapter,

$$u(t) = k_p e(t) + \frac{1}{1-z^{-1}} k_I T_s e(t), \tag{7.17}$$

with gains k_p and k_I with the feedback error $e(t)$. To express the PI controller in the RS terms, we first formulate it as,

$$u(t) = f_1(z^{-1}) k_P e(t) + f_2(z^{-1}) k_I T_s e(t), \tag{7.18}$$

where the functions $f(z^{-1})$ are given by,

$$f_1(z^{-1}) = 1, f_2(z^{-1}) = \frac{1}{1-z^{-1}},$$

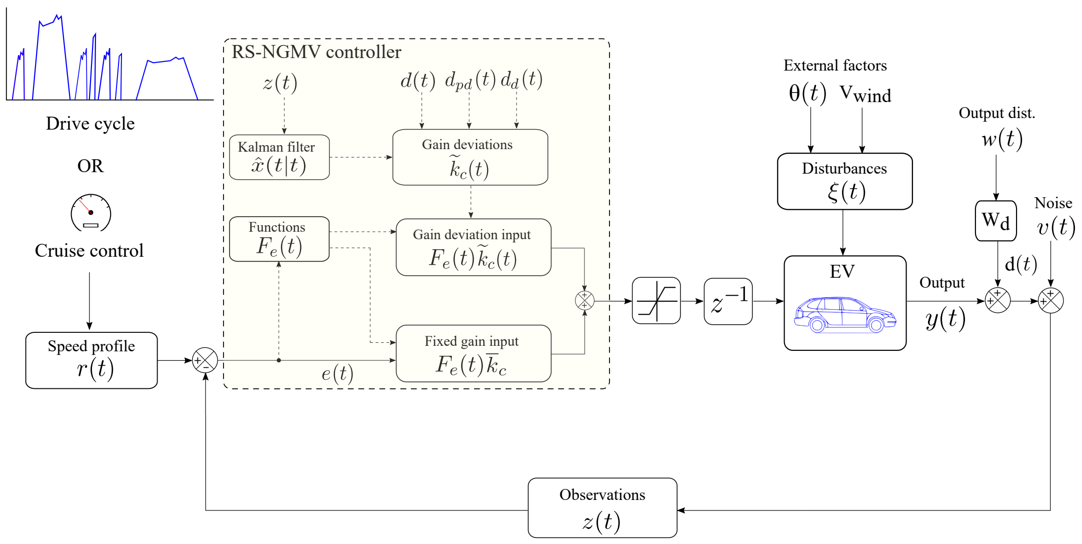


Figure 7.5: RS-NGMV control diagram.

and the feedback gains are denoted by the $\overline{k}_c$,

$$\overline{k}_c = k_{PI} = \begin{pmatrix} k_p \\ k_I \end{pmatrix} = \begin{pmatrix} 15 \\ 0.25 \end{pmatrix},$$

are fixed and will be combined with the NGMV optimized gains. The restricted structure controller results from the product of user pre-specified functions and the feedback gains, thus the RS-PI is defined by,

$$u(t) = F_e(t)\overline{k}_c. \tag{7.19}$$

7.2.2 RS-NGMV Controller

The parallel structure of the RS-NGMV is used as a PI controller with satisfactory results were available. The control strategy is implemented as in the Fig. 7.5 with the RS-NGMV control signal generalized as,

$$u(t) = F_e(t)k_c(t), \tag{7.20}$$

meaning $k_c(t) = \overline{k}_c + \widetilde{k}_c(t)$. The fixed PI gains $\overline{k}_c$, shown previously, are combined with the gain deviations $\widetilde{k}_c(t)$ suggested by the NGMV optimization,

$$\widetilde{k}_c(t) = -X_0(t)^{-1}(P_p(t)(d_{pd}(t+k) + C_{pt+k}\hat{x}(t+k|t)) + \psi_k(t)). \tag{7.21}$$

The RS-NGMV control parameters are given in Table 7.1 chosen by the weighting strategies in chapter 3, followed by trial-and-error modifications.

The objective of the controller is to track the reference speed profile suggested as a drive cycle or constant speed commmand for cruise control. Limitations on acceleration and braking is introduced with saturation blocks and the input non-linearity is considered $W_{1k} = I$.

The output disturbance is filtered by,

$$W_d(z^{-1}) = \frac{0.1z^{-1}}{1 - 0.98z^{-1}}, \tag{7.22}$$

Table 7.1: Controller parameters

Parameter	**Symbol**	**Value**
Dynamic Error Weighting	λ_p	1
Cost-Function Error Weighting	P_c	$1000\frac{1-0.97z^{-1}}{1-z^{-1}}$
Cost-Function Control Weighting	F_{ck}	$0.001\frac{1-z^{-1}}{1-0.1z^{-1}}$
Control Input Weighting	λ_u	0.0001
Gain Deviation Weighting	λ_k	diag(1, 0.1, 0.01)
Deviation Increments Weighting	λ_d	diag(100, 50, 0)

from which the state-space disturbance model is obtained and incorporated into the augmented system. This was not needed in previous case studies because the disturbances did not seem to interfere with the performance, however, for this case study it has been a crucial step. This is another advantage of the RS-NGMV approach in utilizing information which is the known disturbance in this case.

7.3 Simulation Results

The simulation results will be demonstrated after specifying the EV parameters.

7.3.1 The EV Parameters

The parameters used for the vehicle model is a blend of [104] and [109] which uses the speculated specifications of Chevy Volt, a passenger EV. They are shown in Table 7.2. The sampling time is chosen as $T_s = 1s$ for the simulations.

Table 7.2: EV parameters

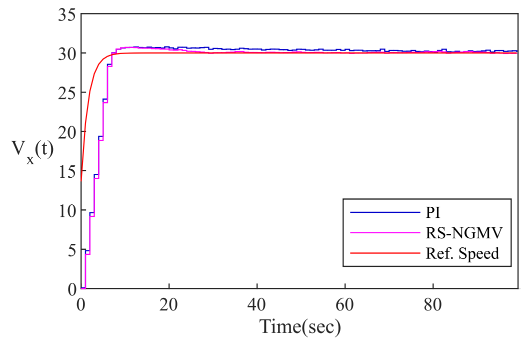
Parameter	**Symbol**	**Value**
Battery OCV	V_{oc}	$340V$
Internal Resistance	R_i	0.1Ω
Initial Battery SOC	z_0	0.95
Accessory Load	P_{aux}	$600W$
Energy Capacity	Q	$12.6kWh$
Gravity	g	$9.81m/s^2$
Vehicle Mass	m	$2200kgs$
Wheel Radius	r_w	$0.34m$
Rolling Resistance Coefficient	C_r	0.01
EV Frontal Area	A	$1.84m^2$
Aerodynamic Drag Coefficient	C_d	0.22
Air Density	ρ	$1.225kg/m^3$
Gear Ratio	G	3.55
Max. Motor Torque	T_{max}	$500Nm$
Rated Motor Torque	T_{rated}	$300Nm$
Max. Motor Speed	ω_{max}	$524rad/s$
Rated Motor Speed	ω_{rated}	$834rad/s$
Motor Constant	c	1.0472
Motor Constant	k_i	0.01
Motor Constant	k_c	0.12
Motor Constant	k_w	$1.2e-05$

7.3.2 Cruise-Control Results

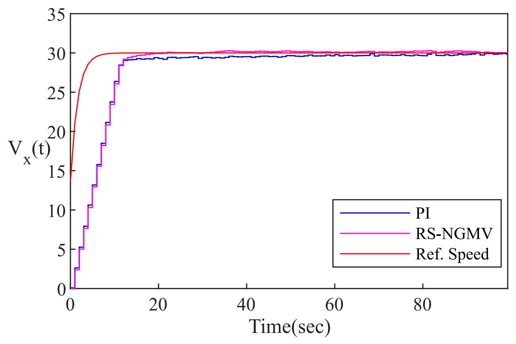
In this scenario, the reference speed is constant as in cruise-control applications. The reference signal is modelled as 1st-order transfer function to represent a smooth acceleration from $V_x(t) = 0$ to $V_x(t) = 30$mph by imitating the acceleration characteristics of market EVs. The Fig. 7.6 shows that the RS-NGMV can handle the disturbance resulting from the road grade and that it reaches steady-state faster than the PI in all cases. For these cases, we consider two road inclination profiles denoted by $\theta_1(t)$ and $\theta_2(t)$. $\theta_1(t)$ is displayed in part (e) of the figure and is applied on the tracking results for parts (a), (b) and (c).

In every simulation, noises are also considered hence results also display the stochastic control performance. For example in part (a), the road is flat ($\theta_1(t) = 0$) and it seen that even in this case the stochastic noises have a negative impact on the PI controller whereas their effect on the RS-NGMV is not significant. In part (b) the road grade is constant at $\theta_1(t) = 0.1$ radians which refers to a 10% inclination and the impact on both controllers are visible. The RS-NGMV manages to keep the traction and reach the steady-state quickly. For the part (c) the inclination starts from zero and increases to 0.05 radians. This can be considered as the vehicle ascending and results are similar to before for the RS-NGMV. However, it is seen that the variations degrade the performance of PI into a more oscillatory response. To demonstrate this more clearly, the $\theta_2(t)$ in part (f) is implemented as a time-varying road grade profile $\theta \simeq 0.1 sin(2ft)$ where $f = 0.005$Hz. The 0.1 radians refers to a 10% inclination and the sine wave enables a hilly route for the EV. The time horizon is extended as well. It is applied for the tracking case in part (d) and highlights the impact of variations.

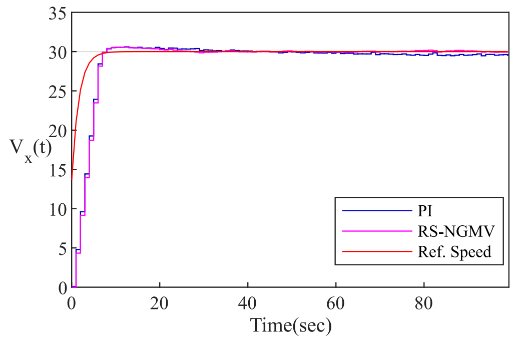
Assessing the overall control performance in the simulations, it may appear as the advanced RS-NGMV control does not improve too much against the PI control. The PI controller in this application does perform very well indeed on its own unless there are disturbances involved. However, when there are



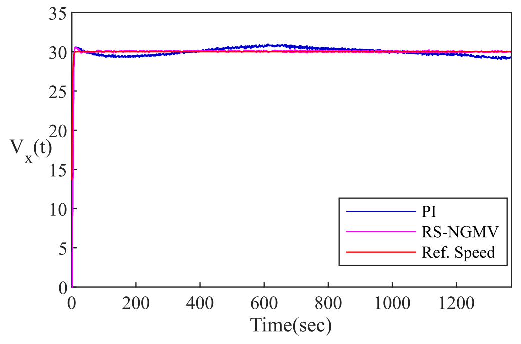
(a) Tracking for $\theta_1(t) = 0$



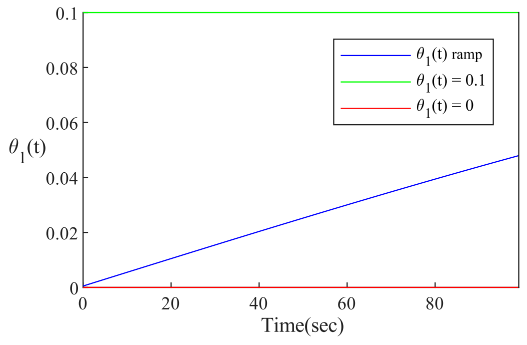
(b) Tracking for constant $\theta_1(t) = 0.1$



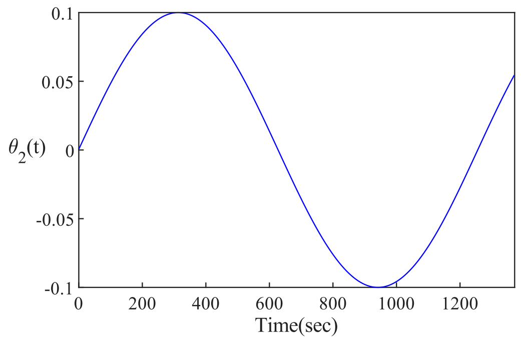
(c) Tracking for $\theta_1(t)$ ramp



(d) Tracking for $\theta_2(t)$

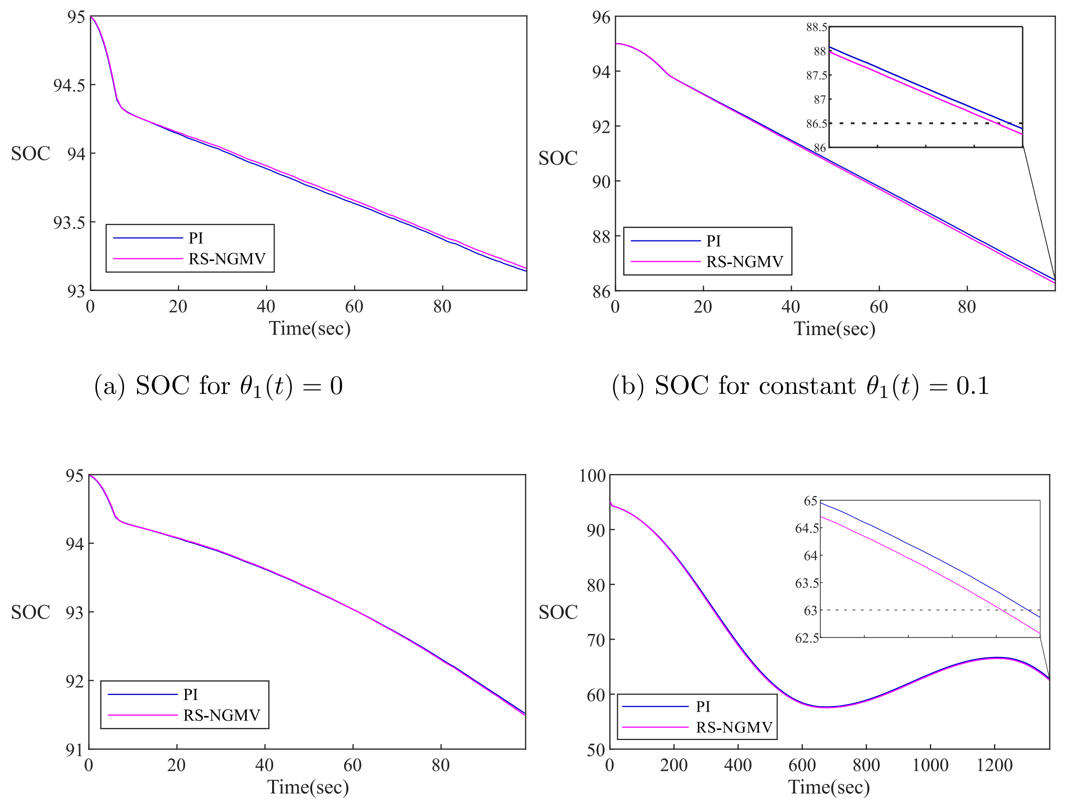


(e) $\theta_1(t)$ road grades



(f) $\theta_2(t)$ road grade

Figure 7.6: EV cruise-control tracking results.



(a) SOC for $\theta_1(t) = 0$

(b) SOC for constant $\theta_1(t) = 0.1$

(c) SOC for varying $\theta_1(t)$

(d) SOC for $\theta_2(t)$

Figure 7.7: EV cruise-control battery SOC results.

disturbances, the PI's performance reduces and the advantage of using the RS-NGMV gets stronger. In the real-world applications, it is also expected that even the small speed variations could impact the passenger comfort quite badly thus keeping the drive as smooth as possible still would improve the quality greatly. Furthermore, the purpose of RS-NGMV is to provide advanced NGMV control in low-order structures (such as PI, PID, PD or general z-transfer function forms) that classically trained staff are more familiar with. They are two different controllers and the RS-NGMV does not claim to be better or an alternative.

Battery SOC Results

Here are the battery SOC results from the previous runs are given in Fig. 7.7.

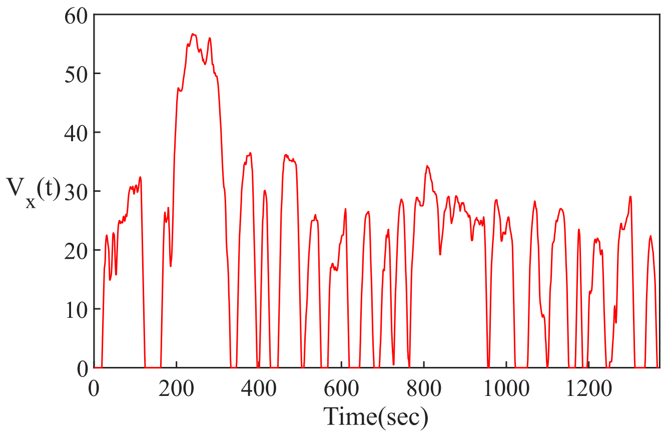
The initial SOC value is always set to $z(0) = 95\%$. The immediate observation is that the SOC values are higher for both controllers in the first three plots as these cases where only run for 100 seconds. It is also seen that the acceleration in the beginning consumes from the battery more compared to the constant speed as one would expect a similar case for conventional vehicles and fuel consumption. When the road grade $\theta(t)$ is zero the RS-NGMV keeps a higher SOC level in the end. However, for the last three plots in (b),(c) and (d) it is seen that the superior tracking performance of the RS-NGMV comes with a cost at the SOC. The difference is very minimal though being only approximately $z_{RS}(\infty) - z_{PI}(\infty) \leq 0.05$. This is because of the control efforts of RS-NGMV increasing due to the disturbance factor from the road grade. For part (d), it is also shown that the regenerative braking manages to charge the vehicle during the decelerations or descends.

7.3.3 Drive-Cycle Results

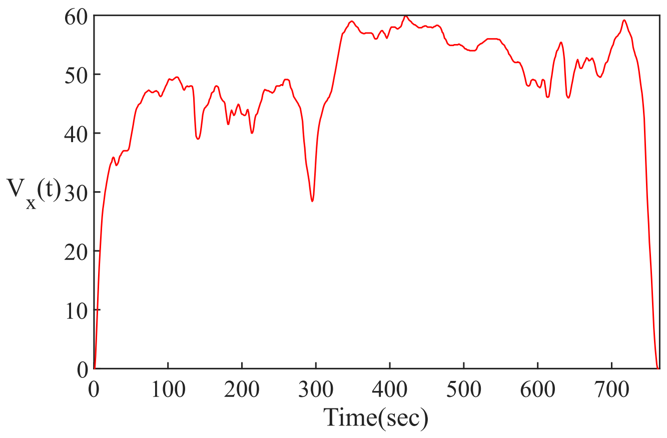
Having concluded the cruise-control case, the well-known EV drive-cycles will now be implemented as the reference speed profile. The first set of simulation results use the Urban Dynamometer Driving Schedule (UDDS) for the reference speed profile to study the EV in city driving conditions. The second set of results use the Highway Fuel Economy Test (HWFET) for the speed profile for highway driving where accelerations and decelerations are less aggressive.

For HWFET results the road grade $\theta_2(t)$ was used whereas in UDDS runs it was altered as $\theta_3(t)$ for the stops the EV comes at unlike the HWFET or cruise-control case where stops were not included. These signals are demonstrated in Fig. 7.8. The UDDS runs for 1369 seconds and the HWFET for 765 seconds.

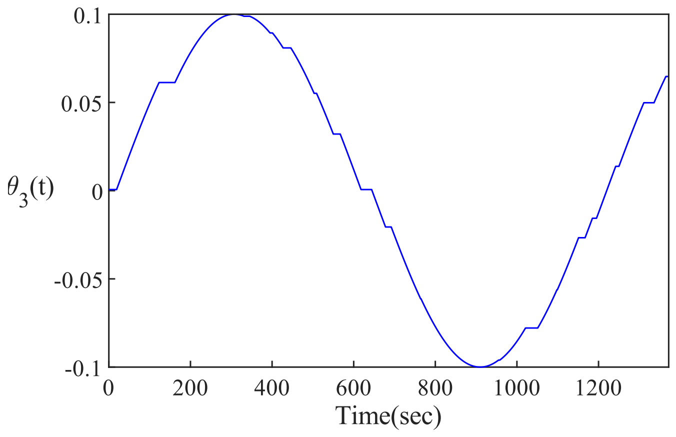
The tracking results for the two drive-cycles are plot in Fig. 7.9. It is observed that both controller show overall success in the tracking of many different velocities demanded by the drive-cycle. When looked closely, it is noticed that their



(a) UDDS drive-cycle.



(b) HWFET drive-cycle.



(c) $\theta_3(t)$ road grade

Figure 7.8: EV drive-cycles and $\theta(t)$ profiles.

performances differ. Except rare overshoots, the RS-NGMV shows better tracking performance for the UDDS drive-cycle. It tracks the reference very closely. The PI shows overshooting for many occasions and is unable to reach steady-state at vehicle stops and for some even go below the zero. For the HWFET in part (b), the PI in the first half is undershooting and mostly overshoot for the second half. The RS-NGMV has higher accuracy one more time.

Since, the drive-cycles are run for a very long simulation time, the controller performances might be difficult to notice without zooming in. Therefore, the tracking results that have just been presented are repeated in Fig. 7.10 and Fig. 7.11 by being split into halves with certain time windows zoomed-in and included in the main figures. They present the UDDS and HWFET tracking results, respectively.

To further support the hypothesis and for the sake of clarity, the RMS values of the tracking errors are also evaluated using,

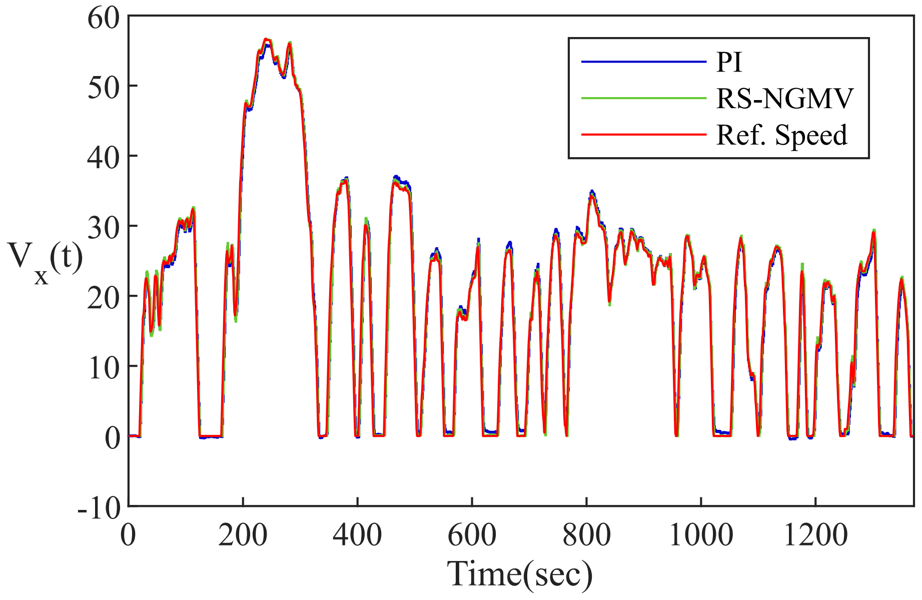
$$RMS(e(t)) = \sqrt{\frac{1}{T}\int_{t-T}^{t} e(\tau)^2 d\tau} \tag{7.23}$$

where T is $1/f$. The RMS values are given for UDDS and HWFET in Fig. 7.12. The RMS analysis verifies the controller tracking performance results.

