ABSTRACT

Title of Dissertation: SENSING AND CONTROL
UNDER RESOURCE CONSTRAINTS AND UNCERTAINTY:
RISK NEUTRAL AND RISK SENSITIVE APPROACHES

David Hartman
Doctor of Philosophy, 2022

Dissertation Directed by: Professor John S. Baras
Department of Electrical & Computer Engineering

In network estimation and control systems like sensor networks or industrial robotic systems, there are often restrictions or uncertainties that must be taken into account. For example, there are often bandwidth and communication constraints on the estimators or controllers. Additionally, the dynamics model is not always known. Lastly, noise or exogenous disturbances can adversely affect your system.

This thesis addresses three problems in sensing and control in both the H_2 and risk-sensitive control setting. The first problem stems from restrictions on the communications and battery life of sensors. Because of these restrictions, when estimating a state in a system we must cleverly schedule which sensors can be active. The second problem also stems from communication restrictions. In this setting, the sensors and actuators can only communicate with a small number of neighboring sensors. Therefore, we must solve a distributed control problem. The third problem stems from the dynamics of a system being unknown. In this regard, we must solve a control problem using simulated data instead of a fixed model. The research in this thesis, utilizes tools from optimization, estimation, control, and dynamic programming.

SENSING AND CONTROL UNDER RESOURCE CONSTRAINTS AND UNCERTAINTY: RISK NEUTRAL AND RISK SENSITIVE APPROACHES

by

David Hartman

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2022

Advisory Committee:
Professor John S. Baras, Chair/Advisor
Professor Eyad H. Abed
Professor Steven I. Marcus
Professor Andre L. Tits
Professor Nikhil Chopra

Acknowledgments

I owe my gratitude to all the people who have made this thesis possible and because of whom my graduate experience has been one that I will cherish forever.

First and foremost I would like to thank my advisor, Professor John Baras for giving me the valuable opportunity to be a part of his research group. I have learnt many lessons under his guidance. Most importantly, he has taught me the valuable lesson of research and of life that you cannot approach a problem with a fixed idea of what the solution should look like. The problem must inform the solution and not the other way around. It has been a pleasure to learn from a such a tireless individual. It would be an understatement to say this thesis would not be possible without his immense efforts. I treasure my time under his guidance.

I would also like to take the opportunity to acknowledge the support of the following sources: Office of Naval Research (ONR) grant N00014-17-1-2622. Northrup Grumman Seed Grant, and Electrical and Computer Engineering Teaching Assistant at the University of Maryland.

I am also particularly thankful to Mrs. Edwards, whose door was always open to answer my pestering questions. She is really the behind the scenes architect who without her the whole research group would not function. I owe her an immense debt of gratitude for her stamina, patience, and immeasurable help. I would also like to thank the amazing staff at the ECE/ISR graduate office, Emily Irwin, Melanie Prange, Alexis Jenkins, and Maria Hoo. Additionally, I would like to thank Professor Eyad Abed, Professor Steven Marcus, Professor Andre Tits,

and Professor Nikhil Chopra for agreeing to serve on my thesis committee and for sparing their invaluable time reviewing my thesis. I would also like to thank Professors Makowski, Tits, and Marcus for their insightful discussions in and out of their classes.

I have been fortunate to meet some great colleagues in my research group. I specifically want to thank my senior colleagues: Aneesh Raghavan and Dipankar Maity. They made my time here that much easier. I want to thank all the members of my research group in particular those I interacted with most: Erfaun Noorani, Christos Mavridis, Nilesh Suriyarachchi, Usman Fiaz, Fatemeh Alimardani, and, Marilyn Duong. And of course, I would like to thank my roommate of four years and fellow PhD traveler David Nahmias. Last but not least, I would be nowhere without my family. They have and continue to believe in me throughout the whole process. They are the real heroes of it all. It is impossible to remember all, and I apologize to those I've inadvertently left out.

On a final, final note, I would like to thank my undergraduate advisor Professor Andrew Packard of UC Berkeley who first got me interested in controls. He will be missed.

Table of Contents

Chapter 1: Introduction

1.1 Introduction to the Problems Investigated

In this section of the thesis, we outline the research we undertook to addresses several problems organized in three parts: sensor scheduling in the Kalman filter and robust (H_∞) estimation settings; distributed risk-sensitive control; and risk-sensitive control for systems with unknown dynamics.