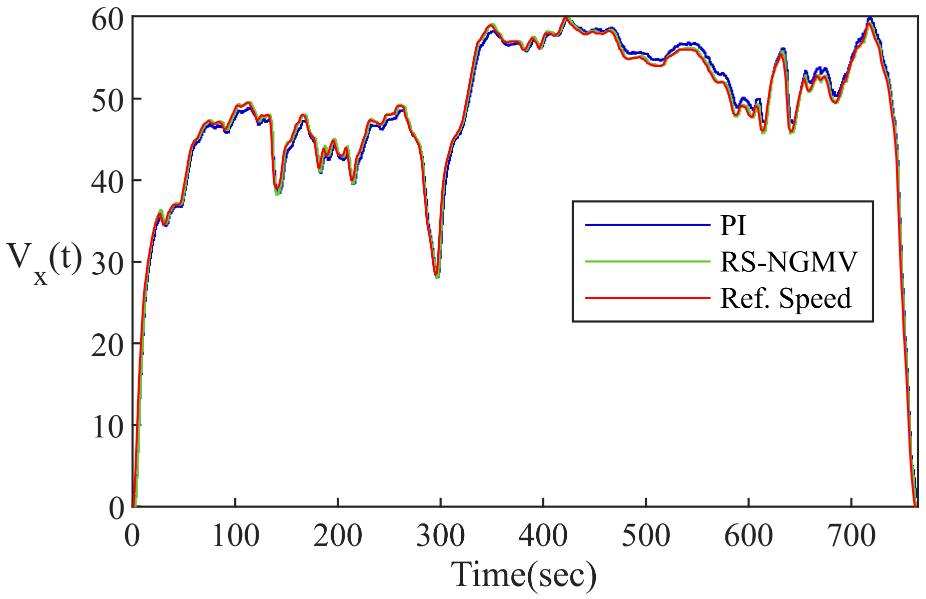
Battery SOC Results

Finally, Fig. 7.13 demonstrates the battery SOC levels for the drive-cycle runs.

For both cases, the SOC level for the RS-NGMV ends up slightly lower but it is almost negligible. It is hard to notice for the HWFET but a bit more visible for the UDDS. This is kind of expected, since the UDDS drive-cycle includes so many abrupt accelerations and decelerations. It also lasts longer there the road grade profile $\theta(t)$ that represents a hilly route is driven even further. It means more ascending and descending events are happening. Therefore, the control

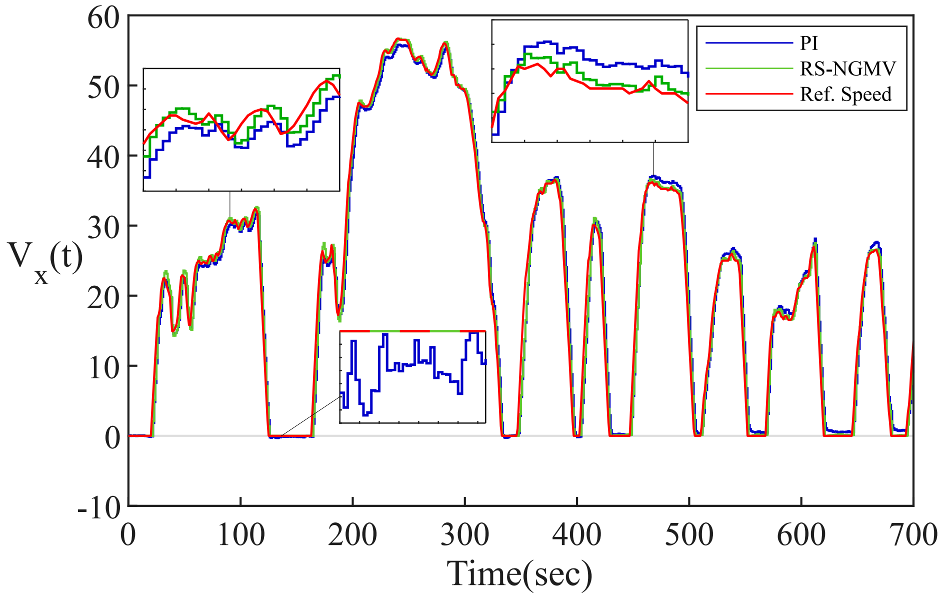


(a) UDDS drive-cycle.

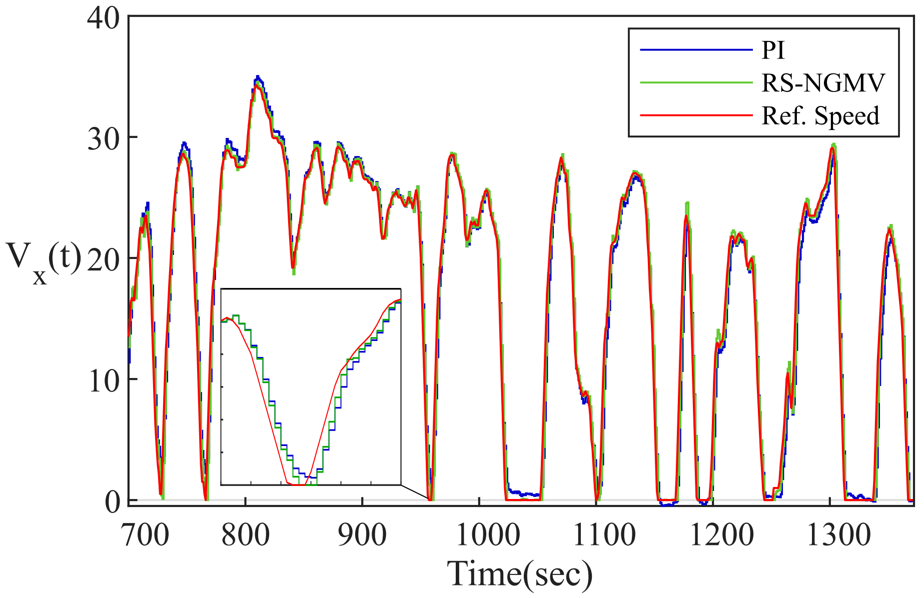


(b) HWFET drive-cycle.

Figure 7.9: UDDS and HWFET tracking results.

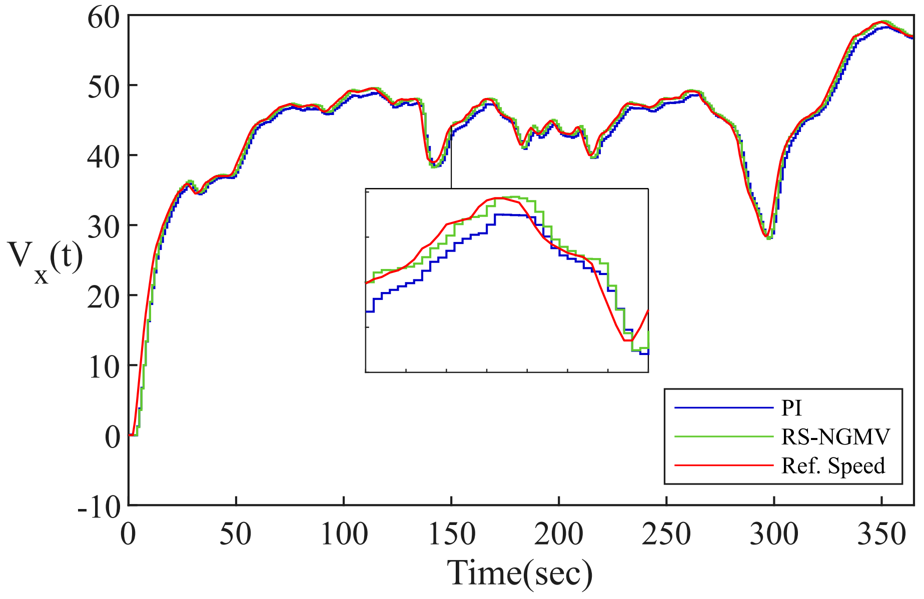


(a) UDDS drive-cycle first half.

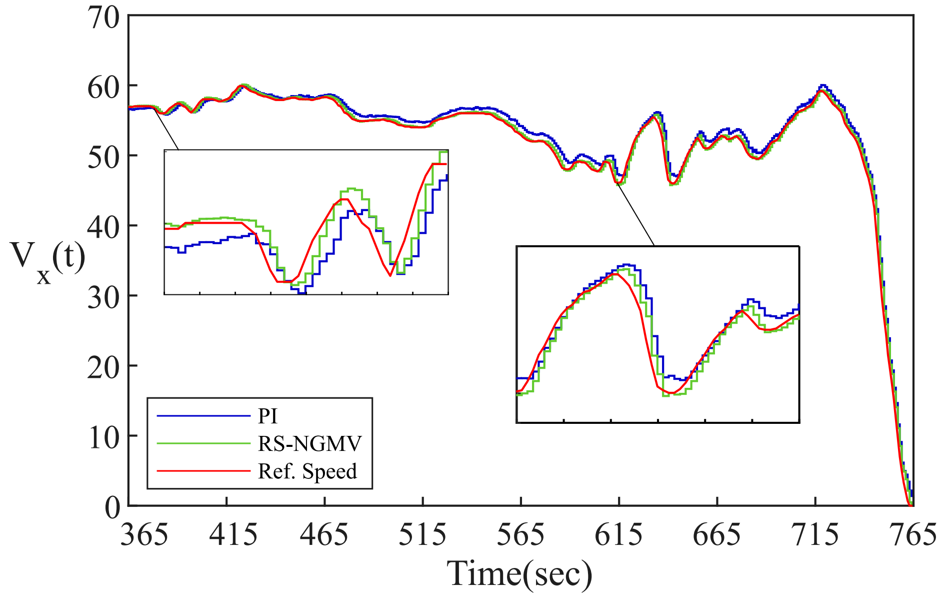


(b) UDDS drive-cycle second half.

Figure 7.10: UDDS drive-cycle split into half.

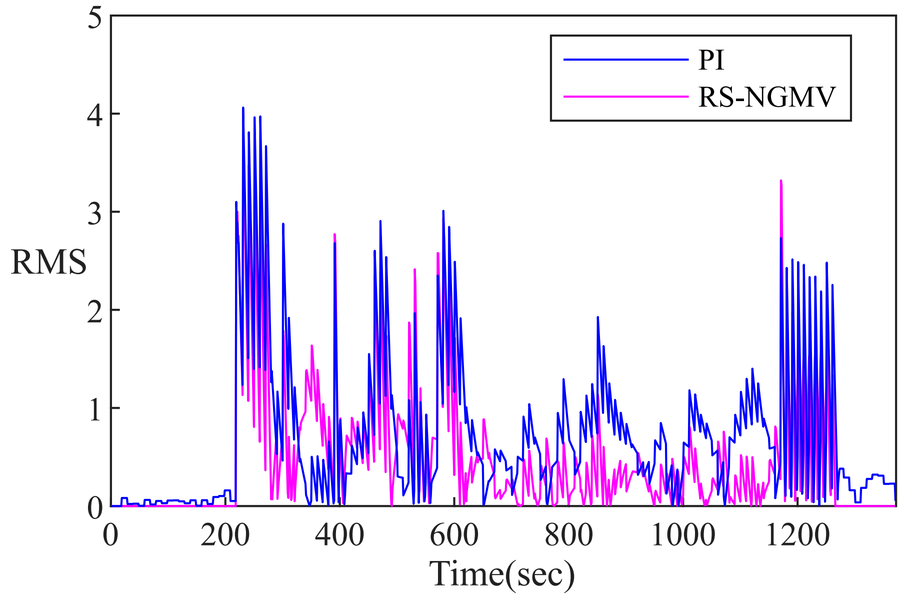


(a) HWFET drive-cycle first half.

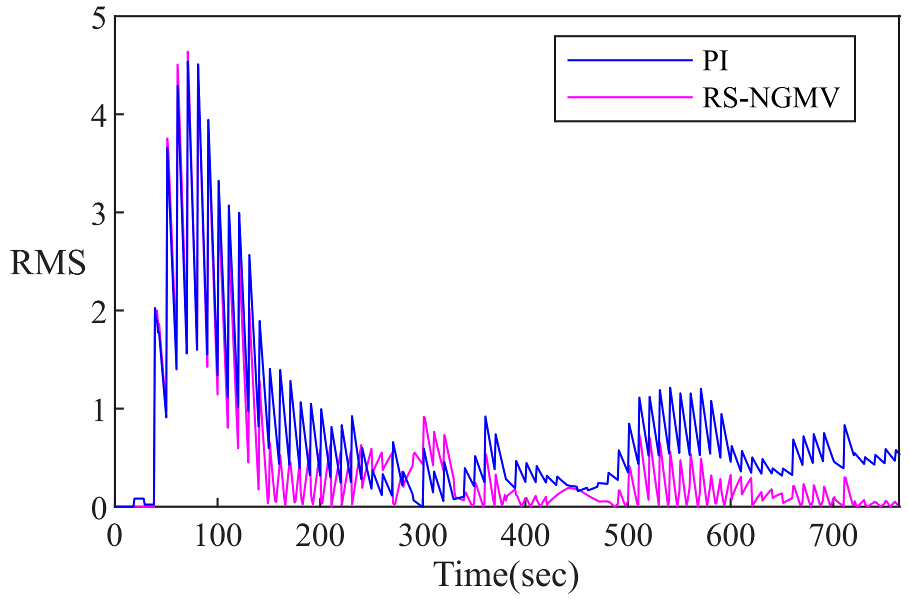


(b) HWFET drive-cycle second half.

Figure 7.11: HWFET drive-cycle split into half.

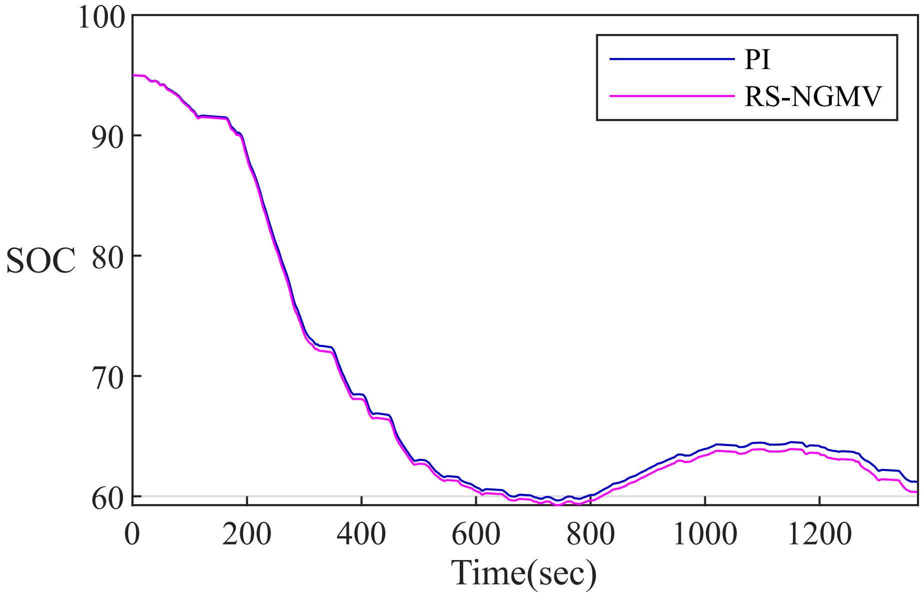


(a) UDDS tracking error RMS.

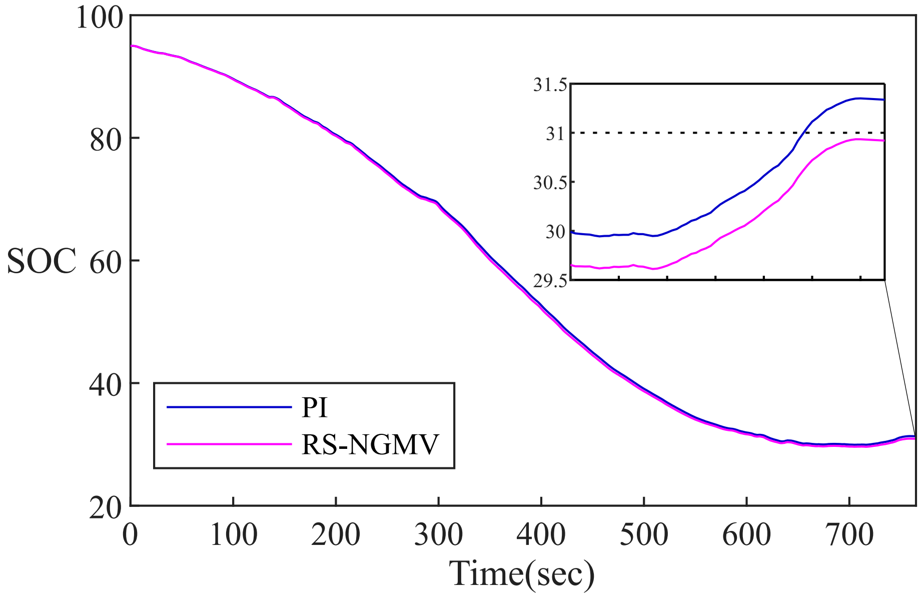


(b) HWFET tracking error RMS.

Figure 7.12: UDDS and HWFET tracking error RMS value plots.



(a) UDDS drive-cycle.



(b) HWFET drive-cycle.

Figure 7.13: UDDS and HWFET Tracking Results.

effort increases in response to these variations.

Range Estimations

The EV range estimation means roughly the number of miles the vehicle is estimated to keep travelling and is calculated at the beginning of a journey for this study. Recall, that the position of the vehicle is related to the velocity by $q = \int V_x dt$. In discrete-time, the position is calculated/updated using,

$$\Delta q = q(t) - q(t-1) = V_x T_s, \tag{7.24}$$

where Δq denotes the change in position between the time instants t and $t-1$. The estimated EV range is extrapolated from the driving cycle information, that is, the total distance travelled and the depletion of the battery SOC [122]. With the battery SOC limits also taken into account, the range is calculated by,

$$\hat{q} = q_{total} \frac{z_{max} - z_{min}}{z(0) - z(\infty)}. \tag{7.25}$$

where $z_{max} = z_0 = 0.95$. The lower SOC limit is chosen as $z_{min} = 0.05$. On the lower limit, the value is entirely up to the manufacturer set up at the factory. For example, the Chevrolet Volt's lower limit is given as $z_{min} = 0.3$ according to [120]. When long-term use is taken into account, this is a safe number and will protect the battery pack from ageing rapidly. On the other hand, for the study in this thesis, EV tracking performance is the main focus. The following results are based on the UDDS and HWFET cycles introduced earlier.

The EV range estimations for the UDDS drive-cycle with are given by,

$$\left(\hat{q}_{RS-NGMV}, \quad \hat{q}_{PI} \right) = \left(40.09mi, \quad 41.79mi \right).$$

for zero road grade $\theta(t) = 0$. When the varying road grade profile $\theta_2(t)$ is selected, the EV range estimation values fall drastically from the previous case,

$$\left(\hat{q}_{RS-NGMV},\quad \hat{q}_{PI}\right) = \left(13.43mi,\quad 13.61mi\right).$$

As for the HWFET drive-cycle, the estimations for the two controllers are,

$$\left(\hat{q}_{RS-NGMV},\quad \hat{q}_{PI}\right) = \left(47.28mi,\quad 47.94mi\right).$$

with $\theta(t) = 0$. For $\theta_2(t)$ road grade profile, the EV range estimations are,

$$\left(\hat{q}_{RS-NGMV},\quad \hat{q}_{PI}\right) = \left(14.67mi,\quad 14.75mi\right).$$

Note that, the battery pack chosen for this study has a very small energy capacity of $Q = 12.6kWh$. Most of the EV parameters chosen in this thesis are based on the speculated values of the Chevrolet Volt as our reference work [109] used them prior to the vehicle's release. The first generation models of this vehicle equipped battery packs with approximately 30% higher energy capacity [120]. However, the results here are still similar to those of the actual vehicle's.

The EV range estimations of this section are related and support the battery SOC results introduced earlier. The reason RS-NGMV shows a very small difference in the above results is because its final SOC value $z(\infty)$ is slightly lower than that of the PI which is caused by the control efforts in keeping up the vehicle performance.

These results show that the impact of the road grade is significant on the vehicle range and highlight the importance of the battery SOC information in calculating it which is a critical factor in planning a journey for the driver.

7.4 Closing Remarks

The results have shown the affect of the disturbances on the tracking performance. It is seen that the impact is even stronger when the disturbances are time-varying. This highlights the importance of using advanced control strategies to deal with harsh real-world conditions that are often changing dynamically. In this case study, it is verified that the RS-NGMV controller can sustain the desired response with minimal losses at the presence of heavy disturbances as well.

It was assumed that the initial battery SOC starts at 95%. However, in practical scenarios it cannot be expected for the SOC to be at this level every single time a journey begins. The user might stop for the day or take a break to start another without charging the battery [122]. Thus, it is very important to estimate the initial value of the SOC to approximate the driving range properly which is crucial when planning a journey [115]. Future work will consider the SOC estimation and a more advanced lithium-ion battery ECM model including using RS-NGMV for its charging. Recent studies have also investigated the performance of MPC [116 − 117] in charging of EV batteries. However, it is noticed that in these studies the control horizon does not impact the performance much. Therefore, RS-NGMV could be studied as an alternative being a less computationally expensive solution.

Of course, instead of advanced ECM approaches, the physics based modelling of EV batteries could also be considered for these future works. When battery ageing aspects are taken into account, the physics based modelling approach looks quite promising when designing controllers for long-term use. In addition, the SOC results hint a link between the speed tracking performance and the battery response which is another topic to investigate for the future research. A modified RS-NGMV cost function for a compromise between tracking and battery which considers different operating conditions might be a good idea. The last bit could also be linked to the Scheduled RS-NGMV approach in the next chapter.

In this chapter, another case study has been completed to verify the usefulness of the RS-NGMV method. To achieve this, the longitudinal speed tracking problem for which the PI technique has been the most common control solution, has been chosen. This has enabled the implementation of the restricted structure approach and provided a baseline controller to start building on. The speed tracking control problem has been applied on an electric vehicle which belongs to one of the most demanded research topics today. In addition to the RS-NGMV implementation of chapter 5, the state-space qLPV model has also included input disturbances but it has been shown that the RS-NGMV is capable of handling them.

Chapter 8

Scheduled RS-NGMV Control

"*The true art of memory is the art of attention.*"

Samuel Johnson, Selected Essays.

This chapter introduces the Scheduled RS-NGMV controller. The approach takes on the RS-NGMV algorithm presented in chapter 3 for its basis. The objective is to create an efficient methodology to find out the best possible RS-NGMV gains for different operating conditions or set-points without too much manual effort and store these gains. Then have the controller assign the stored values rapidly, next time the same conditions are encountered hence becoming the Scheduled RS-NGMV controller.

The amount of time for controller calibration covers a substantial part of design and verification process. A typical example is in the automotive industry where engine controls are required to cover a wide range of non-linear operating conditions and must be tuned accordingly. Therefore, it is high-cost task for the companies and only gets higher if more resources such as man hours of testing and tuning become necessary. If successful, the Scheduled RS-NGMV methodology is expected to save an important amount of time since the engineers will not be needed to test the engine at every operating point having the recommended gains obtained from the simulations. Some results are already available for this idea and will be presented in the following section.

8.1 SI Engine Application Example

The results from the initial analysis on the Scheduled RS-NGMV idea will be given here. The Scheduled RS-NGMV studies consider an SI engine model for which the control problem is reference engine speed tracking. The equations of the SI engine and its model are explained next.

8.1.1 SI Engine Model

The reference engine model we use is obtained from a built-in Matlab/Simulink demo "sldemo_enginewc" that results from the original theoretical work in [112]. The linearised plant model and its output are given by,

$$\dot{x}(t) = \begin{pmatrix} 0.8967 & -0.00036333 \\ 8.4576 & 1.0053 \end{pmatrix} \begin{pmatrix} P_{im}(t) \\ N(t) \end{pmatrix} + \begin{pmatrix} 0.0096241 \\ 0.043672 \end{pmatrix} \theta(t). \tag{8.1}$$

$$y(t) = \begin{pmatrix} 0 & 9.5493 \end{pmatrix} \begin{pmatrix} P_{im}(t) \\ N(t) \end{pmatrix}. \tag{8.2}$$

where states $P_{im}(bar)$ and $N(rad/s)$ denotes the intake manifold pressure and the engine speed, respectively. The output N is in RPM units and the θ represents the throttle angle which is the input signal.

For the model above, the engine model is linearised at the operating points specified at $x_0 = \begin{pmatrix} 0.543bar \\ 209rad/s \end{pmatrix}$ and $u_0 = 8.98^\circ$ for the nominal engine speed $N = 2000$RPM. Similarly, the other operating conditions are specified using Matlab's findop function. The model is discretized similarly to the previous implementations with the sampling-time $T_s = 0.0150s$.

8.1.2 Simulation Studies

The simulation studies first consider the design of a RS-NGMV controller following similar steps to the case studies before and later implement the Scheduled RS-NGMV steps.

Reference Engine Speed Control

The traditional PI control law used widely for this problem is given in,

$$u(t) = C_0(z^{-1})e(t) = k_P e(t) + k_I T_s \frac{z}{z-1} e(t), \tag{8.3}$$

and the reference engine speed tracking error is defined by,

$$e(t) = N_{ref}(t) - N(t). \tag{8.4}$$

The PI control law in 8.3 is parametrized using the expression below and will serve as the fixed gain component of the parallel form restricted structure controller,

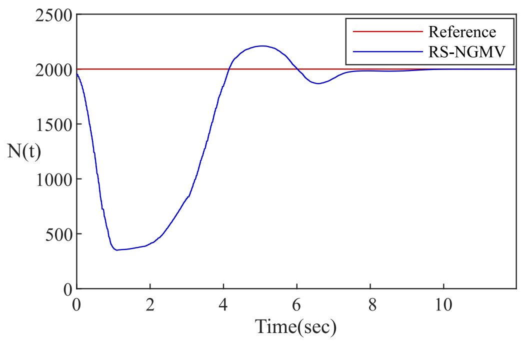
$$u(t) = F_e(t)\overline{k}_c, \tag{8.5}$$

where the function terms $F_e(t)$ have,

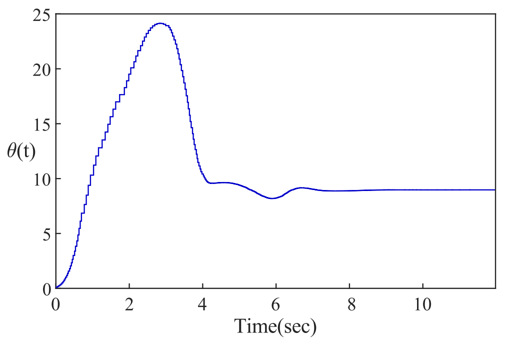
$$\begin{pmatrix} f_1(z^{-1}) \\ f_2(z^{-1}) \end{pmatrix} = \begin{pmatrix} 1 \\ 1/(1-z^{-1}) \end{pmatrix}, \tag{8.6}$$

and the parallel structure uses the fixed PI gains chosen as,

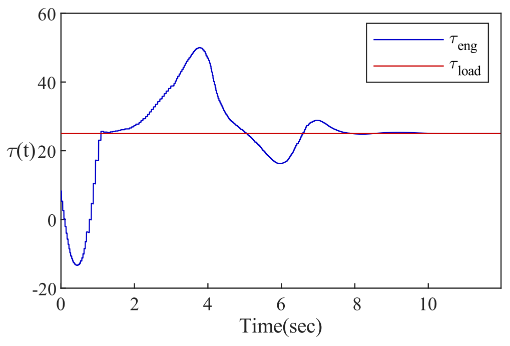
$$\overline{k}_c = k_{PI} = \begin{pmatrix} k_p \\ k_I \end{pmatrix} = \begin{pmatrix} 0.045 \\ 0.145 \end{pmatrix}. \tag{8.7}$$



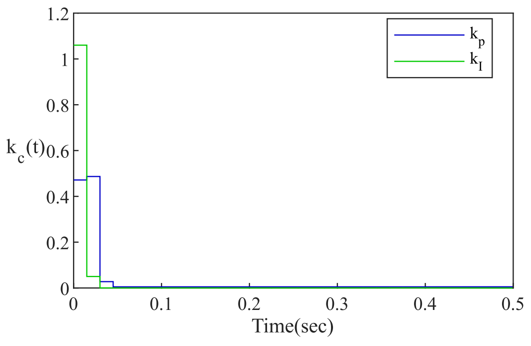
(a) Engine speed



(b) Engine torque vs. load torque



(c) Control input



(d) RS-NGMV gains $k_c(t)$

Figure 8.1: RS-NGMV reference engine speed tracking.

For the gain deviation component of the restricted structure controller, the optimized feedback gains are calculated with,

$$\widetilde{k}_c(t) = -X_0(t)^{-1}(P_p(t)(d_{pd}(t+k) + C_{pt+k}\hat{x}(t+k|t)) + \psi_k(t)). \tag{8.8}$$

The engine speed tracking results of the RS-NGMV controller are shown in Fig. 8.1 for $N_{ref} = 2000$RPM.

The plot (a) shows that the controller is able to track the reference engine speed and the plot (b) shows that the engine torque is kept constant. The part (c) visualizes the throttle angle as the control input applied to reach the desired engine speed. In the part (d), the RS-NGMV gains $k_c(t)$ as a combination of

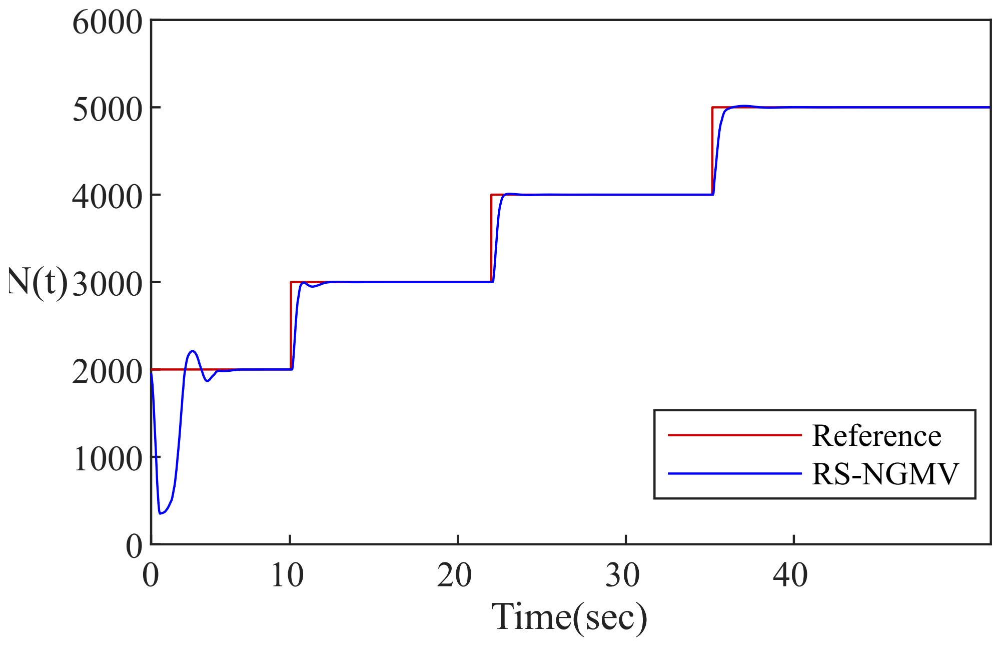


Figure 8.2: RS-NGMV reference engine speed profile tracking.

fixed and deviating components are plotted illustrating how they are updated until the steady-state has been reached.

Storage of Calculated RS-NGMV Gains

The next step is to extend the case above to an engine speed profile stated as,

$$N_{ref} = \Big(2000, \quad 3000, \quad 4000 \quad 5000\Big),$$

for which the set-points change at time steps below,

$$T_{step} = \Big(0, \quad 10, \quad 20 \quad 30\Big),$$

where it is assumed that the engine is speeding up. The tracking results for this case is shown in Fig. 8.2. Note that the time steps are not entirely exact, this is because buffers are used within the reference generator.

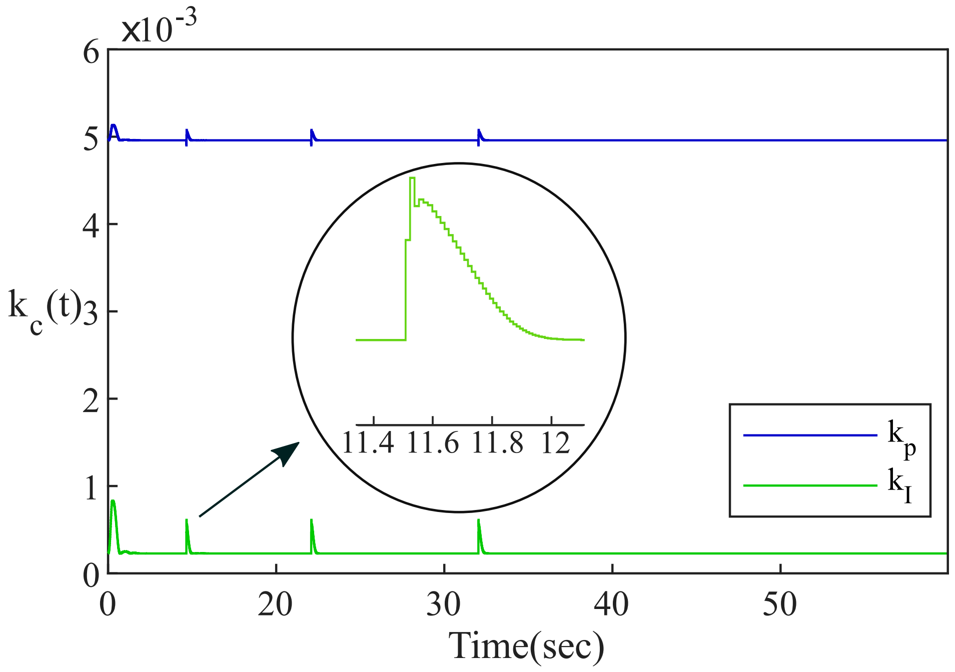


Figure 8.3: RS-NGMV reference engine speed profile gains.

The next task is collecting the gains $k_c(t)$ for the reference speeds and matching them in 1-D look-up tables. The RS-NGMV optimized feedback gains $k_c(t)$ are shown in Fig. 8.3. It is seen from the figure that the gains are updated during the set-point changes. The gains are collected around these update events,

$$T_{collect} = \begin{pmatrix} 0.3, & 11.5, & 21.5 & 31.5 \end{pmatrix}.$$

as visualized in the zoomed out portion of the figure. It is not yet discovered how to calculate which time-instant the gains should be collected. However, we do know that values chosen at transitions work a lot better than values chosen at steady-state which are often unstable. A transient analysis using classical control techniques is in order. For now, values from the transition area are used in trial and error fashion.

After the gains are recorded in the look-up table, they are ready for the Scheduled RS-NGMV controller.

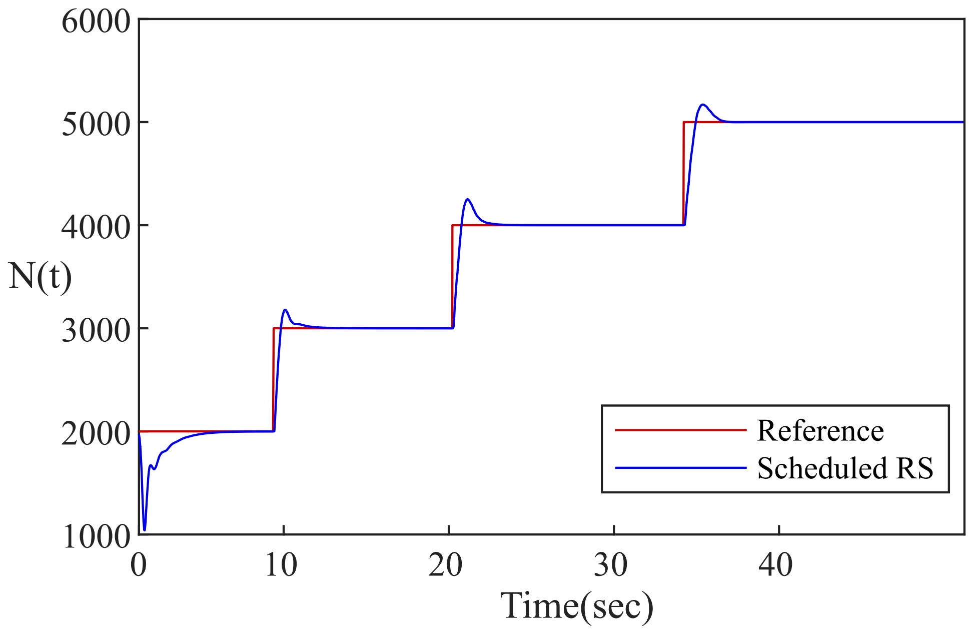


Figure 8.4: Scheduled RS reference engine speed profile tracking.

Real-time Use of Stored RS-NGMV Gains

In the Simulink model the original RS-NGMV block used for capturing the gain values is switched with the scheduled RS control block. The controller will not need to calculate anything but simply will use the recorded gains from the previous run. The results from running the scheduled RS controller are presented in Fig. 8.4.

Firstly, the scheduled RS simulation run was much faster due to not needing optimizations. This can be an advantage for this type of controller in the actual applications. Secondly, when compared to the RS-NGMV results in Fig. 8.2, it is visible that the reference tracking is successful and the engine speed reaches the steady-state value for the same amounts of time. However, there are slightly larger overshoots for the last three set-points. On the contrary to it, the overshoot performance looks better than the first run for the first set-point $N_{ref} = 2000$RPM.

8.2 Closing Remarks

In this chapter a Spark Ignition (SI) has been used as the application for the initial results from the Scheduled RS-NGMV research. The Scheduled RS-NGMV is promising to solve a substantial engineering problem where several operating conditions must be considered and the calibration task is demanding countless man hours. If generalized as a method, it could be used in other applications including EVs as well (recall the battery SOC vs. tracking performance remark of chapter 7).

The results has shown that the approach of choosing the gains around the set-point change events can get close or even better but definitely needs be systematized with either some transient analysis or another technique. Classical control techniques can be considered for the transient analysis and/or machine learning techniques could be used after gathering a lot of data by using drive cycles or trimming a large set of operating points instead of having a few set-points. The analysis of the data with machine learning techniques can find out the best possible values.

In summary, this chapter has shown the potential of the Scheduled RS-NGMV. Even in its simplest way, the principle of storing the RS-NGMV gains and applying them later without the need to calculate complex control computations each time, has improved the speed of the application.

Chapter 9

Conclusions and Future Work

This thesis has presented the RS-NGMV algorithm with special focus on the control of LPV or SD systems. For these systems, theoretical results were found and verified with cases studies. Features of the RS-NGMV algorithm were investigated which has mostly evolved around the question of how the RS-NGMV controller can deal with parameter-variations, set-point changes, disturbances and external information. It was shown how the method is capable of adapting to these factors. The low order restricted structure were shown to be naturally robust as well.

With the parallel form for the restricted structure, it was also demonstrated how methods like PI/PID can be used in the RS-NGMV design and highlighting the benefit of reaching the majority of the classically trained engineers in the industry. While the systems focus was set on LPV or SD systems, a special solution of the RS-NGMV capable of handling systems with input non-linearities or black terms were also derived.

Towards the goals of this thesis, the introduction chapter has set the scene for the developments in non-linear, optimal and predictive control field. Next, background theory and literature was presented to explain the foundations of the RS-NGMV technique. The LPVKF was formulated as well as the augmented state-space system RS-NGMV uses. In chapter 3, the RS-NGMV solutions that are mentioned above were formulated. The preview extension for RS-NGMV algorithm was considered in chapter 4 and resulted in the SDRE approach to the preview control problem. SDRE and LPV-RE control laws were derived. Another extension idea which is the Scheduled RS-NGMV has been proposed in chapter 8 with case study results on a linearised SI engine model. Regarding the preview extension, the RS-NPGMV is proposed in the Appendix A. Matlab/Simulink codes, diagrams and implementation tips are given in Appendix B through F.

The algorithmic extensions that this thesis has considered, contribute to the latest directions in the optimal/predictive theory and not yet fully researched questions such as the preview control. Meanwhile today's engineering problems have also been at the center of these attempts. In chapters 5, 6 and 7, cases studies use a robotic manipulator, autonomous vehicle and electric vehicle respectively. Especially, the autonomous and electric vehicles are recently very popular topics researchers work on.

Some research questions were mentioned in the closing remarks of some chapters. For instance, chapter 4 suggests designing RS-NLQG or RS-NPGMV methods for the preview control problem and stability analysis using satisficing, chapter 6 suggests an investigation on the optimum preview length and chapter 7 leads the path to BMS algorithm designs and using RS-NGMV for EV battery charging case. Chapter 8 is a great case for involving machine learning and/or AI for the RS-NGMV control. Steps are already being taken to realize the last two ideas above.

There is still much to discover regarding the features of the RS-NGMV controller. The RS is a general z-transfer function, thus different types of low-order controllers could be the basis of a new RS-NGMV example. The control solution for the systems with non-linear input subsystems or black-box terms is also worth looking into. The compensation problem for a non-linear actuator can be studied by defining its non-linearity in the input subsystem and developing a RS-NGMV solution which has the low-order structure of a compensator.

There is also much to explore in the area of robustness. As a start, the Monte-Carlo simulation runs could be tried for the case studies in this thesis. In an earlier study, a robust NGMV controller was designed to be used in fault-monitoring [113]. The controller was designed to be less sensitive when uncertainties were present in the models. Similar study may be done on the RS-NGMV as well. Another work [114] also considers the robustness problem for the NGMV but from a totally different perspective. The paper presents a modified Kalman filter and resilient NGMV controller against a cyber attack.

The cyber security of industrial controllers became part of a huge debate in 2010 when a malicious computer worm called the Stuxnet was used to target SCADA and PLC systems at an Iranian plant. A decade after the incident, on the path of industry 4.0, motivations to develop networked smart machines integrated with the IoT have been creating a far greater demand for the security of systems. It would be interesting to consider the RS-NGMV problems from the IT and communications perspective as well, since the strength of the methods shown in our case studies indicate one thing in common and that is the importance of utilizing information to improve the controller performance and features.

Bibliography

[1] K. J. Åström, *Introduction to Stochastic Control Theory*, Academic Press, 1970.

[2] D.W. Clarke and R. Hastings-James, "Design of Digital Controllers for Randomly Disturbed Systems," *Proc. IEEE*, vol. 118, pp. 1503–1506, 1971.

[3] T. Shiino, K. Kawada, T. Yamamoto, M. Komichi, T. Nishioka, "Gimbals Control with the Camera for Aerial Photography in RC Helicopter," *Proc. Int. Conf. Control, Automation and Systems*, pp. 1135–1139, 2008.

[4] S. Camcioglu, Z. Zeybek, H. Hapoglu, M. Alpbaz and A. Akpinar, "Generalized Minimum Variance (GMV) Control in Waterborne Wastewater Treatment," *Chemical Engineering Transactions*, vol. 21, pp. 823–828, 2010.

[5] W.T.Chung and A.K.David, "Digital and Laboratory Implementation of a Generalised Minimum Variance Controller for an HVDC Link," *IEE Proc. Gener. Transm. Distrib.*, vol. 146, no 2, pp. 181–185, 1999.

[6] K. J. Åström and B. Wittenmark, *Computer Controlled Systems: Theory and Practice*, Prentice-Hall, Inc., Upper Saddle River, NJ, 3rd edition, 1997.

[7] P. Wellstead, M. Zarrop, *Self-Tuning Systems: Control and Signal Processing,* Wiley, 1991.

[8] D.W Clarke, P.J. Gawthrop, “Self-Tuning Controller,” *Proc. IEE*, vol. 122, pp. 929–934, 1975.

[9] M.J. Grimble, “Generalized Minimum Variance Control Law Revisited,” *Optimal Control Applications & Methods*, vol. 9, pp. 63–77, 1988.

[10] M.J. Grimble, *Industrial Control Systems Design,* Chichester: John Wiley, 2001.

[11] M. J. Grimble, “GMV Control of Non-Linear Multivariable Systems,” *UKACC Control Conference*, Bath, UK, 2004.

[12] M. J. Grimble, “Non-linear Generalized Minimum Variance Feedback, Feedforward and Tracking Control,” *Automatica*, vol.41, pp. 957-969, 2005.

[13] M. J. Grimble, “GMV Control of Non-linear Continuous-time Systems Including Common Delays and State-Space Models,” *International Journal of Control*, vol.80, pp. 150–165, 2007.

[14] M. J. Grimble and Y. Pang, “NGMV Control of State Dependent Multivariable Systems,”*IEEE CDC Conference*, pp. 1628-1633, 2007.

[15] Y. Pang, M.J. Grimble, “State Dependent NGMV Control of Delayed Piecewise Affine Systems,” *IEEE CDC Conference*, pp. 7192-7197, 2009.

[16] M. Grimble, P. Majecki, *Nonlinear Industrial Control Systems,* Springer, 1st edition, 2020.

[17] J. R. Cloutier, “State-Dependent Riccati Equation Techniques: An Overview,” American Control Conference, pp. 932–936, 1997.

[18] T. Cimen, “State-Dependent Riccati Equation (SDRE) Control: A Survey,” *17th IFAC World Congress*, pp. 3761-3775, 2008.

[19] C. Cebeci, M.J. Grimble, L. Recalde-Camacho and R. Katebi, "SDRE Preview Control for a LPV Modelled Autonomous Vehicle," *3rd IFAC Workshop on Linear Parameter-Varying Systems*, Eindhoven, The Netherlands, 2019.

[20] M. J. Grimble, P. Majecki, "Nonlinear Predictive GMV Control," *ACC*, pp. 1190-1195, 2008.

[21] M. J. Grimble, P. Majecki, "Polynomial Approach to Non-Linear Predictive Generalised Minimum Variance Control," *IET Control Theory and Applications*, pp. 411-424, 2009.

[22] M. J. Grimble, P. Majecki, "State-Space Approach to Non-linear Predictive Generalised Minimum Variance Control," *International Journal of Control*, vol. 83, pp. 1529-1547, 2010.

[23] M. J. Grimble, P. Majecki, "Non-Linear Predictive Generalised Minimum Variance State-Dependent Control," *IET Control Theory & Applications*, pp. 1-13, 2015.

[24] P. Savvidis, J. Wang, M. R. Katebi, M.J. Grimble, "NGMVC Scheme on Marine Surface Vessels Dynamic Positioning and Manoeuvring," *14th International Ship Control Systems Symposium*, 2009.

[25] P. Savvidis, M. J. Grimble, P. Majecki, Y. Pang, "Nonlinear Predictive Generalized Minimum Variance LPV Control of Wind Turbines," *5th IET International Conference on Renewable Power Generation*, 2016.

[26] P. Savvidis, "Nonlinear Control : an LPV Nonlinear Predictive Generalised Minimum Variance Perspective," PhD Thesis, Dept. of Electronic & Electrical Eng., University of Strathclyde, 2017.

[27] M.J. Grimble, P. Majecki and M.R. Katebi, “Extended NGMV Predictive Control of Quasi-LPV Systems,” *American Control Conference*, vol. 50, issue 1, pp. 4101–4107, 2017.

[28] M. Tomizuka, “Optimal Continuous Finite Preview Problem,” *IEEE Trans. Autom. Control*, pp. 362–365, 1975.

[29] A.S. Silveira, A.A.R. Coelho, “Generalised Minimum Variance Control State-Space Design,” *IET Control Theory and Applications*, vol. 5, pp. 1709–1715, 2011.

[30] S. Okada, S. Masuda, “Data-driven Linearly Parametrized Controller Design Based on Minimum Variance Evaluation,” *Asian Control Conference*, 2017.

[31] J.S. Shamma, M. Athans (1990), “Analysis of Gain Scheduled Control for Nonlinear Plants,” *IEEE Transactions on Automatic Control*, vol. 35, pp. 898–907, 1990.

[32] J.S. Shamma, M. Athans, “Guaranteed Properties of Gain Scheduled Control for Linear Parameter-Varying Plants,” *Automatica*, vol. 27, pp. 559–564, 1991.

[33] J.S. Shamma, “Gain Scheduling: Potential Hazards and Possible Remedies,” *IEEE Control Systems*, 1992.

[34] R. Toth, *Modeling and Identification of Linear Parameter-Varying Systems,* Springer-Verlag Heidelberg, 2010.

[35] A. P. White, G. Zhu, J. Choi, *Linear Parameter Varying Control for Engineering Applications,* Springer, 2013.

[36] O. Sename, P. Gaspar, J, Bokor, *Robust Control and Linear Parameter Varying Approaches: Application to Vehicle Dynamics,* Springer, 2013.

[37] J. Mohammadpour, C. W. Scherer, *Control of Linear Parameter Varying Systems with Applications,* Springer, 2012.

[38] C. Hoffmann, H. Werner, "A Survey of Linear Parameter-Varying Control Applications Validated by Experiments or High-Fidelity Simulations," *IEEE Transactions on Control Syst. Tech.*, vol. 23, no 2, pp. 416–433, 2015.

[39] C. Briat, *Linear Parameter Varying and Time-delay Systems: Analysis, Observation, Filtering & Control,* Advances in Delay and Dynamics, vol.3, 2015.

[40] J. B. Gary, "Linear Parameter-Varying Control And Its Application to Aerospace Systems," *ICAS*, 2002.

[41] M. Grimble, M. Johnson, *Optimal Control and Stochastic Estimation, Volume I and II,* John Wiley, London, 1988.

[42] M. A. Johnson, M. H. Moradi, *PID Control: New Identification and Design Methods,* 1st ed., Springer-Verlag London, 2005.

[43] M. H. Moradi, M. R. Katebi and M. A. Johnson, "Predictive PID control: A New Algorithm," *IECON'O1: The 27th Annual Conference of the IEEE Industrial Electronics Society*, vol. 1, pp. 764-769, 2001.

[44] M.J Grimble, "Restricted Structure LQG Optimal Control for Continuous-Time Systems," *IEE Proc. Pt D*, vol. 147, no.2, 2000.

[45] M.J Grimble, "Restricted Structure Control Loop Performance Assessment for State-Space Systems," *Proc. ACC*, vol. 2, pp. 1633-1638, 2002.

[46] M.J Grimble, "Restricted Structure Control Loop Performance Assessment for PID Controllers and State-Space Systems," *Asian J. Control*, vol. 5, no.1, pp. 39-57, 2003.

[47] D. Greenwood, M.A. Johnson and M.J Grimble, "Multivariable LQG Optimal Control Restricted Structure Control for Benchmarking and Tuning," *ECC*, 2003.

[48] M.J Grimble, "Robustness of Full Order and Restricted Structure Optimal Control Systems," *IEEE CDC*, pages 227-232, 2003.

[49] M.J Grimble, "Robustness of Full-order and Restricted-Structure Optimal Control Systems," *Int. J. Systems Science*, vol. 35, no. 6, pp. 375-388,2004.

[50] M.J Grimble, "Restricted Structure Optimal Linear Estimators," *IEE Proc. Visual Image Signal Process*, pp. 400-408, 2004.

[51] M.J Grimble, "Optimal Restricted Structure Control with Pre-specified Gain or Phase Margins," *IEE Proc. Pt. D.*, vol. 151, no. 3, pp. 271-277, 2004.

[52] M. J. Grimble, "Restricted Structure Predictive Optimal Control," *Optimal Control Applications and Methods*, vol. 25, no. 3, pp. 107-145, 2004.

[53] M.J Grimble, "Restricted Structure Feedforward and Feedback Stochastic Optimal Control," *IEEE CDC Conf.*, 1999.

[54] D. Uduehi, A. Ordys,M.J Grimble, "Multivariable PID Controller Design using Online Generalised Predictive Control Optimization," *IEEE Int. Conf. Control Applications*, vol.1, pp. 272-277 2002.

[55] M.J Grimble, "Restricted Structure Controller Tuning and Performance Assesment," *IEE Proc. Pt. D.*, vol.149, no.1, pp. 8-16, 2002.

[56] M.J Grimble, "Performance Assesment and Benchmarking LQG Predictive Optimal Controllers for Discrete-Time State-Space Systems," *Transactions on Inst. Meas. Control*, vol. 25, no.3, pp. 239-264, 2003.

[57] M.J Grimble, P. Majecki, “GMV and Restricted-Structure GMV Controller Performance Assessment Multivariable Case,” *ACC Conference*, 2004.

[58] M.J. Grimble, “Reduced-Order Non-Linear Generalised Minimum Variance Control for Quasi-Linear Parameter Varying Systems,” *IET Control Theory and Applications*, vol. 12, issue 18, pp. 2495-2506, 2018.

[59] C. Cebeci, M.J. Grimble, R. Katebi and L.F. Recalde, “Restricted Structure Non-Linear Generalized Minimum Variance Control of a 2-Link Robot Arm,” *UKACC 12th International Conference on Control*, pp. 367-372, Sheffield, UK, 2018.

[60] M.J. Grimble, P. Majecki, “Restricted Structure Predictive Control for Linear and Non-linear Systems,” *Int. Journal of Control*, 2018.

[61] M.J. Grimble, “Three Degrees of Freedom Restricted Structure Optimal Control for Quasi-LPV Systems,” *57th IEEE Conference on Decision and Control, CDC*, Miami, FL, USA, pp. 2470-2477, 2018.

[62] A. G. J. Macfarlane, “Return-difference and return-ratio matrices and their use in analysis and design of multivariable feedback control systems,” *Proc. IEEE*, vol. 118, no. 7, 1971.

[63] M. J. Grimble, P. Majecki, “Weighting Selection for Controller Benchmarking and Tuning,” Technical report, Industrial Control Centre, University of Strathclyde, UK, 2005.

[64] T.B. Sheridan, “Three Models of Preview Control,” *IEEE Transactions on Human Factors in Electronics*, 1966

[65] E.K. Bender, “Optimum Linear Preview Control With Application to Vehicle Suspension,” *ASME J. Basic Eng.*, vol. 90, issue 2, pp. 213–221, 1968.

[66] M. Tomizuka and D.E. Whitney (1975), “Optimal Discrete Finite Preview Problems: Why and How Is Future Information Important?,” *ASME J. Dyn. Syst., Meas., Control*, pp. 319–325, 1975.

[67] K. Takaba, “A Tutorial on Preview Control Systems,” *Proceedings of the SICE Annual Conference*, pages 1388–1393, Japan, 2003.

[68] R. S. Sharp and V. Valtetsiotis, “Optimal Preview Car Steering Control,” *Vehicle System Dynamics Supplement*, vol. 35, pp. 101–117, 2001.

[69] E. Uzunsoy and V. Erkilinc, “Development of a Trajectory Following Vehicle Control Model,” *Advances in Mechanical Engineering*, vol. 8, issue 5, pp. 1–11, 2016.

[70] M.M. Negm, A.H. Mantawy, M.H. Shwehdi, “ A Global ANN Algorithm for Induction Motor Based on Optimal Preview Control Theory,” *Proceedings of IEEE Bologna Power Tech Conference*, pp. 1–7, 2003.

[71] N. Birla and A. Swarup (2014), “Optimal Preview Control: A Review,” *Optimal Control Applications and Methods*, 2014.

[72] A. Boyali, V. John, Z. Lyu, R. Swarn and S. Mita, “Self-Scheduling Robust Preview Controllers for Path Tracking and Autonomous Vehicles,” *Asian Control Conference*, 2017.

[73] H. Tobata, K. Fukuyama, T. Kimura and N. Fukushima, “Advanced Control Methods of Active Suspension,” *Vehicle System Dynamics*, vol. 22, pp. 347–358, 1993.

[74] L. Saleh, P. Chevrel, F. Claveau, JF. Lafay, and F. Mars (2013), “Shared Steering Control Between a Driver and Automation Stability in the Presence of Driver Behavior Uncertainty,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no, 2, 2013

[75] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi and H. Hirukawa, "Biped Walking Pattern Generation by Using Preview Control of Zero-Moment Point," *Proceedings of the IEEE International Conference on Robotics & Automation*, Taipei, Taiwan, 2003.

[76] M.M. Negm, A.F. Kheireldin, "Properties of Adaptive, Optimal and Preview Controllers Based on MVC and LQS Optimal Controller Application to Robotic Manipulator," *Proceedings of the International Conference on Control*, pp. 323–328, 1991.

[77] A. Stotsky and B. Egardt, "Model-Based Control of Wind Turbines: Look-Ahead Approach," *Proc IMechE Part I: J. Systems and Control Engineering*, pp. 1029–1038, 2012.

[78] A. J. Hazell. *Preview Control Toolbox.* (2018).[Online]. Available: `https://github.com/ajhazell/preview_control_toolbox.`

[79] A. J. Hazell, "Discrete-Time Optimal Preview Control," PhD thesis, Imperial College, London, 2008.

[80] A.W. Ordys, M. Tomizuka and M.J. Grimble, "State-Space Dynamic Performance Preview Predictive Controllers," *Journal of Dynamic Systems Measurement and Control - Transactions of the ASME*, vol. 129, issue 2, pp. 144–153, 2007.

[81] D. J. Cole, A. J. Pick and A. M. C. Odhams, "Predictive and Linear Quadratic Methods for Potential Application to Modelling Driver Steering control," *Vehicle System Dynamics*, vol. 44, no 3, pp. 259–284, 2006.

[82] M. Ulusoy. *Designing an MPC controller with Simulink.* (2021)[Online].

Available: (https://www.mathworks.com/matlabcentral/fileexchange/68992-designing-an-mpc-controller-with-simulink), MATLAB Ctr. File Exchange. Retrieved July 24, 2021.

[83] H. Ito. *Toyota Front-Loads Development of Engine Control Systems Using Comprehensive Engine Models and SIL.* (2019). [Online]. Available:`https://uk.mathworks.com/company/user_stories/reducing-time-to-market-using-model-based-design-qa-with-toyota.html`

[84] A. Shawky, D. Zydek, Y. Z. Elhalwagy, A. Ordys, "Modeling and Nonlinear Control of a Flexible-link Manipulator," *Applied Mathematical Modelling*, vol. 37, pp. 9591–9602, 2013.

[85] A. A. G. Siqueira, M. H. Terra and M. Bergerman, *Robust Control of Robots: Fault Tolerant Approaches,* Springer-Verlag London, 2011.

[86] H. Abbas, S. M. Hashemi and H. Werner, "Decentralized LPV Gain-Scheduled PD Control of a Robotic Manipulator," *Proc. ASME Dynamic Systems and Control Conference*, 2009.

[87] S. M. Hashemi, U. Gürcüoğlu, H. Werner, "Interaction Control of an Industrial Manipulator using LPV Techniques," *Mechatronics*, vol. 23, pp. 689–699, 2013.

[88] M. Löhning, "LPV and LFR Modelling of Elastic Robots for Controller Synthesis," *8th IEEE International Conference on Control and Automation*, pp. 522-527, 2010.

[89] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modelling and Control,* John Wiley & Sons, 1st edition, 2005.

[90] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning, and Control,* Cambridge U. Press, 2017.

[91] F.L.Lewis, D. M.Dawson, C. T.Abdallah, *Robot Manipulator Control Theory and Practice,* Marcel Dekker, Inc., 2nd edition, 2003.

[92] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms in MATLAB,* Springer Tracts in Advanced Robotics, 1st edition, 2011.

[93] B. Lantos, L. Marton, *Nonlinear Control of Vehicles and Robots,* Advances in Industrial Control, 2011.

[94] JJ. E. Slotine, W. Li, *Applied Nonlinear Control,* Prentice Hall, 1988.

[95] N. T. Nguyen, *Model-Reference Adaptive Control - A Primer,* Springer International Publishing, 2018.

[96] G. M. Hoffmann, C. J. Tomlin, M. Montemerlo, S. Thrun, "Autonomous Automobile Trajectory Tracking for Off-Road Driving: Controller Design, Experimental Validation and Racing," *American Control Conference*,2007

[97] SAE. *Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems.* (2014). [Online]. Available: `https://www.sae.org/standards/content/j3016_201401/`.

[98] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, "Predictive Active Steering Control for Autonomous Vehicle Systems," *IEEE Transactions on Control Systems Technology*, vol. 15, no. 3, 2007.

[99] C. M. Kang, Y. S. Gu, S. J. Jeon, Y.S. Son, W. Kim, S.H. Lee and C.C. Chung (2016), "Lateral Control System for Autonomous Lane Change System on Highways," *SAE Int. J. Passeng. Cars - Mech. Syst.*, vol. 9, issue 2, pp. 877–884, 2016.

[100] National Highway Traffic Safety Administration. *Analysis of Lane-Change Crashes and Near-Crashes.* (2009). [Online]. Available: `nhtsa.gov/sites/nhtsa.dot.gov/files/811147.pdf`

[101] Rajesh Rajamani, *Vehicle Dynamics and Control,* Springer, New York, 2nd edition, 2012.

[102] Dept. for Transport, Office for Low Emission Vehicles, Dept. for Business, Energy & Industrial Strategy, The Rt H. A. Sharma MP, and The Rt H. G. Shapps MP. *Government Takes Historic Step Towards Net-zero with End of Sale of New Petrol and Diesel Cars by 2030.* (Nov. 18, 2020). Accessed on: Mar. 1, 2021. [Online]. Available: https://www.gov.uk/government/news/government-takes-historic-step-towards-net-zero-with-end-of-sale-of-new-petrol-and-diesel-cars-by-2030.

[103] P.A. Eisenstein, NBC news. *GM to Go All-electric by 2035, Phase out Gas and Diesel Engines.* (Jan. 29, 2021). Accessed on: Mar. 1, 2021. [Online]. Available: https://www.nbcnews.com/business/autos/gm-go-all-electric-2035-phase-out-gas-diesel-engines-n1256055.

[104] MathWorks Student Competitions Team. *MATLAB and Simulink Racing Lounge: Vehicle Modeling.* (2021). Accessed on: Jan. 12, 2021. [Online]. Available: https://github.com/mathworks/vehicle-modeling/releases/tag/v4.1.1.

[105] T. D. Gillespie, *Fundamentals of Vehicle Dynamics*, SAE Inc., 1992.

[106] Z. Sun, G. G. Zhu, *Design and Control of Automotive Propulsion Systems*, CRC Press, 2015.

[107] B. Zhang, C. C. Mi and M. Zhang, “Charge Depleting Control Strategies and Fuel Optimization of Blended-mode Plug-in Hybrid Electric Vehicles,” *IEEE Trans. on Vehicular Technology*, 2011.

[108] Max Pixel. *Automobile Design Plan Blueprint Drawing Technical.* (2021). Accessed on: Jul. 8, 2021. [Online]. Available: https://www.maxpixel.net/Automobile-Design-Plan-Blueprint-Drawing-Technical-30577

[109] G. L. Plett, *Battery Management Systems, Volume II: Equivalent-Circuit Methods*, Artech House, 2016, vol.2, ch. 3.

[110] M. A. Goodrich, W. C. Stirling and R. L. Frost,"A Theory of Satisficing Decisions and Control," *IEEE Tran. on Systems, Man, and Cybernetics*, 1998.

[111] J. W. Curtis and R. W. Beard,"Ensuring Stability of State-dependent Riccati Equation Controllers via Satisficing," *Proceedings of the 41st IEEE Conference on Decision and Control*, 2002.

[112] R. W. Weeks and J. W. Moskwa, "Automotive Engine Modeling for Real-Time Control Using MATLAB/SIMULINK," *SAE International*, 1995.

[113] S. Hur and M. J. Grimble, "Robust Nonlinear-Generalised Minimum Variance Control and Fault Monitoring," *International Journal of Control, Automation and Systems*, 2015.

[114] Y. Pang , H. Xia, M. J. Grimble, "Resilient Nonlinear Control for Attacked Cyber-Physical Systems," *IEEE Transactions on Systems, Man and Cybernetics*, 2018.

[115] G.L. Plett, "Extended Kalman Filtering for Battery Management Systems of LiPB-based HEV Battery Packs Part 3. State and Parameter Estimation," *Journal of Power Sources*, 2004.

[116] G.Florentino and M.S. Trimboli, “Lithium-ion Battery Management Using Physics-based Model Predictive Control and DC-DC Converters,” *IEEE Transportation Electrification Conference and Expo (ITEC)*, 2018.

[117] M.A. Xavier and M.S. Trimboli, “Lithium-ion Battery Cell-level Control Using Constrained Model Predictive Control and Equivalent Circuit Models,” *Journal of Power Sources*, 2015.

[118] A. A. Brown. *Github page.* (2014). [Online]. Available: `https://gist.github.com/Alexanderallenbrown/c63ca13079bea1a67812`.

[119] E. Bertolazzi. *Clothoids Toolbox.* (2020). [Online]. Available: `https://github.com/ebertolazzi/Clothoids`.

[120] L. Edsall and C. Endres, *Chevrolet Volt: Charging Into the Future*, Minneapolis, Minnesota: Motorbooks, 2010.

[121] S. Waslander and J. Kelly, Self-Driving Cars Specialization: “Introduction to Self-Driving Cars”, University of Toronto on Coursera, 2019. [Online]. Available: `https://www.coursera.org/learn/intro-self-driving-cars?specialization=self-driving-cars`.

[122] G. Plett, “Algorithms of Battery Management Systems Specialization”, University of Colorado Boulder on Coursera, 2021. [Online]. Available: `https://www.coursera.org/specializations/algorithms-for-battery-management-systems`.

Appendix A

RS-NPGMV Control

This chapter follows from the research made in chapter 4 and its case study in chapter 6. For this study, previewing is considered as an enhancement for the RS-NGMV controller. It will be concluded that the improvement would require a the design of a different RS controller, that is the RS-NPGMV. The RS-NGMV we have presented in chapter 3 of this thesis, is a single degree-of-freedom controller and direct approaches to including future reference information in the augmented model like the Tomizuka's original study would not solve the problem.

This chapter is organized in two sections. In the first section, the motivations for a preview enhancement to the RS-NGMV are explained by comparing to the MPC using simulations. In the second section, the attempts that had been taken to realize the objective of adding preview action will be explained. Conclusions will be made and the RS-NPGMV will be proposed.

The roadmap ahead for the investigation includes modifying the RS-NGMV cost-function for the RS-NPGMV problem first. The solution for the new cost-function will return the RS-NPGMV optimal control law. Once the controller is derived, preview procedures similar to those in previous chapters, will be taken and simulations will be made to verify.

A.1 Preview Extension of RS-NGMV Control

Recall the preview control concept introduced in section 4.1, which hints that it is possible to consider preview as an action for a feedback controller if the future reference information is available. As was stated, the NGMV is not a predictive controller (and so is not the RS-NGMV) and despite its several other advantages, it cannot utilize future information. Therefore, we have considered the preview action for the RS-NGMV controller by following the same intuition as that of the LQ-Preview control, in incorporating the future reference signal in the system model and then following with optimal control solutions. The result would be the RS-NGMV preview controller.

It was shown in chapter 6 that a good application example for the preview control can be given as the lane-changing of autonomous vehicles where the future reference trajectory is the lane-changing manoeuvre and is ideally known. A simulation demo from Mathworks [82] has impressive results for this control problem using MPC and Adaptive MPC. Thus, as an initial attempt an RS-NGMV controller was designed and applied on the systems in the demo for comparative purposes. The results of the attempt are shown in the section below.

A.1.1 Verifying Motivations with Comparisons to MPC

Model-Based control design is becoming very popular in the industry [83] owing to its great advantage in enabling fast product testing and verification. Similar to using building blocks, its possible to apply a method for different system models rapidly. Model-Based control methodology has been the approach of this thesis as well with the Restricted Structure NGMV.

Among the Model-Based approaches, the MPC control solutions are the most demanded. Although, it is not the main target of this thesis the predictive control techniques like the MPC have also been studied for understanding key concepts, investigating connections with the NGMV techniques and as in this section states the possible extension for the RS-NGMV control.

For providing a brief summary of the MPC technique used in this study, consider the standard quadratic MPC cost function below,

$$J\big(e(t), \Delta u(t)\big) = \sum_{i=0}^{H_p} W_e\big(e_{t+i}(t)\big)^2 + \sum_{i=0}^{H_c} W_{\Delta_u}\big(\Delta u_{t+i}(t)\big)^2, \tag{A.1}$$

$$\begin{aligned} s.t. \quad & \delta_{min} \leq u_i(t) \leq \delta_{max}, \\ & \Delta u_{min} \leq \Delta u_i(t) \leq \Delta u_{max}, \\ & y_{min} \leq y(t) \leq y_{max}, \end{aligned}$$

minimization of which over the prediction horizon H_p and control horizon H_c returns the MPC controller. The error signals are denoted by,

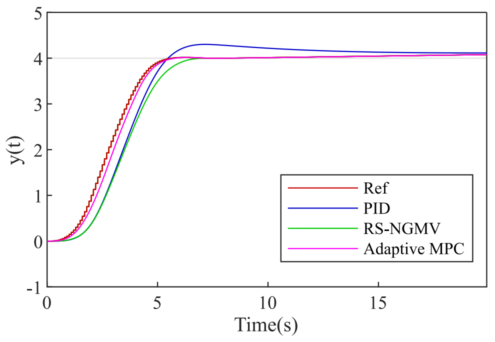
$$e_i(t) = y_i(t) - y_{ref}(t), \;\; i = \{t, \ldots, t + H_p\}.$$

and the control input increments are denoted by,

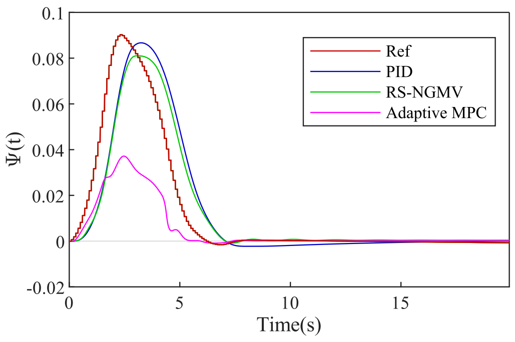
$$\Delta u_i(t) = u_i(t) - u_i(t-1), \;\; i = \{t, \ldots, t + H_c\}.$$

The error and control signals are penalized by weights W_e and W_{Δ_u} respectively.

The lane change simulations provided by the Mathworks demo uses the MPC formulation given above within the MPC and the Adaptive MPC blocks (uses a Kalman filter in addition) for the reference tracking of the lateral position and yaw angle parameters. For the simulations of this section the Adaptive MPC was chosen since it is a closer equivalent to the RS-NGMV and would make comparisons fair. PID was also chosen as the other baseline technique. The results are shown in Fig. A.1. In part (a) it is seen that although they have similar tracking performances, the Adaptive MPC is faster than the RS-NGMV



(a) Tracking of lateral position $y(t)$.



(b) Tracking of yaw angle $\psi(t)$.

Figure A.1: MPC vs. RS-NGMV

due to using predictions. In part (b) the RS-NGMV shows more robust tracking characteristics for the yaw angle $\psi(t)$ which is also anticipated owing to its natural robustness features. However, it has shown slower transient characteristics.

A.1.2 Results

Motivated by these findings, the second phase included incorporating the future reference signals within the augmented RS-NGMV state-space models and apply the RS-NGMV optimizations. The attempts to enable preview action did not show any improvement for the RS-NGMV. First, the LQ-Preview approach provided earlier was taken. Later, Sharp's vehicle modelling (different to Mathworks) and modified LQ-Preview approach were used [68, 118] .

After several attempts, we came to the conclusion that the problem with the RS-NGMV had been fundamental and structural. RS-NGMV is a 1-DOF controller using feedback errors only. Therefore, even when the future reference information are included in the models, it does not make a difference on the control action. The desired restricted structure preview controller needs to include the previewed reference signals in another degree of freedom which means a different controller than the RS-NGMV. However, due to the fact that NGMV algorithms are constructed to use k-steps ahead predictions only, it is question-

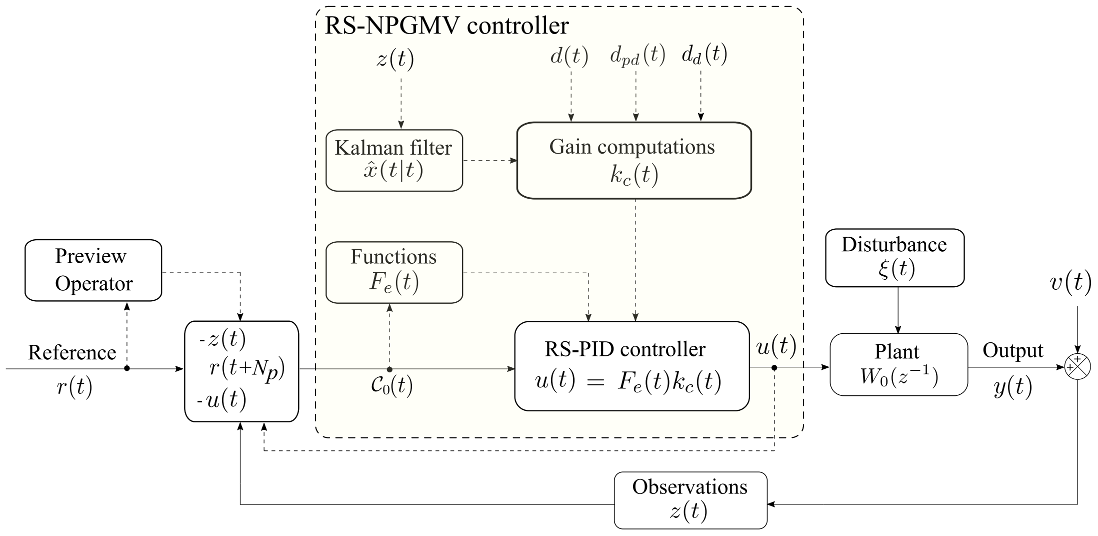


Figure A.2: RS-NPGMV controller.

able how useful previewing would be in this case. The preview horizon may not be fully covered in k-steps (i.e. $k \leq N_p$). Hence, the RS-NPGMV is proposed.

A.2 RS-NPGMV Controller

The proposed RS-NPGMV is given in Fig. A.2. 1-DOF feedback controllers use,

$$e(t) = r(t) - z(t), \tag{A.2}$$

and since we seek to include future reference information, a 2-DOF strategy should be considered,

$$e(t) = \begin{pmatrix} -z(t) \\ r(t+N_p), \end{pmatrix} \tag{A.3}$$

with N_p considering the preview steps and would cancel the action if set to zero. Recall the restricted structure in general z-transfer function form,

$$C_0(z^{-1}) = \frac{C_{0,num}(z^{-1})}{C_{0,den}(z^{-1})} = \frac{C_{0,num} + C_{1,num}z^{-1} + \cdots + C_{n,num}z^{-n}}{1 + C_{1,den}z^{-1} + \cdots + C_{m,den}z^{-m}}, \tag{A.4}$$

and express a controller using (A.4),

$$u(t) = C_0(z^{-1})e(t), \tag{A.5}$$

for which the cross product with numerator and denumerator terms yields,

$$u(t) = C_{0,num}(z^{-1})e(t) - (C_{0,den}(z^{-1}) - 1)u(t). \tag{A.6}$$

Recall the SISO RS-NGMV controller from (3.1),

$$u(t) = \sum_{j=1}^{N_e} f_j(z^{-1}, k_j(t))e(t), \tag{A.7}$$

and expressing it in the new terms described above,

$$u(t) = \sum_{j=1}^{N_e} f_j(z^{-1}, k_j(t))\mathcal{C}_0(t), \tag{A.8}$$

where the term $\mathcal{C}_0(t) = \begin{pmatrix} e(t) \\ -u(t) \end{pmatrix}$ and hence the RS-NPGMV input. Once again the control input can be parametrised as $u(t) = F_e(t)k_c(t)$ like before but for now it is not necessary to demonstrate the algebraic parametrisation process. However, attention must be paid that the controller (A.8) will be computed for the full horizon and therefore a vector of future control inputs will be used,

$$U_{t,N} = \begin{pmatrix} u(t) \\ u(t+1) \\ \vdots \\ u(t+N) \end{pmatrix} = \begin{pmatrix} F_e(t) \\ F_e(t+1) \\ \vdots \\ F_e(t+N) \end{pmatrix} k_c(t), \tag{A.9}$$

where $k_c(t)$ are the feedback gains calculated by the optimizations. The augmented state-space model is expected to remain the same as the RS-NGMV,

$$x(t+1) = A_t x(t) + B_t u_0(t-k) + D_t \xi(t) + d_d(t),$$
$$z(t) = C_t x(t) + E_t u_0(t-k) + d(t) + v(t).$$

The state and error predictor models for the LPVKF,

$$\hat{x}(t+k|t) = A_t^k \hat{x}(t|t) + \sum_{j=1}^{k} A_{t+j}^{k-j} B_{t+j-1} u_0(t+j-1-k) + d_{dd}(t+k-1),$$
$$e_p(t+i) = C_{pt+i} x(t+i) + E_{pt+i} u_0(t+i-k) + d_p(t+i),$$

also apply but these results should be collected in larger matrices for the future predicted values along the prediction horizon N for the RS-NPGMV controller.

Since the error and control signals are in 2-DOF format, plus a prediction horizon N is taken into account, the biggest difference will appear in RS-NPGMV cost-function.

Appendix B

General Matlab/Simulink Codes

The scripts and code snippets demonstrated here are used in general throughout the Matlab/Simulink documentation of this thesis. The case study specific simulation files will start in the following Appendix.

B.1 Generating Subsystem Models & Matrices

This code snippet shows how the W_d and disturbance model matrices and deterministic linear reference model W_r are created. They are used in main.m mostly or in Simulink as LTI model blocks. The W_d may receive outputs from random number blocks as white-noise and W_r receives reference signals.

```
%Disturbance and Reference Models and Matrices
nxd = 1; %size of disturbance states
nxr = 1; %size of reference states
Wd = .1*eye(nxd)*filt([0 1],[1 -0.98],Ts);
[Ad,Bd,Cd,Ed] = ssdata(Wd);
Wr = eye(nxr)*filt([0 1],[1 -0.9999],Ts);
```

The dynamic control weighting F_{ck} and the error weighting P_c are created and their subsystem matrices are extracted similarly to above. The P_c matrices are directly used in controllers but F_{ck} matrices are used in a state-space block on

Simulink and sent to RS-NGMV s-functions. Like the reference and disturbance models they are also defined in main.m mostly.

```
%Control and Error Weight Models and Matrices
Fck = .0001*zpk( 1,.1,10,Ts);
% Dynamic control weightings
[Af,Bf,Cf,Df] = ssdata(Fck);
Pc = 100*zpk(.97,1,10,Ts);
% Dynamic error weightings
[Ap,Bp,Cp,Ep] = ssdata(Pc);
```

The input weighting on $u_0(t)$ if needed, would be created using the code chunk below and then applied on the Simulink model using LTI block.

```
% Input weighting on u0(t)
au1 = 0.93; b1 = 0.95;
Wu = 1*(1-au1)/(1-b1)*filt([1 -b1],[1 -au1],Ts);
[zWu,kWu] = zero(Wu); pWu = pole(Wu);
% [Au,Bu,Cu,Eu] = ssdata(Wu);
[Au,Bu,Cu,Eu] = zp2ss(zWu,pWu,kWu);
nxu = size(Au,1);
Wux = ss(Au,Bu,eye(nxu),zeros(nxu,nu),Ts);
xu_0 = zeros(nxu,1);
```

Most parameters that are defined in main files are stored in Matlab structs to be called by functions as well as s-functions in Simulink models. Below is a sample.

```
%parameter struct for RS-NGMV cost functions, fixed PID gains, sizes and
     controller limits.
p0 = struct('kp',kp,'kI',kI,'kd',kd,'lambdap',lambdap,...
     'lambdau',lambdau,'lambdak',lambdak,'lambdad',lambdad,...
     'nyc',nyc,'nym',nym,'nu',nu,'nx0',nx0,'nd',nd,'Fck',Fck,'Pc',Pc,...
     'u_max',u_max,'u_min',u_min,'c2d_flag',c2d_flag);
%Subsystem matrices.
p1 = struct('Name','par',...
     'Ap',Ap,'Bp',Bp,'Cp',Cp,'Ep',Ep,...
     'Au',Ap,'Bu',Bp,'Cu',Cp,'Eu',Ep,...
     'Af',Ap,'Bf',Bp,'Cf',Cp,'Df',Df,...
     'Ad',Ap,'Bd',Bp,'Cd',Cp,'Cm',Df,...
     'nxp',nxp,'nxi',nxi,'nxr',nxr,...
     'QN',QN,'QN2',QN2,'RN',RN,...
     'A',A,'B',B,'C',C,'D',D,'Ts',Ts);
p1 = catstruct(p1,p0); %merging two structs.
```

B.2 Discretizations

It is shown here how the discretizations of the models are performed. The parameter cd2_flag is simply an indicator for the code to perform the desired discretization technique. In this thesis, this option was coded but only Euler technique was used for the discretizations.

```
% Model discretization
if (p1.c2d_flag==0)
        A0 = eye(p1.nx0) + Ts*A;
        B0 = Ts*B;
        D0 = Ts*D;
elseif (p1.c2d_flag==1)
        XX = [A B D; zeros(p1.nxu+nxd,p1.nx0+p1.nxu+nxd)];
        YY = expm(XX*Ts);
        A0 = YY(1:p1.nx0,1:p1.nx0);
        B0 = YY(1:p1.nx0,p1.nx0+1:p1.nx0+p1.nxu);
        D0 = YY(1:p1.nx0,p1.nx0+p1.nxu+1:end);
```

B.3 Figure Plotting

The script “plotmyfigure.m” has been used for plots. Each case study has it in its directory. This one is a sample. Each example contains modified versions to extract the results wanted. To workspace blocks were placed in Simulink usually named in the format “out.myResult” where data was collected during the run and then plot by running this script manually.

```
width_figure = 8.59;
height_figure = width_figure/1.618;
width = 7.3;
height = width/1.618;
marginw = .7;

FolderName = 'D:\...\ngmv_control_software_v1\cagatay_examples\EV\RS-NGMV
    control of EV (scalar)\results';
tinit = 0; %
tinit = Ts; %sampling time Ts
```

```
tend = TimeUDDS;
%color palete
color1cyan = [102 204 204]./255;
color2orange = [255 153 81]./255;
color3purple = [204 51 153]./255;
color4green = [102 204 51]./255;
color5maroon = [204 51 51]./255;
color6navy = [0 0 204]./255;
color7greener = [0 204 0]./255;
color8red = [204 0 0]./255;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% plot figure 1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fig_LL1 = figure('Name','RS-NGMV vs PI','NumberTitle','off');
%xlim([0 tend-tinit]);
xlim([tinit tend]);
set(fig_LL1,'units','centimeters');
set(fig_LL1,'position',[5 20 width_figure height_figure]);
set(fig_LL1,'paperunits','centimeters');
set(fig_LL1,'papersize',[width height])
set(fig_LL1,'paperpositionmode','auto');

hold on; grid on
plot(setuniformtime(getsampleusingtime(out.SOCPI, tinit,tend),'Interval',Ts),'-'
    ,'Color',color6navy);
plot(setuniformtime(getsampleusingtime(out.SOCRS, tinit,tend),'Interval',Ts),'-'
    ,'Color',color2orange);
title('UDDS drive-cycle')
hold off

xlim([0 tend-tinit]);
set(gca,'fontsize',8);
set(gca, 'FontName', 'Times New Roman')
set(gcf, 'Renderer', 'Painters')
% ylim([0 0.4])
xlabel('Time(sec)','Interpreter', 'Tex', 'units','centimeters');
ylabel('%SOC','Interpreter', 'Tex','units','centimeters');
legend('PI','RS-NGMV','Location','best')
% yticks('auto')
% xticks([0:(tend-tinit)/4:(tend-tinit)])
print('-dsvg',strcat(FolderName,'\','socUDDS'))
print(gcf,'-dpng','-r300',strcat(FolderName,'\','socUDDS'))
```

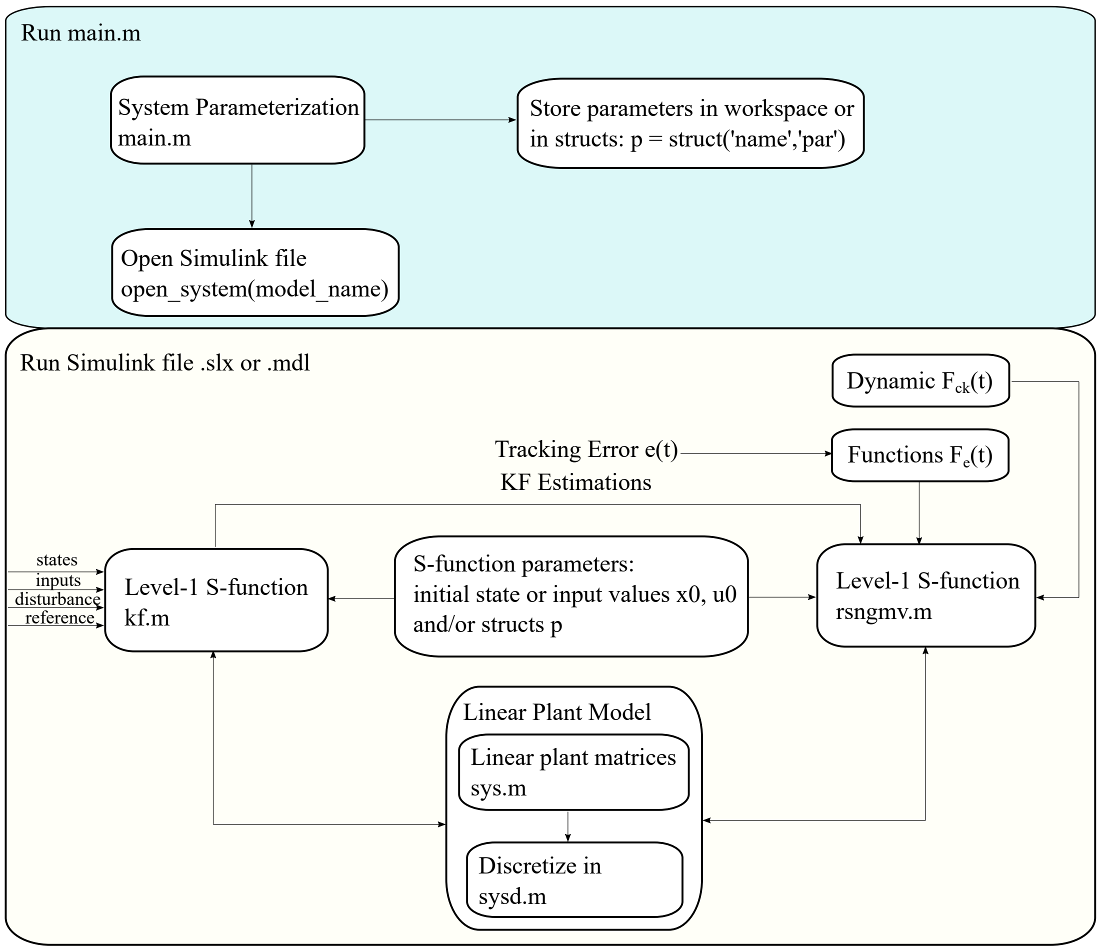

Run main.m
System Parameterization main.m
Store parameters in workspace or in struct: p = struct('name','par')
Open Simulink file open_system(model_name)
Run Simulink file .slx or .mdl
Dynamic $F_{ck}(t)$
Tracking Error e(t)
KF Estimations
Functions $F_e(t)$
states
inputs
disturbance
reference
Level-1 S-function kf.m
S-function parameters: initial state or input values x0, u0 and/or structs p
Level-1 S-function rsngmv.m
Linear Plant Model
Linear plant matrices sys.m
Discretize in sysd.m

Figure B.1: Matlab/Simulink files summarized implementation strategy.

B.4 Implementation Steps and Structure

In Fig. B.1 the overall methodology of using Matlab/Simulink files is given. The results can be observed through scopes or by extracting plots using plotmyfigure.m in only 2 steps, that are:

- Run main.m (opens Simulink model)
- Run model_name.slx or model_name.mdl

The outputs of the RS-NGMV s-function are gains and the control signal that is used by the physical model to be controlled (e.g. robot, car and etc.).

Appendix C

2-link Robotic Manipulator

The main.m for parametrisation is given below.

```
Ts   = .005; %Sampling Time
%% Robot params and initial cond.
damp1 = 0; damp2 = 0;                          % damping of the model (plant)
damp1_m = damp1; damp2_m = damp2;        % damping of the model (controller)
q1_0=0;q2_0=0;q1dot_0=0;q2dot_0=0;
q0=[q1_0;q2_0];qdot0=[q1dot_0;q2dot_0];
l1=1;lc1=0.5;m1=1;I1=0.12;lc2=0.6;m2=2;I2=0.25;delta_e=30*pi/180;g=0;
a1 = I1 + m1*lc1^2 + I2 + m2*lc2^2 + m2*l1^2;
a2 = I2 + m2*lc2^2;
a3 = m2*l1*lc2*cos(delta_e);
a4 = m2*l1*lc2*sin(delta_e);
param = {l1,lc1,m1,I1,lc2,m2,I2,delta_e,damp1,damp2,a1,a2,a3,a4};
%% Reference and Disturbance Models.
stop_time = 300; %t = 150
q1_d = [0,60*pi/180; 50,30*pi/180; 100,60*pi/180];      q1_dt = q1_d';
q2_d = [0,90*pi/180; 50,45*pi/180; 100, 90*pi/180];     q2_dt = q2_d';
reference_robot = siggen(stop_time,{'steps',q1_dt(:),'steps',q2_dt(:)},Ts);
Td1 = [15,0.01;12,0.01;17,0.01;18,0.02;19,0.01;20,0.02;21,0.03]';
Td2 = [6,-0.03;7,-0.01;8,0.01;9,0.02;10,0.05;15,0.03;20,0.009]';
load_disturbance =siggen(stop_time,{'steps',Td1(:),'steps',Td2(:)},Ts);
%% System Information
nx0 = 4;% states
nu = 2; % inputs
nyc = 2;% controlled outputs
nym = 2;% measured outputs
%% Create Reference and Disturbance models/signals
```

```
Wr = eye(nyc)*filt([0 1],[1 -0.9999],Ts);
nxr = order(Wr); %references
Wd = 0.001*eye(nyc)*filt([0 1],[1 -0.001],Ts);
nd = order(Wd); % disturbances
[Ad,Bd,Cd,Ed] = ssdata(Wd)
Cdm = Cd;
xd_0 = zeros(nd,1);
%% error weightings
lambdap = [1e-4 0;0 1e-4]; Pc = [zpk(.97,1,87.5,Ts) 0; 0 zpk(.97,1,87.5,Ts)];
% Dynamic error weighting
[Ap,Bp,Cp,Ep] = ssdata(Pc);
nxp = size(Ap,1);
Pcx = ss(Ap,Bp,eye(nxp),zeros(nxp,nyc),Ts);
xp_0 = zeros(nxp,1);
%% control weightings
lambdau = [0.05 0;0 0.05];
lambdak = (10^-6)*diag([3 .5 7, 3 .5 7, 3 .5 7, 3 .5 7]);
Fck = [zpk( .9,.4,0.016,Ts) 0; 0 zpk( .9,.4,0.16,Ts)];
% Dynamic control weighting
[Af,Bf,Cf,Df] = ssdata(Fck);
lambdad = 80*diag([1 1 1, 1 1 1, 1 1 1, 1 1 1]);
% Total number of states
nx = nx0 + nd + nxp;
%% Covariances for the Kalman Filter
qn_x1 = 0.1;% x1 state uncertainty
qn_x2 = 0.1;% x2 state uncertainty
qn_x3 = 0.1;% x3 state uncertainty
qn_x4 = 0.1;% x4 state uncertainty
qn_ymd1 = 0.1;% ym1
qn_ymd2 = 0.1;% ym2
rn_ym1 = 10;% ym sensor variance
rn_ym2 = 10;% ym sensor variance
QN = diag([qn_x1 qn_x2 qn_x3 qn_x4 qn_ymd1 qn_ymd2]);   %4-state model
QN2 = diag([qn_x1 qn_x2 qn_x3 qn_x4]);  %4-state model
RN = diag([rn_ym1 rn_ym2]);
%% Initilization
x0_0 = [0; 0; 0; 0];
x_0 = [x0_0 xd_0 xp_0];
u0_0 = [0 0];
%Fixed PID gains
kp = 3;
kI = 0.5;
kd = 7;
```

```
alpha   = k3*10;
% Defining kc_bar for parallel RS form
kc_bar1 = [0;kp]';
kc_bar2 = [0;kI]';
kc_bar3 = [0;kd]';
p1 = struct('l1',l1,'lc1',lc1,'m1',m1,'I1',I1,...
     'lc2',lc2,'m2',m2,'I2',I2,'delta_e',delta_e,...
     'damp1',damp1,'damp2',damp2,'a1',a1,'a2',a2,'a3',a3,'a4',a4,...
     'x0_0', x0_0, 'xd_0',xd,'xp_0',xp_0,'x_0',x_0,'u0_0',u0_0,...
     'nx0',nx0, 'nu',nu,'nyc', nyc,'nym',nym,'nx',nx,'nd',nd,...
     'nd', nd, 'nxr', nxr,'nxp',nxp,'c2d_flag',c2d_flag,'Ts',Ts,...
     'QN',QN,'QN2',QN2,'RN',RN,...
     'Ap',Ap,'Bp',Bp,'Cp',Cp,'Ep',Ep,'Af',Af,'Bf',Bf,'Cf',Cf,...
     'Df',Df, 'Ad',Ad,'Bd',Bd,'Cd',Cd,...
     'kp',kp,'kI',kI,'kd',kd,'alpha',alpha,'t_sim',t_sim,...
     'Fck',Fck,'Pc',Pc,'lambdap',lambdap,'lambdak',lambdak,...
     'lambdad',lambdad,'lambdau',lambdau);
```

The robot manipulator equations are entered in the script given here. They are used within the interpreted Matlab function shown in Fig. C.1. Its parameters were loaded in the array param from running the main.m. It is also possible to build plant to be controlled using Simulink blocks.

```
function [out] = robot_links(in,param)
[l1,lc1,m1,I1,lc2,m2,I2,delta_e,damp1,damp2,a1,a2,a3,a4] = param{:};
tau1 = in(1);
tau2 = in(2);
q1 = in(3);
q2 = in(4);
q1dot = in(5);
q2dot = in(6);
H(1,1) = a1 + 2*a3*cos(q2) + 2*a4*sin(q2);
H(1,2) = a2 + a3*cos(q2) + a4*sin(q2);
H(2,1) = H(1,2);
H(2,2) = a2;
h = a3*sin(q2) - a4*cos(q2);
inv_H = 1/(H(1,1)*H(2,2)-H(1,2)*H(2,1))*[H(2,2) -H(1,2); -H(2,1) H(1,1)];
C = [-h*q2dot+damp1 -h*(q1dot+q2dot);
          h*q1dot          damp2           ];
 out = inv_H*([tau1;tau2] - C*[q1dot;q2dot]); %qddot = y = H^-1*(T - C.qdot)
end
```

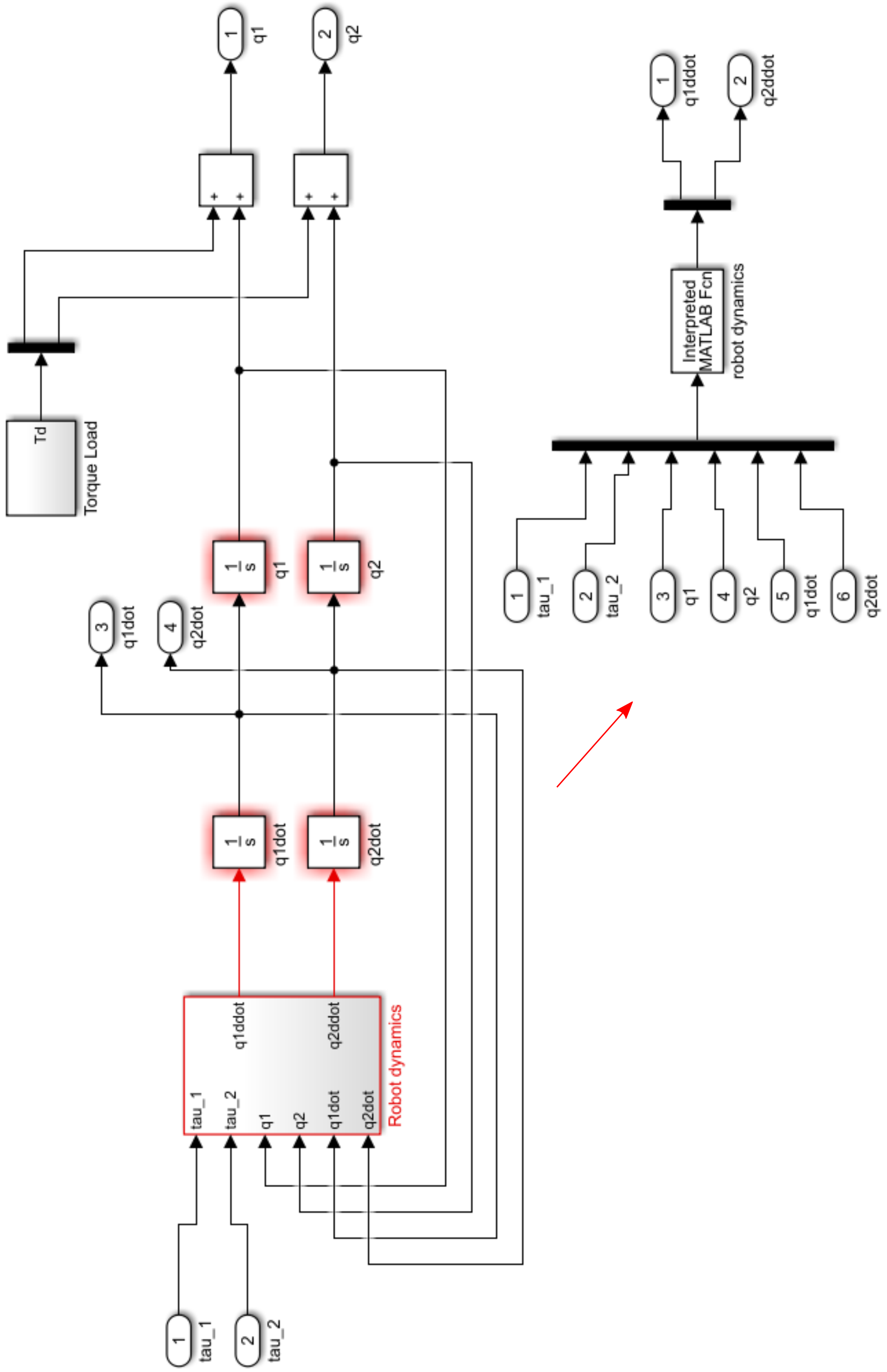


Figure C.1: 2-link robotic manipulator Simulink block.

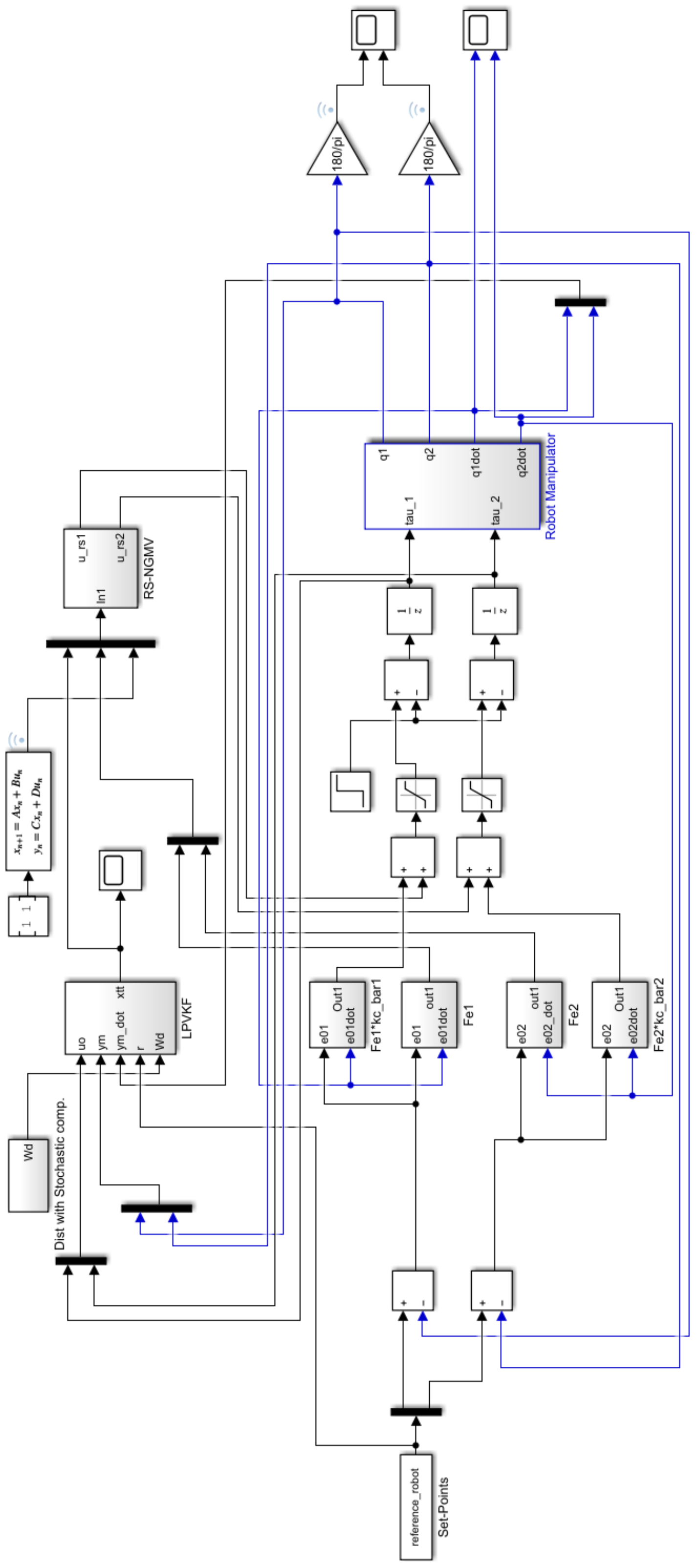


Figure C.2: 2-link robotic manipulator Simulink control diagram.

The Fig. C.2 shows the RS-NGMV control diagram for the 2-link robot manipulator study.

C.1 Reference and Torque Load Generation

These signals were created using siggen function for this application, the code chunk is in the main.m but annotated below for demonstration.

```
stop_time = 300; %Simulation time
%Reference signal generation.
q1_d = [0,60*pi/180; 50,30*pi/180; 100,60*pi/180]; %format is [time instant,
    value]
q1_dt = q1_d';
q2_d = [0,90*pi/180; 50,45*pi/180; 100, 90*pi/180];
q2_dt = q2_d';
reference = siggen(stop_time {'steps',q1_dt(:),'steps',q2_dt(:)},Ts);
%Torque load signal generation for both links.
load_disturbance =siggen(stop_time,{'steps',[15,0.01,16,0.01],'steps'
    ,[6,-0.03,7,-0.01,35,-0.01]},Ts); %format is [time instant,value]
```

C.2 q-LPV Plant Model and Discretization

The following codes sys.m and sysd.m represent the linear plant model W_0 for the LPVKF and the RS-NGMV. Note that they are the same as the Simulink plant (the physical plant) but in other cases iy may be different. These are the plant models that the LPVKF and the RS-NGMV knows. The latter case could mean model mismatch if the internal models are not realistic enough.

```
function [A0,B0,G0,Cc,Ec,Hc,Cm,Em,Hm] = sys(p,p1)
%qLPV robot model
struct2vars(p1) % struct2vars copies structure fields to local workspace
[q1,q2,q1dot,q2dot] = vsplit(p); %splits the vector into components. To modify
    length alternative use vsplit(p,DIMS).
H(1,1) = a1 + 2*a3*cos(q2) + 2*a4*sin(q2);
H(1,2) = a2 + a3*cos(q2) + a4*sin(q2);
H(2,1) = H(1,2);
```

```
H(2,2) = a2;
h = a3*sin(q2) - a4*cos(q2);
inv_H = 1/(H(1,1)*H(2,2)-H(1,2)*H(2,1))*[H(2,2) -H(1,2); -H(2,1) H(1,1)];
K = [-h*q2dot+damp1  -h*(q1dot+q2dot);
              h*q1dot         damp2  ]; %coriolis
A = [zeros(2) eye(2);
        zeros(2) -inv_H*K];
B = [zeros(2);inv_H];
%x1  x2  x3  x4
G0 = [0;  0;  0;  0];
% Construct controlled output matrices
%x1  x2  x3  x4
Cc = [ zeros(2) -inv_H*K]; %yc
Ec = inv_H; Hc = 0;
% Construct measured output matrices
%x1  x2  x3  x4
Cm = [ zeros(2) -inv_H*K];%ym
Em = inv_H; Hm = 0;
```

The sysd.m calls sys.m to extract the information and discretize the linear plant model.

```
function [A0,B0,Cc,G0,Ec,Hc,Cm,Em,Hm] = sysd(p,p1,Ts)
%qLPV discrete-time robot model
% x(n+1) = A0(x)*x(n) + B0(x)*u(n-k) + G0(x)*d(n)
% yc(n+1) = Cc(x)*x(n) + Ec(x)*u(n-k) + Hc(x)*d(n)      -- controlled outputs
% ym(n+1) = Cm(x)*x(n) + Em(x)*u(n-k) + Hm(x)*d(n)      -- measured outputs
% x = [q1,q1dot,q2,q2dot], u = u0, d = Wd, y = yc = ym.
[A0,B0,Cc,G0,Ec,Hc,Cm,Em,Hm] = sdsys(p,p1);
Ts = 0.05;
% Model discretization
if (p1.c2d_flag==0)
        A0 = eye(pr1.nx0_robot) + Ts*A0;
        B0 = Ts*B0;
        G0 = Ts*G0;
elseif (p1.c2d_flag==1)
        XX = [A0 B0 G0; zeros(pr1.nu+nd,pr1.nx0+pr1.nu+nd)];
        YY = expm(XX*Ts);
        A0 = YY(1:pr1.nx0,1:pr1.nx0);
        B0 = YY(1:pr1.nx0,pr1.nx0+1:pr1.nx0+pr1.nu);
        G0 = YY(1:pr1.nx0,pr1.nx0+pr1.nu+1:end);
end
```

C.3 LPVKF

The LPVKF block contains the s-function lpvkf.m given below. It calls the sdysd.m to access the linear plant model and then through the s-function parameters (p1 struct loaded after main.m), it builds the augmented state-space model to carry forward with the estimation equations.

```
function [sys,x0,str,ts] = lpvkf(t,x,u,flag,p1)
%LPV Kalman Filter in s-function form
% State:        x(t|t-1)
% Output:       x(t|t)
% Input:        [u(t-k), ym(t), d(t), r(t)]
% u = u0, d = Wd, y = yc = ym
persistent P xtt QN2 RN
persistent nx0 nd nxp nu
persistent Ts t_sim
persistent Ad Bd Cd Ap Bp Cp Ep

switch flag
        case 3
                [u01,u02,ym1,ym2,ym1_dot,ym2_dot,r1,r2,Wd1, Wd2] = vsplit(u);
                ut = [u01 u02]';
                rt = [r1 r2]';
                ymt = [ym1 ym2]';
                ymt_dot = [ym1_dot ym2_dot]';
                [q1,q2] = vsplit(ymt(1:2));
                [q1dot,q2dot] = vsplit(ymt_dot(1:2));
                p = [q1;q2;q1dot;q2dot];
                [A0,B0,G0,Cc,Ec,Hc,Cm,Em,Hm] = sdsysd(p,p1,Ts);
                %AUGMENTED MODEL
                A = [A0,            zeros(nx0,nd), zeros(nx0,nxp);
                     zeros(nd,nx0),            Ad, zeros(nd,nxp);
                     -Bp*C0,                  -Bp*Cd,          Ap];
                B = [B0;          zeros(nd,nu);-Bp*Ec];
                C = [-Ep*Cc, -Ep*Cd, Cp];
                E = -Ep*Ec;
                D0 = eye(nx0_robot);
                D = [D0, zeros(nx0,nd) ; zeros(nd,nx0), Bd; zeros(nxp,nx0+nxp)];
                dxt = [0;0;0;0];
                dyt = [0 0]'; % measurable dist.

                % state update
                x1 = A*xtt + B*ut + [dxt; zeros(nd,1); Bp*(rt - dyt)];
                P = A*P*A' + D*QN2*D'; % P correction
                % Measurement update
                ytpred = C*x1 + E*ut + dyt;
                Kf = P*C'/(C*P*C'+RN);
                % State estimate update
                % x(t|t) = x(t|t-1) + Kf_t*(y(t) - y(t|t-1))
                err = ymt - ytpred;
                xtt = x1 + Kf*err;
                xErr = (xtt-x1)*(xtt-x1)';
                P = cov(xErr);
                P = P - Kf*C*P; % P prediction
                P = 0.5*(P+P'); % make sure P matrix is symmetric
                sys = [xtt;p;ytpred];

        case 2
                sys = [];
        case 0
                struct2vars(p1)
                s = simsizes;
                s.NumContStates = 0;
                s.NumDiscStates = 8;
                s.NumOutputs = 14;
                s.NumInputs = 10;
                s.DirFeedthrough = 10;
                s.NumSampleTimes = 1;
                sys = simsizes(s);
                x0 = [0 0 0 0 0 0 0 0]';
                str = [];
                ts  = [Ts Ts*.5];
                P = 1e1*diag([1 1 1 1,  1 1,   1 1]);% size x0
                xtt = x0;
                QN2 = 0.05*eye(6);
                RN = eye(2);

        otherwise
                sys = [];
                x0 = [];
end
```

C.4 RS-NGMV

The RS-NGMV block contains the s-function rsngmv.m given below. It calls the sdysd.m as well to access some model matrices. The s-function parameters (p1 struct loaded after main.m). It takes LPVKF estimations, RS terms $F_e(t)$ and dynamic weighting F_{ck} for inputs and return the gains and the control signals for the links.

```
function [sys1,x01,str1,ts] = rsngmv(t,x,u,flag,p1)
persistent Ap Bp Cp Ep Cd Ed Ts kc a kp kI kd
persistent lambdap lambdad lambdau lambdak nd
switch flag
    case 3
        [xtt1,xtt2,xtt3,xtt4,xtt5,xtt6,xtt7,xtt8,ym1,ym2,ym1_dot,ym2_dot,
         r1,r2,Wd1,Wd2,fe11_1,fe11_2,fe11_3,fe22_1,fe22_2,fe22_3,
        Fck1,Fck2] = vsplit(u);
        d = [Wd1 Wd2]';
        ymt = [ym1 ym2]';
        ymt_dot = [ym1_dot ym2_dot]';
        [q1,q2] = vsplit(ymt(1:2));
        [q1dot,q2dot] = vsplit(ymt_dot(1:2));
        p = [q1;q2;q1dot;q2dot];
        xhat = [xtt1,xtt2,xtt3,xtt4,xtt5,xtt6,xtt7,xtt8]';
        rt = [r1 r2]';

        fe_11 = [fe11_1,fe11_2,fe11_3];
        fe_22 = [fe22_1,fe22_2,fe22_3];
        ef1 = [fe_11 zeros(1,9)];
        ef2 = [zeros(1,9) fe_22];
        Fe = [ef1;ef2];
%allocating space for RS gains.
        if t < 0.1
            kc_11 = zeros(3,1);
            kc_12 = zeros(3,1);
            kc_21 = zeros(3,1);
            kc_22 = zeros(3,1);
            kc_1 = [kc_11;kc_12];
            kc_2 = [kc_21;kc_22];
            kc = [kc_1;kc_2];
        else
            [A0,B0,G0,C0,E0,H0,Cm,Em,Hm] = sysd(p,p1,Ts);
            C = [-Ep*C0, -Ep*Cd, Cp];
            dp = Ep*(rt-d);
            dxt = [0;0;0;0];
            % measurable dist.
            dd  = [dxt; zeros(nd,1); Bp*(rt - d)];
            dpd = dp + C*dd;
            kc_bar = [kp;kI;kd;zeros(6,1);kp;kI;kdt];
            Pp = Fe'*Ep'*lambdap;
            phi_k = -lambdak*kc_bar - lambdad*kc;
            W1k = eye(2);
            Fck = diag([Fck1 Fck2]);
            Fck = Fck;
            % alternative kc from eq. 54
            X0 = lambdak + lambdad + Fe'*(Fck + (Ep'*lambdap*Ep + lambdau)*W1k)*
                Fe;
            kc= -inv(X0)*(Pp*(dpd + C*xhat) + phi_k);
            u_rs = Fe*kc;
            sys = [kc;u_rs];
        end
    case 2
        sys1 = [];
    case 0
        struct2vars(p1)
        s = simsizes ;
        s.NumContStates = 0 ;
        s.NumDiscStates = 0 ;
        s.NumOutputs = 14;
        s.NumInputs = 24;
        s.DirFeedthrough =24;
        s.NumSampleTimes = 1 ;
        sys1 = simsizes(s) ;
        x01 = [];
        kc_11= [k1 k2 k3]';
        kc_12= [k1 k2 k3]';
        kc_21= [k1 k2 k3]';
        kc_22= [k1 k2 k3]';
        str = [];
        ts  = [Ts Ts*.5];
    otherwise
        sys = [];
        x0 = [];
end
```

Appendix D

Autonomous Vehicle Steering

```
function xdot = fcn(Vx,x,u)
% Model parameters for the LPV plant
m = 1575;
Iz = 2875;
lf = 1.2;
lr = 1.6;
Cf = 19000;
Cr = 33000;
% Continuous-time model
A = [-(2*Cf+2*Cr)/m/Vx, 0, -Vx-(2*Cf*lf-2*Cr*lr)/m/Vx, 0;
      0, 0, 1, 0;
      -(2*Cf*lf-2*Cr*lr)/Iz/Vx, 0, -(2*Cf*lf^2+2*Cr*lr^2)/Iz/Vx, 0;
      1, Vx, 0, 0];
B = [2*Cf/m 0 2*Cf*lf/Iz 0]';
C = [0 0 0 1; 0 1 0 0];
D = zeros(2,1);
xdot = A*x + B*u;
%y = C*x + D*u;
```

The LPV model used by the Mathworks MPC demos and the RS-NGMV comparisons in chapter 4 is presented in the script below. For RS-NGMV it is modified under sys.m/sysd.m scripts with the standard linear plant modelling style presented earlier. The Adaptive MPC and RS-NGMV control diagrams are shown in Fig. D.1 and Fig. D.2, respectively. The RS-NGMV files have been created following the methodologies introduced in Appendix B and C.

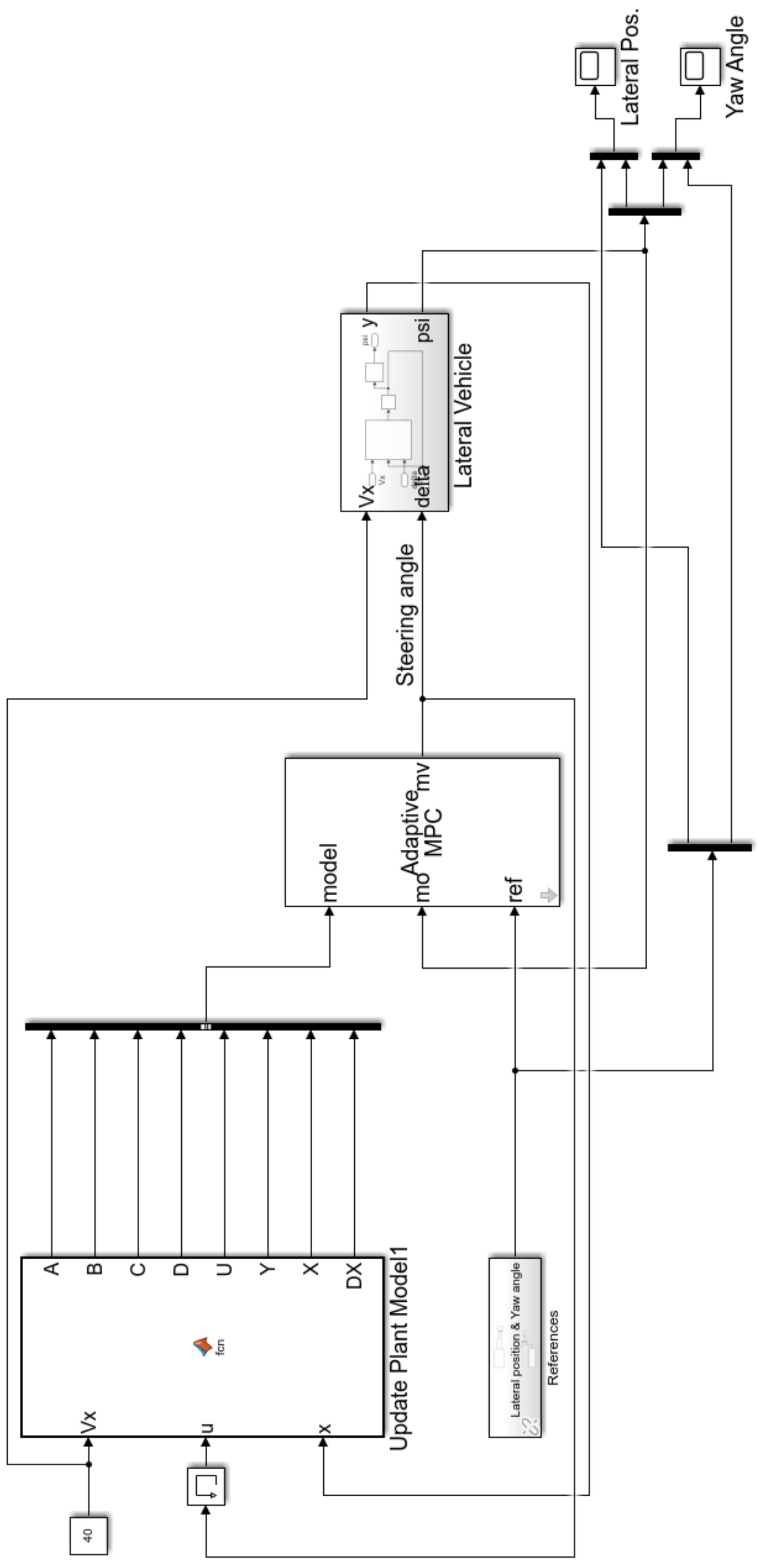


Figure D.1: Adaptive MPC control diagram.

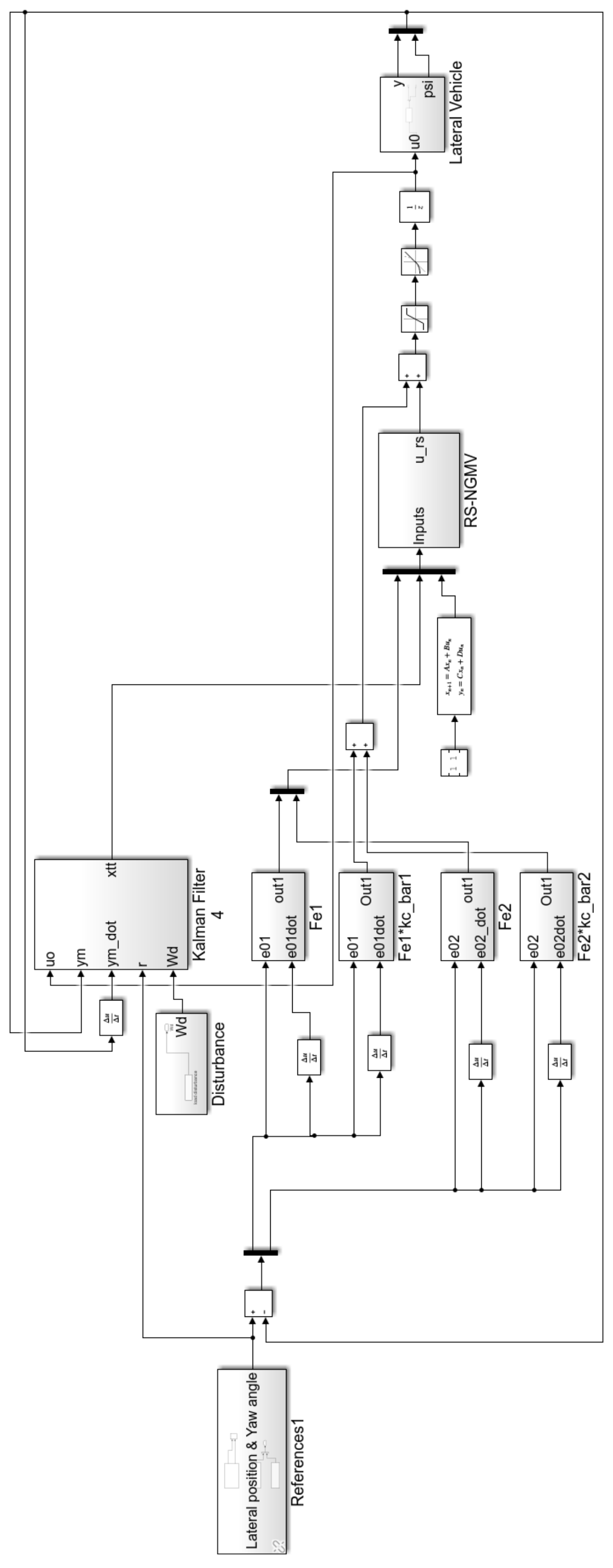


Figure D.2: RS-NGMV control diagram.

D.1 LPV-RE Preview Control

The LPV-RE Preview control diagram for the case study in chapter 6 is found in Fig. D.3. The implementation methodology is a bit different than that of the standard RS-NGMV's. The study uses code pieces from the Sharp's Preview controller implemented in [118], modified for the RS-NGMV implementation methodology of this thesis. Unique scripts and code chunks are presented starting with the reference generation.

The road reference for the lane changing were modelled using linspace function and lagged for N_p for the preview controller.

```
Ref = [zeros(1,2/Ts) linspace(0,3,4/Ts) 3*ones(1,10/Ts)];
% save('ref.mat','Ref');
%lag the reference vector for preview
Ref_lagged = [Ref(1)*ones(1,Np) Ref(1:end-Np)];
```

The lane changing was visualized using maneuver.m and Automated Driving System Toolbox of Mathworks. A common way of creating road reference is by using line segments or defining linearly placed way points like in maneuver.m and store the data as .mat files for the actual simulations. The script was modified by following the approaches of Mathworks demo files such as the Stanley controller and the MPC demo shown previously. When the script is run, it visualizes the lane change maneuver for the study in this thesis. However, it can be observed that it is a bit cumbersome how the way points are defined. For this reason, the maneuver.m was only used for visualizations and linspace approach was preferred for simulations. As mentioned earlier in the thesis, clothoids are promising and mathematically accurate methods for modelling of roads or tracks. An example is the clothoids toolbox available in [119]. It was considered for the thesis as well but was not continued further due its complexity and the fact that linspace approach was sufficient.

```
function [allData, scenario, sensor] = maneuver()
%maneuver - Returns sensor detections
```

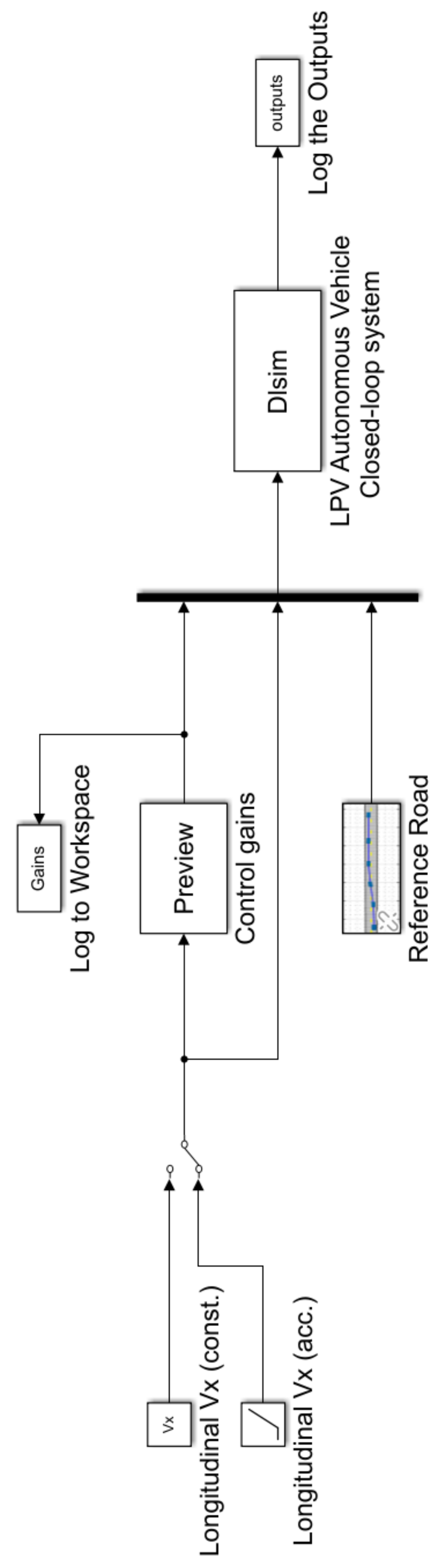


Figure D.3: LPV-RE Preview control diagram.

```
%    allData = maneuver returns sensor detections in a structure
%    with time for an internally defined scenario and sensor suite.
%    [allData, scenario, sensors] = maneuver optionally returns
%    the drivingScenario and detection generator objects.
% Generated by MATLAB(R) 9.5 and Automated Driving System Toolbox 1.3.
% Generated on: 26-Feb-2019 13:40:37
% Create the drivingScenario object and ego car
[scenario, egoCar] = createDrivingScenario;
% Create all the sensors
sensor = createSensor(scenario);
allData = struct('Time', {}, 'ActorPoses', {}, 'ObjectDetections', {}, '
    LaneDetections', {});
running = true;
while running
    % Generate the target poses of all actors relative to the ego car
    poses = targetPoses(egoCar);
    time  = scenario.SimulationTime;
    % Generate detections for the sensor
    laneDetections = [];
    [objectDetections, numObjects, isValidTime] = sensor(poses, time);
    objectDetections = objectDetections(1:numObjects);
    % Aggregate all detections into a structure for later use
    if isValidTime
        allData(end + 1) = struct( ...
            'Time',       scenario.SimulationTime, ...
            'ActorPoses', actorPoses(scenario), ...
            'ObjectDetections', {objectDetections}, ...
            'LaneDetections',   {laneDetections}); %#ok<AGROW>
    end
    % Advance the scenario one time step and exit the loop if the scenario is
        complete
    running = advance(scenario);
end
% Restart the driving scenario to return the actors to their initial positions.
restart(scenario);
% Release the sensor object so it can be used again.
release(sensor);
%%%%%%%%%%%%%%%%%%%%
% Helper functions %
%%%%%%%%%%%%%%%%%%%%
% Units used in createSensors and createDrivingScenario
% Distance/Position - meters
% Speed             - meters/second
% Angles              - degrees
% RCS Pattern         - dBsm
function sensor = createSensor(scenario)
% createSensors Returns all sensor objects to generate detections
% Assign into each sensor the physical and radar profiles for all actors
profiles = actorProfiles(scenario);
sensor = visionDetectionGenerator('SensorIndex', 1, ...
    'SensorLocation', [3.7 0], ...
    'MaxRange', 100, ...
    'DetectorOutput', 'Objects only', ...
    'Intrinsics', cameraIntrinsics([1814.81018227767 1814.81018227767],[320
        240],[480 640]), ...
    'ActorProfiles', profiles);
function [scenario, egoCar] = createDrivingScenario
% createDrivingScenario Returns the drivingScenario defined in the Designer
% Construct a drivingScenario object.
scenario = drivingScenario;
% Add all road segments
roadCenters = [-15 -0.1 0;
              125.6 0.2 0];
marking = [laneMarking('Solid')
    laneMarking('Dashed', 'Color', [1 1 0])
    laneMarking('Solid')];
laneSpecification = lanespec(2, 'Width', 4, 'Marking', marking);
road(scenario, roadCenters, 'Lanes', laneSpecification);
% Add the ego car
egoCar = vehicle(scenario, ...
    'ClassID', 1, ...
    'Position', [-10 0 0]);
waypoints =   [-10.0000         -2          0;
                -8.5000    -0.0003-2          0;
                -7.0000     0.0001-2          0;
                -5.5000     0.0020-2          0;
                -4.0000     0.0059-2          0;
                -2.5000     0.0127-2          0;
                -1.0001     0.0230-2          0;
                 0.4999     0.0375-2          0;
                 1.9997     0.0566-2          0;
                 3.4996     0.0808-2          0;
                 4.9993     0.1107-2          0;
                 6.4988     0.1471-2          0;
                 7.9982     0.1908-2          0;
                 9.4973     0.2424-2          0;
                10.9961   0.3027-2        0;
                12.4944   0.3725-2        0;
                13.9923   0.4524-2        0;
                15.4896   0.5432-2        0;
                16.9861   0.6448-2        0;
                18.4820   0.7558-2        0;
                19.9773   0.8746-2        0;
                21.4721   0.9996-2        0;
                22.9664   1.1295-2        0;
                24.4605   1.2626-2        0;
                25.9544   1.3974-2        0;
                27.4483   1.5325-2        0;
                28.9424   1.6663-2        0;
                30.4365   1.7984-2        0;
                31.9309   1.9288-2        0;
                33.4253   2.0572-2        0;
                34.9200   2.1836-2        0;
                36.4149   2.3076-2        0;
                37.9099   2.4291-2        0;
                39.4052   2.5481-2        0;
                40.9007   2.6642-2        0;
                42.3964   2.7774-2        0;
                43.8924   2.8874-2        0;
                45.3886   2.9940-2        0;
                46.8851   3.0966-2        0;
                48.3819   3.1949-2        0;
                49.8790   3.2885-2        0;
                51.3763   3.3769-2        0;
                52.8740   3.4599-2        0;
                54.3721   3.5369-2        0;
                55.8704   3.6075-2        0;
                57.3690   3.6714-2        0;
                58.8680   3.7282-2        0;
                60.3671   3.7778-2        0;
                61.8665   3.8208-2        0;
                63.3661   3.8576-2        0;
                64.8657   3.8890-2        0;
                66.3655   3.9153-2        0;
                67.8653   3.9372-2        0;
                69.3652   3.9551-2        0;
                70.8652   3.9696-2        0;
                72.3651   3.9812-2        0;
                73.8651   3.9905-2        0;
```

```
                     75.3651     3.9981−2            0;
                     76.8651     4.0042−2            0;
                     78.3651     4.0090−2            0;
                     79.8651     4.0124−2            0;
                     81.3651     4.0147−2            0;
                     82.8651     4.0159−2            0;
                     84.3651     4.0163−2            0;
                     85.8651     4.0159−2            0;
                     87.3651     4.0148−2            0;
                     88.8650     4.0132−2            0;
                     90.3650     4.0113−2            0;
                     91.8650     4.0090−2            0;
                     93.3650     4.0067−2            0;
                     94.8650     4.0044−2            0;
                     96.3650     4.0022−2            0;
                     97.8650     4.0003−2            0;
                     99.3650     3.9988−2            0;
                    100.8650     3.9977−2            0;
                    102.3650     3.9971−2            0;
                    103.8650     3.9968−2            0;
                    105.3650     3.9969−2            0;
                    106.8650     3.9971−2            0;
                    108.3650     3.9976−2            0;
                    109.8650     3.9982−2            0;
                    111.3650     3.9988−2            0;
                    112.8650     3.9995−2            0];
speed = 15; %Might be changed but heading angle remains the same.
trajectory(egoCar, waypoints, speed);
plot(scenario)
```

The gains of the preview controller is calculated in s-function preview.m.

```
function [sys,x0,str,ts] = preview(t,x,u,flag,p1)
persistent Ts Vx
persistent Np Ref
switch flag
    case 3
        Vx = u;
        p = Vx;
        [Aaug,Baug,B2aug, X0,Ad,Bd, C, G,Cc,Ec,Hc,Cm,Em,Hm] = sysd(p,p1,Ts);
        Caug = zeros(2,Np+4);
        Caug(:,1:6) = [1 0 0 0 −1 0;0 0 1 0 1/(Vx*Ts) −1/(Vx*Ts)];
        Q = [.1 0; 0 0];
        R1 = Caug'*Q*Caug;
        R2 = 1;
        [Kf,Sf,Ef] = dlqr(Aaug,Baug,Caug'*Q*Caug,1);
        sys1 = Kf;
    case 2
        sys1 = [];
    case 0
        struct2vars(p1)
        s = simsizes ;
        s.NumContStates = 0 ;
        s.NumDiscStates = 0 ;
        s.NumOutputs = 4+Np;
        s.NumInputs = 1;
        s.DirFeedthrough = 4+Np;
        s.NumSampleTimes = 1 ;
        sys1 = simsizes(s) ;
        x0 = [];
        str = [];
        ts  = [Ts Ts*.5];
    otherwise
        sys = [];
        x0 = [];
end
```

The plant and the road model is given in sys.m.

```
function [A, B, C, D, Ar, Br, G, Cc, Ec, Hc, Cm, Em, Hm] = sys(p,p1)
struct2vars(p1)
Vx = p;
%vehicle state vector is [y ydot psi psidot]'
A = [0 1 0 0;...
    0 -(Cf+Cr)/(m*Vx) (Cf+Cr)/m (b*Cr-a*Cf)/(m*Vx);...
    0 0 0 1;...
    0 (b*Cr-a*Cf)/(Iz*Vx) (a*Cf-b*Cr)/Iz -(a^2*Cf+b^2*Cr)/(Iz*Vx)];
B = [0 Cf/(G*m) 0 a*Cf/(G*Iz)]';
C = [1 0 0 0; 0 0 0 0];
D = [0 0]';
Ar = zeros(Np);%allocate space
Ar(1:end-1,2:end) = eye(Np-1);%set up the shift register
Br = zeros(Np,1);%input vector for shift register
Br(end) = 1;
G = [0; 0; 0; 0];
% Construct controlled outputs
```

```
Cc = [0 0 0 1;0 1 0 0];
Ec = [0 0]';
Hc = 0;
% Construct measured outputs
Cm = [0 0 0 1;0 1 0 0];
Em = [0 0]';
Hm = 0;
```

The model in sys.m is discretized and used to build the augmented form in sysd.m.

```
function [Aaug, Baug, B2aug, X0, Ad, Bd, C, G, Cc, Ec, Hc, Cm, Em, Hm] = sysd(p,
    p1,Ts)
% struct2vars(p1)
nd = 1;
[A,B, C, D, Ar, Br, G, Cc, Ec, Hc, Cm, Em, Hm] = sys(p,p1);
% Model discretization
if (pr1.c2d_flag==0)
    sysd = c2d(ss(A,B,C,D),Ts); %discrete time state space description.
    Ad = sysd.a;
    Bd = sysd.b;
    Aaug = blkdiag(Ad,Ar);%automatically creates the big a matrix
    Baug = [Bd_sdre;zeros(size(Br))];%input vector for steering angle
    B2aug = [zeros(size(Bd));Br];%input vector for road
    X0 = zeros(size(Baug));%set initial conditions to zero
elseif (pr1.c2d_flag==1)
        XX = [A B G; zeros(p1.nu+nd,p1.nx0+p1.nu+nd)];
        YY = expm(XX*Ts);
        A0 = YY(1:p1.nx0,1:p1.nx0);
        B0 = YY(1:p1.nx0,p1.nx0+1:p1.nx0+p1.nu);
        G0 = YY(1:p1.nx0,p1.nx0+p1.nu+1:end);
end
```

The closed-loop discrete-time state-space system was simulated using the dlsim(A,B,C,D,U,X0) function of Matlab used within the dlsim.m s-function. The function accepts the state matrices, reference and initial states in the given order.

```
function [sys,x0,str,ts] = dlsim(t,x,u,flag,p1)
persistent Ts Vx Ts t
persistent Np Vx Kf
switch flag
    case 3
        Kf = u(1:4+Np)';
        U_sdre = u(1+4+Np);
        p = Vx;
        C_sim= Kf;
        [Aaug, Baug, B2aug, X0, Ad, Bd, C, G, Cc, Ec, Hc, Cm, Em, Hm] = sysd(p,
            p1,Ts);
        [y,x] = dlsim(Aaug-Baug*Kf,B2aug,C_sim,0,r,X0);
        sys1 = y(:,1);
    case 2
        sys = [];
    case 0
        struct2vars(p1)
        s = simsizes ;
        s.NumContStates = 0 ;
        s.NumDiscStates = 0 ;
        s.NumOutputs = 800;
        s.NumInputs = 2+4+Np;
        s.DirFeedthrough =2+4+Np;
        s.NumSampleTimes = 1 ;
        sys1 = simsizes(s) ;
        x01 = [];
        str1 = [];
        ts  = [Ts Ts*.5];
    otherwise
        sys = [];
        x0 = [];
end
```

Appendix E

Electric Vehicle

The RS-NGMV control diagram for the EV control is given in Fig. E.1. The expanded EV powertrain and the longitudinal vehicle plant is given in Fig. E.2. The EV dynamics, components and the baseline PI controller have been adapted from [104] and modified for this study. The road grade profile for UDDS is created in the main.m where parameters are assigned in the vehicle struct.

```
inclinationAngle = 0.1;
freq = 0.001;
[hill] = genHill(inclinationAngle,freq,TimeUDDS);
timeHill = (0:TimeUDDS)';
waveHill.time = timeHill;
waveHill.signals.values = hill';
```

The RS-NGMV coding approach follows from the principles introduced earlier. The lpvkf.m and rsngmv.m files have some important differences for the EV application due to the disturbance factor characteristic to the case.

```
function [sys,x0,str,ts] = lpvkf(t,x,u,flag,vehicle)
persistent P x0tt xtt QN RN
persistent Ts
persistent A0 B0 D0 G0 C0 E0 H0 C0m E0m H0m Ap Bp Cp Ep Ad Bd Cd Ed A B C Ar
Ts = 1;
switch flag
   case 3
         [u, actualSpeed, roadGrade, r] = vsplit(u);
```

```
        theta = roadGrade;
        x0tt = actualSpeed;
        y0t = actualSpeed;
        ut = u; %feedback input
        desSpeed = r; %reference
        p = x0tt;
        [A0,B0,D0,G0,C0,E0,H0,C0m,E0m,H0m] = sys(p,vehicle,Ts);
        %Augmented system notes
        % states x0, xd, xp, xr
        % disturbances dd = [d0d rd - d]' rd deterministic ref. recall d
        % is the deterministic part of d0 output disturbance.
        % recall also augmented error model's dp = Ep(rd-d)
        % stochastic xi = [xi0 Wd Wr]'
        %Consider predictions now for the KF,
        %predicted known disturbance ddd (deterministic)
        nxp = 1; %for error weight on states
        nxr = 1; %for this example we do not consider reference model (matrices
            full zeros)
        nxd = 1; %for this example we do not consider stochastic output dist.(
            matrices full zero)
        nx0 = 1; %number of states
        Ar = zeros(nxr,nxr); Br = zeros(nxr,1); Cr = zeros(nxr,nxr); Er = zeros(
            nxr,1);
        %assigned in struct 'vehicle' inside main.m after
        %using ssdata(Pck)..
        Ap = vehicle.driveControlParams.Ap_longitudinal;
        Bp = vehicle.driveControlParams.Bp_longitudinal;
        Cp = vehicle.driveControlParams.Cp_longitudinal;
        Ep = vehicle.driveControlParams.Ep_longitudinal;
        %assigned in struct 'vehicle' inside main.m after
        %using Wd = .1*eye(nyc)*filt([0 1],[1 -0.98],Ts);
        %ssdata(Wd)...
        Ad = vehicle.driveControlParams.Ad_longitudinal;
        Bd = vehicle.driveControlParams.Bd_longitudinal;
        Cd = vehicle.driveControlParams.Cd_longitudinal;
        Ed = vehicle.driveControlParams.Ed_longitudinal;
        A = [A0,          0,          0,      0;
               0,        Ad,          0,      0;
          -Bp*C0,     -Bp*Cd,     Ap,      0;
                0,            0,      0,     Ar];
        B = [B0;          0;      -Bp*E0;  0];
        C = [C0,          Cd,        0,      0]; % [C0 Cd 0 0]
        rd = desSpeed;
        d = 0; %in this case we do not measure the output disturbance so can
            assign 0
        g = 9.81;
        d0d = -g*theta; %input dist.
        ddt = [G0*d0d; 0; Bp*(rd - d); 0]; % [G0 0; 0 0 ; 0 Bp; 0 0]*[d0d (rd-d)
            ]'
        %State update
        xtt = [x0tt;  0; 0; 0]; %x0 xd xp xr
        x1 = A*xtt + B*ut +ddt;
        P = A*P*A' + QN*eye(4);
        % Measurement update
        ytpred = C*x1 + E0m*ut;
        Kf = P*C'/(C*P*C'+RN);
        % State estimate update
        % x(t|t) = x(t|t-1) + Kf_t*(y(t) - y(t|t-1))
        err = y0t - ytpred;
        xtt = x1 + Kf*err;
        P = P - Kf*C*P;
        P = 0.5*(P+P');
        sys = [xtt];
    case 2
        sys = [];
    case 0
        struct2vars(vehicle)
        s = simsizes ;
        s.NumContStates = 0 ;
        s.NumDiscStates = 4;
        s.NumOutputs = 4;
        s.NumInputs = 4;
        s.DirFeedthrough = 4;
        s.NumSampleTimes = 1 ;
        sys = simsizes(s) ;
        x0 = [0 0 0 0]';
        str = [];
        ts = Ts;
        P = 1*diag([1 1, 1 1]);
        xtt = x0;
        QN = 1*eye(4);
        RN = .5*eye(1);
        otherwise
                sys = [];
                x0 = [];
end
```

The RS-NGMV s-function is then given in the script below.

```
function [sys,x0,str,ts] = rsngmv(t,x,u,flag,vehicle)
persistent Ts kc k1_bar k2_bar k3_bar
Ts = 1;
switch flag
    case 3
        [fe11_1,fe11_2,fe11_3,u,x1,x2,x3,x4,r,Fck] = vsplit(u);
        xhat = x1;
        p = xhat;
        xtt = [xhat; x2; x3; x4]; % add in xd and xp (dist. and error states)
        Fe = [fe11_1,fe11_2,fe11_3];
        if t < 0.1
            kc= zeros(3,1);
        else
            [A0,B0,D0,G0,C0,E0,H0,C0m,E0m,H0m] = sys(p,vehicle,Ts);
            Ap = vehicle.driveControlParams.Ap_longitudinal;
            Bp = vehicle.driveControlParams.Bp_longitudinal;
            Cp = vehicle.driveControlParams.Cp_longitudinal;
            Ep = vehicle.driveControlParams.Ep_longitudinal;
            Cd = vehicle.driveControlParams.Cd_longitudinal;
            Cr = 0;
            Er = 0;
            Cpt = [-Ep*C0, -Ep*Cd, Cp, Cr];
            Ept = -Ep*E0;

            k1_bar = vehicle.driveControlParams.k1_bar;
            k2_bar = vehicle.driveControlParams.k2_bar;
            k3_bar = vehicle.driveControlParams.k3_bar;
            kc_bar = [k1_bar;k2_bar;k3_bar];

            lambdap = vehicle.driveControlParams.lambdap_longitudinal;
            lambdad = vehicle.driveControlParams.lambdad_longitudinal;
            lambdak = vehicle.driveControlParams.lambdak_longitudinal;
            lambdau = vehicle.driveControlParams.lambdau_longitudinal;

            Pp = Fe'*Ept'*lambdap^2;
            phi_k = -lambdak^2*kc_bar - lambdad^2*kc;
            W1k = 1;
            rd = r; %the reference is wholly deterministic and since the ref.
                model is 2x2 0 is added for symmetry.
            d = 0;
```

```
                g = 9.81;
                d0d = vehicle.equivMass*g;
                        ddt = [G0*d0d;0;Bp*(rd - d);0]; % [G0 0; 0 0 ; 0 Bp; 0 0]*[
                            d0d (rd-d)]'
                dp = Ept*(rd-d); %consider feedforward terms for d, blank for now.
                dpd = dp + Cpt*ddt;
                X0 = lambdak^2 + lambdad^2+ Fe'*(Fck + ...
                (Ept'*lambdap^2*Ept + lambdau^2)*W1k)*Fe;
                kc = -inv(X0)*(Pp*(dpd + Cpt*xtt) + phi_k);
                kc_out = [kc(1) kc(2) kc(3)]';
                u_rs = Fe*kc;
                sys1 = [kc_out;u_rs];
            end
    case 2
        sys1 = [];
    case 0
        struct2vars(vehicle)
        s = simsizes ;
        s.NumContStates = 0 ;
        s.NumDiscStates = 0 ;
        s.NumOutputs = 4;
        s.NumInputs = 10;
        s.DirFeedthrough = 10;
        s.NumSampleTimes = 1 ;
        sys1 = simsizes(s) ;
        x01 = [];
        str1 = [];
        ts  = [-1 0];
    otherwise
        sys = [];
        x0 = [];
end
```

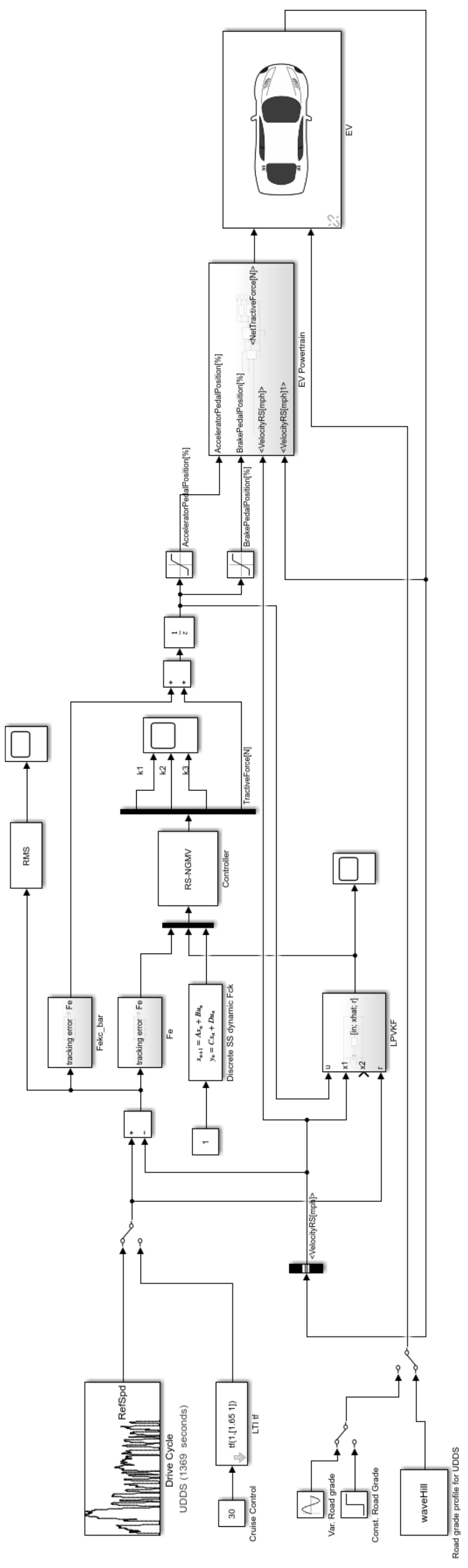


Figure E.1: RS-NGMV control diagram.

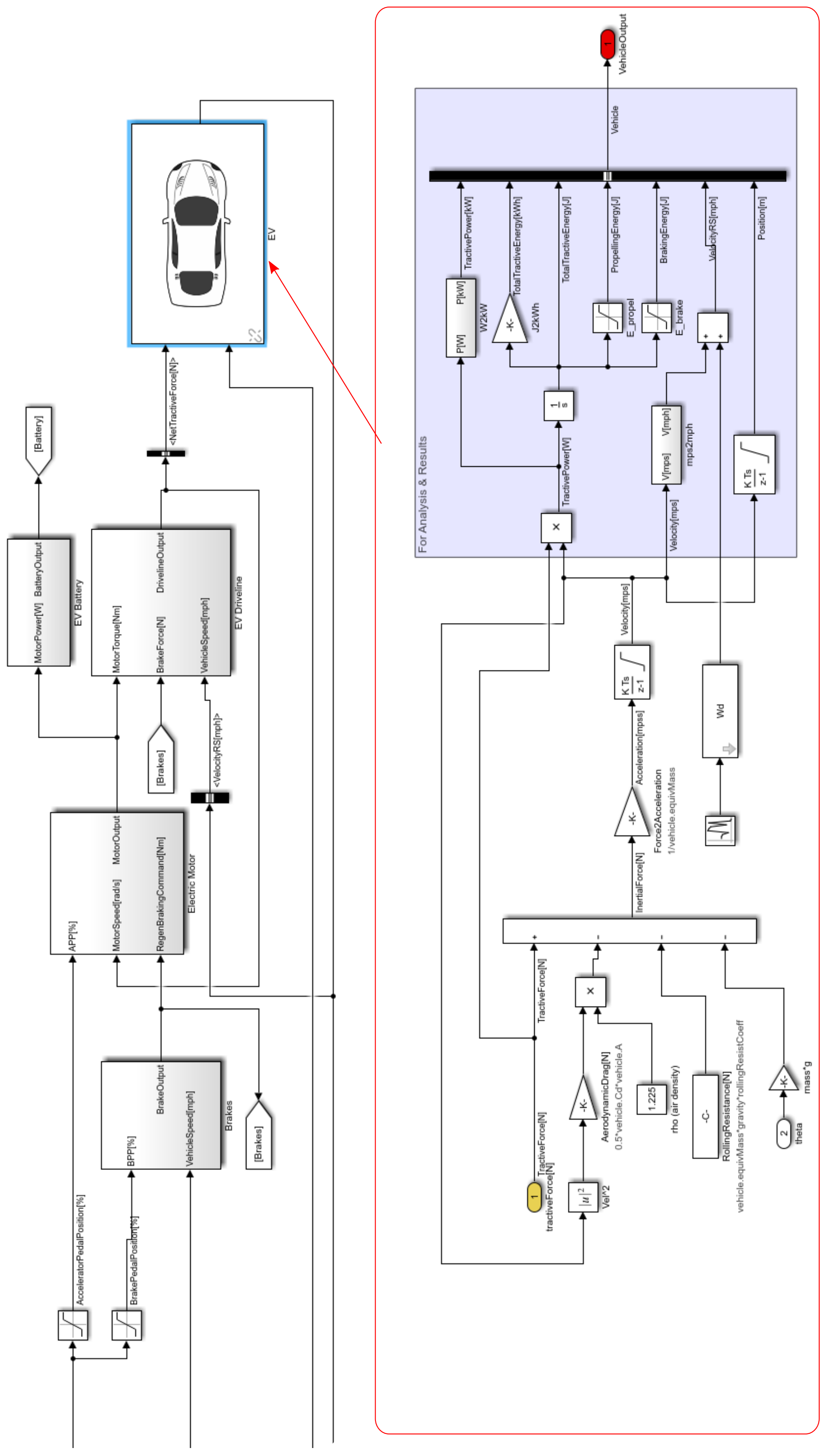


Figure E.2: EV model.

Appendix F

Scheduled RS-NGMV

F.1 Collecting Controller Gains

Before using the Scheduled RS-NGMV, the script “collectgains.m” must collect the RS-NGMV gains of the previous run from the desired time instants “ctimes” and store in 1-D look-up tables.

```
%% Collect gains from a Scenario and save into LUTs
% PID structure: kp, ki, kd, k2
scen = 1;
switch scen
        case 1
                ctimes = [0.3, 7.1]';
                [ntimes1,ntimes2,kp_lut,ki_lut,kd_lut,k2_lut,N_lut] = deal(zeros
                    (size(ctimes)));
                for i=1:length(ctimes)
                        ntimes1(i) = find(ScopeK.time >= ctimes(i),1,'first');
                        ntimes2(i) = find(ScopeOUT.time >= ctimes(i),1,'first');
                        N_lut(i) = ScopeOUT.signals(1).values(ntimes2(i),1);
                        kp_lut(i) = ScopeK.signals(1).values(ntimes1(i));
                        ki_lut(i) = ScopeK.signals(2).values(ntimes1(i));
                        kd_lut(i) = ScopeK.signals(3).values(ntimes1(i));
                        k2_lut(i) = ScopeK.signals(4).values(ntimes1(i));
                end
                save RSdata N_lut kp_lut ki_lut kd_lut k2_lut
end
```

This is just the initial approach to generate the data. The script could be modified to collect from many more time instants and store the data. If there are already enough data collected from previous RS-NGMV runs, the .mat files can be used directly on the Scheduled RS-NGMV without having a need to run the RS-NGMV first. It could be advantageous for comparisons to figure out the best values.

The variable names ScopeK and ScopeOUT are defined from the actual scopes in the Simulink model by choosing logging tab from configuration properties. This way, the data is logged from the desired parameter during Simulink runs.

The SI engine Simulink model is found in Fig. F.1. The RS-NGMV control diagram for chapter 8 using the SI engine is found in Fig. F.2 adapted from "sldemo_enginewc". The scheduled RS control diagram and the controller are given in Fig. F.3 and Fig. F.4. Note that Fig. F.4 is the portion starts from input ports 4 and 5, this is because the main Scheduled RS block also has LPVKF in it but was not used in the Scheduled RS controller. It was included for work in progress. The Matlab/Simulink design style is pretty much the same apart from the details given here or in chapter 8. The main.m parametrises the system, the linearisation is made by modifying the Mathworks SI engine demo files, the linearised plant model is coded in sys.m and called by s-functions of kalman filter and RS-NGMV, lpvkf.m and rsngmv.m.

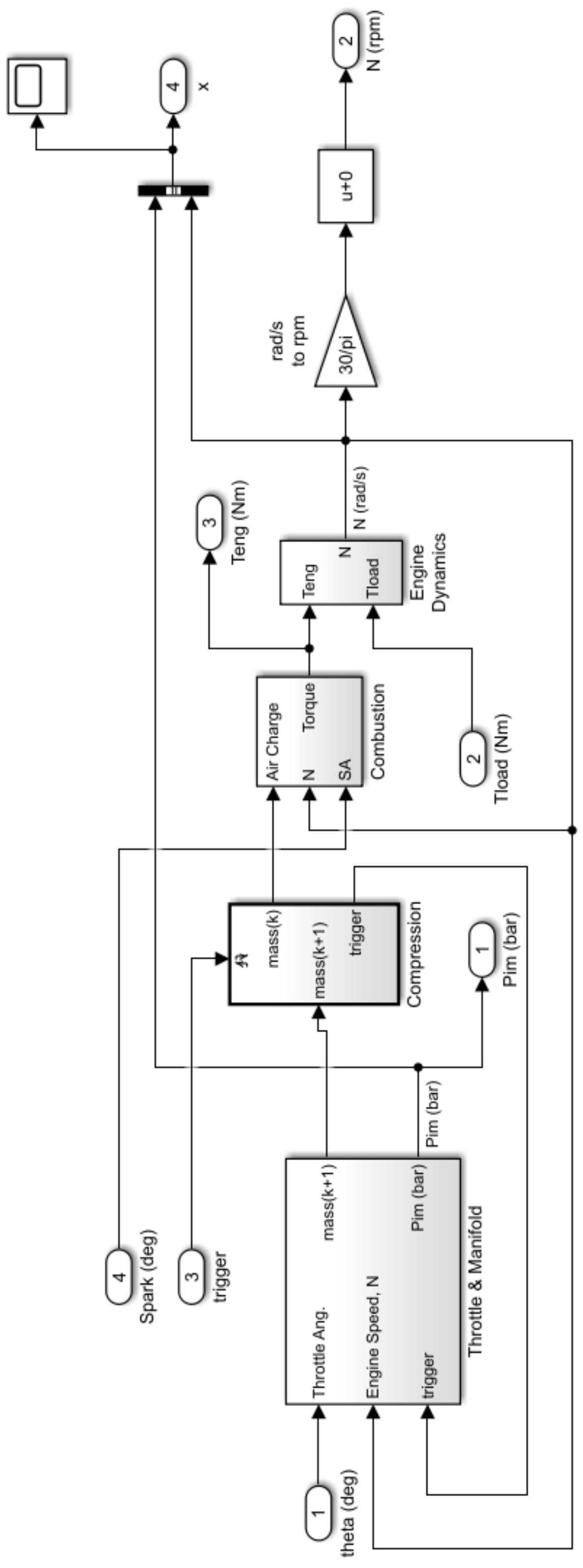


Figure F.1: SI engine Simulink diagram.

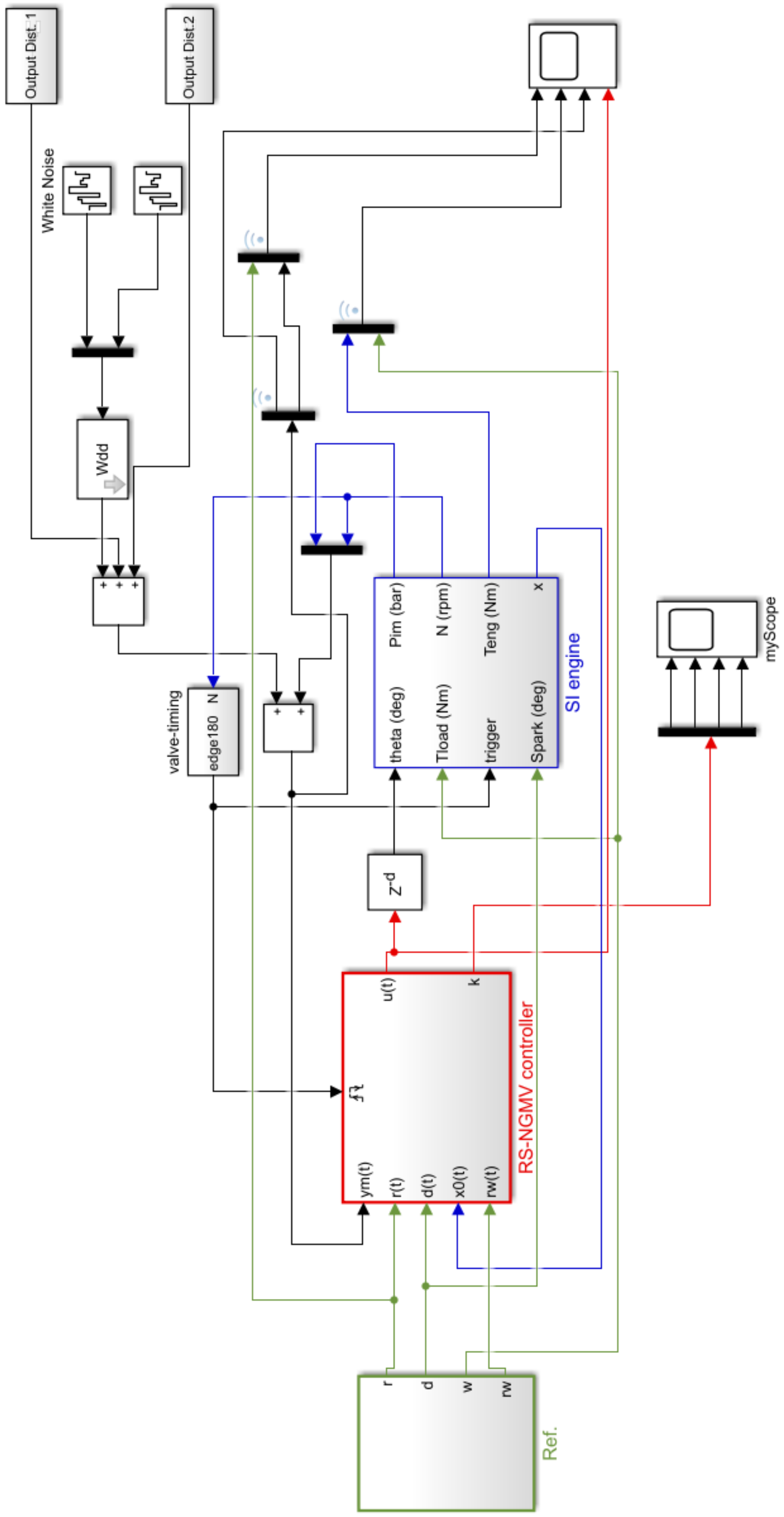


Figure F.2: RS-NGMV Simulink diagram.

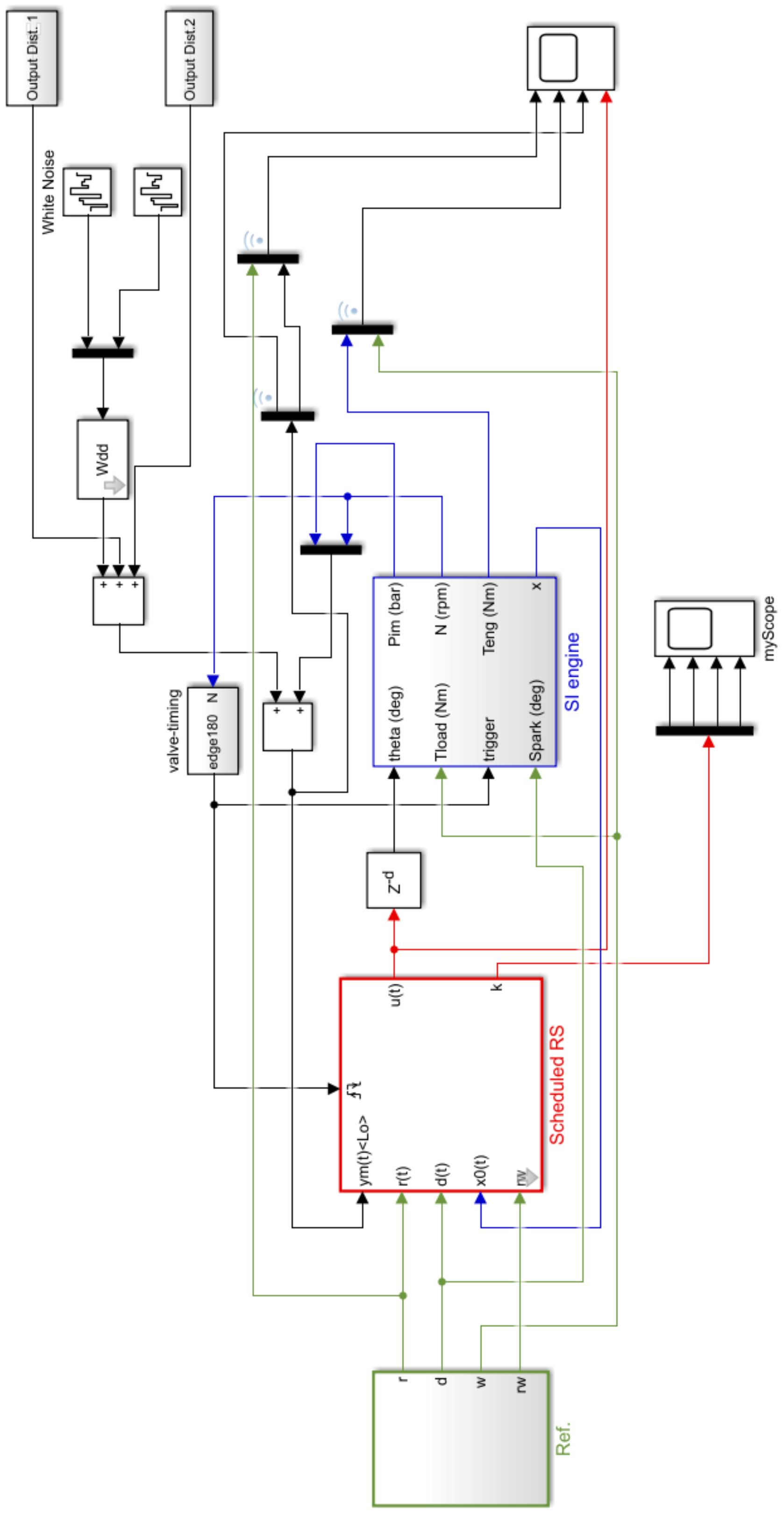


Figure F.3: Scheduled RS Simulink diagram.

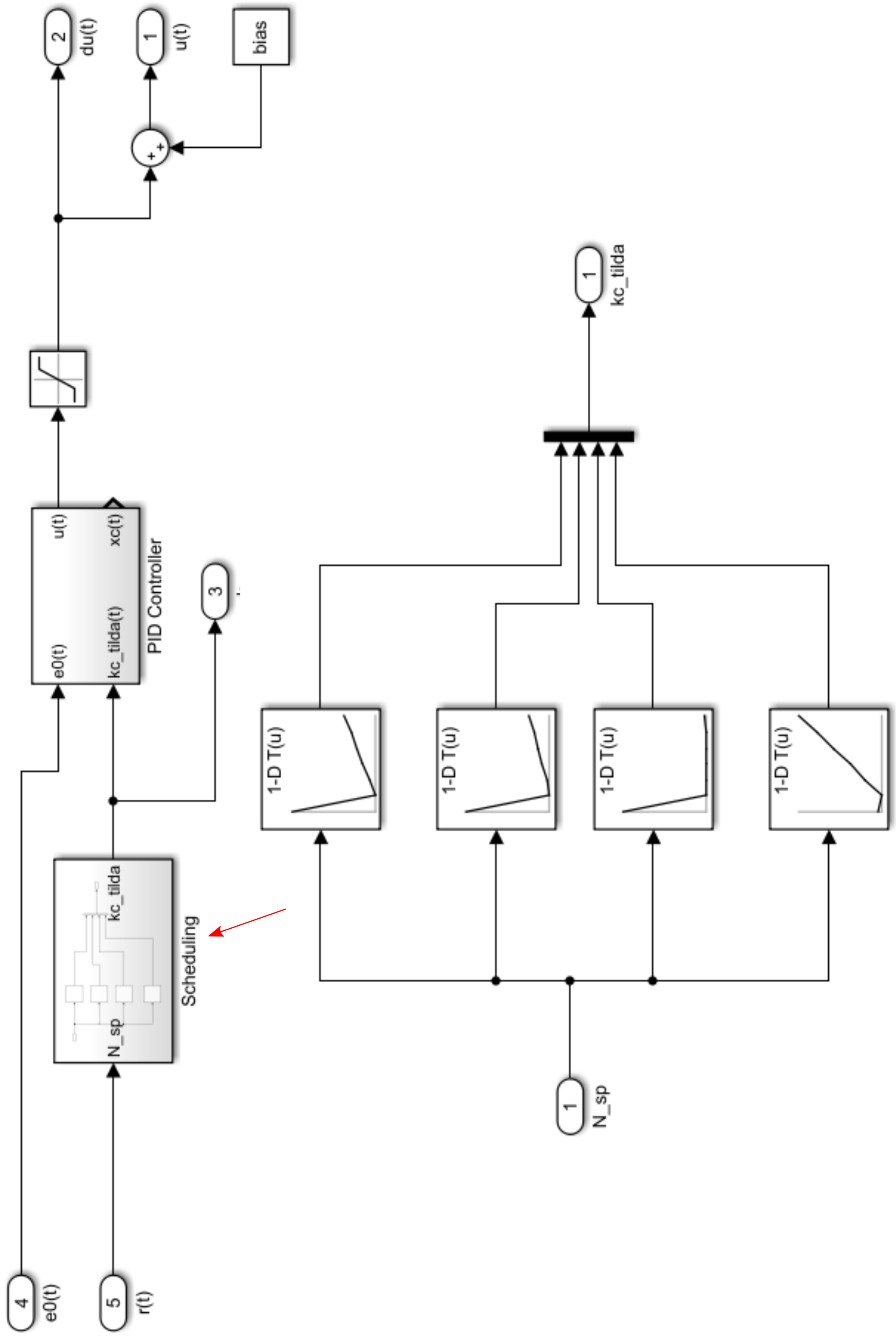


Figure F.4: Scheduled RS controller block