Part 1: Sensor Scheduling in Kalman and Robust Filtering

Problems with sensor networks related to applications such as environmental monitoring and target tracking involve limitations on battery life and bandwidth capacity. Hence, we must frugally use the resources at hand to minimize estimation errors. The estimation error criteria can either be the minimum mean square error (MMSE) as in the Kalman filter (i.e. H_2 setting) or min-max error as in the robust (i.e., H_∞) setting. The limitation on sensor resources can be addressed by capping the number of times a sensor is activated. This is the issue of sensor sampling. For example, if we are estimating the state over an interval of 10 time units, we may allow the sensor to be activated for only half of the 10 time units. Additionally, we can cap the number of sensors that are activated at any given time. This is the issue of sensor scheduling. For example, if there are 10 sensors available, we may allow only half of the 10 sensors to be

activated per time unit. The sensor sampling and scheduling problem can be formulated as a mixed-integer convex problem. One major question is, does the structure of the problem lend itself to manageable algorithms to solve the optimization problem. Additionally, can we find suitable upper bounds on the solution and can we bound how suboptimal the solution is? We explore solutions to these questions in chapters $3-5$. Some of these questions have been answered in papers [1], [2], [3].

Part 2: Distributed Control in the Risk-Sensitive or Robust Setting

Part 1 above focuses on centralized estimation. Because of communication restrictions, this communication may not be possible. In this part of the thesis, we investigate distributed risk-sensitive or robust control. Distributed, in this case, means agents in a multi-agent system do not communicate with a central hub but merely with their immediate neighbors. Each agent has a local cost function. The goal is that each agent minimizes their own local cost function such that the total cost function is then minimized. Distributed linear quadratic regulators (LQR) has been studied in [4]. We study the problem for risk-sensitive control in the linear quadratic setting (linear exponential quadratic Gaussian, LEQG). We present the distributed control problem in chapter 6.

Part 3: Risk-sensitive or Robust Control with Unknown Dynamics

It is not always the case in a control problem that the dynamics are known. In the third part of the thesis, we investigate discrete time risk-sensitive or robust control problems when the dynamics are unknown. One proposed solution is to directly solve this risk-sensitive control problem using gradients of the cost function with respect to matrices. When the dynamics are unknown, the

gradients can be approximated via simulation. This has been proposed by Zhang et al. [5]. We extend his approach to the LEQG setting. Additionally, we can solve this problem with a model-based solution where we first estimate the uncertain matrices by least squares and then incorporate uncertainty in the estimator when calculating the risk-sensitive controller. We present this work in chapter 7. Our approaches build off the work by Zhang et al. [5] and the work by Dean et al. [6].

1.2 Contributions of the Thesis

In this section, we give a succinct description of the problems addressed in this paper and the solution.

1. **Problem 1.** Find a sensor scheduling strategy that minimizes the mean square error (or H_∞ norm) subject to a constraint on the usage of the sensor. See section 3.3 for a description of this problem.

   (a) **Solution 1.** Formulated a semidefinite program (SDP) solution to a relaxed version of the problem and applied it to a localization use case. See sections 3.4-3.6 for the solution and use case. Also, applied the SDP solution to a trust-based distributed estimation problem in section 4.6. We also formulated an SDP-based solution for sensor scheduling in the robust estimation setting in section 5.3. Lastly, we extend our sensor scheduling algorithm to sensor sampling in section 3.9.

   Relevant publication:

   Maity, Dipankar, David Hartman, and John S. Baras. "Sensor Scheduling for Linear

Systems: A Covariance Tracking Approach." arXiv preprint arXiv:2110.08924 (2021).

(b) **Solution 2.** Composed sufficient conditions for when a greedy algorithm is theoretically justified to solve Problem 1. "Theoretically justified" here means we can find a bound on the solution provided by the greedy algorithm. See sections 4.4-4.5 for the solution.

Relevant publication:

Hartman, David, and John S. Baras. "Near-Optimal Solution to the Non-Uniform Sampling Problem in Kalman Filtering." 2019 IEEE 58th Conference on Decision and Control (CDC). IEEE, 2019.

(c) **Solution 3.** Found an upper bound on the mean square error. Minimize this upper bound instead of minimizing the mean square error itself. This approach improves on algorithms based on observability indices. See section 4.6 for the solution.

2. **Problem 2.** Find a distributed controller in the risk-sensitive control multi-agent setting. Each agent has a local cost function. The goal is to minimize the total sum of cost functions with the constraint that each controller can only interact with their neighbors. See sections 6.1 for a description.

(a) **Solution.** Formulated a consensus-based controller where the controller for an agent is the average of neighboring controllers. This solution is inspired by consensus-based distributed linear filters ([7]). Consensus here means consensus on the controller grain. A bound is formulated between the cost function from the proposed distributed controller and the cost function from an optimal global controller. Distributed controllers and bounds are formulated for both the linear quadratic Gaussian (LQR) and the

risk-based linear exponential quadratic Gaussian (LEQG) settings. See chapter 6 for details.

Relevant upcoming publication:

Hartman, David, and John S. Baras. "A Distributed LQG and LEQG Controller." (In Preparation)

3. **Problem 3.** Find a solution to the risk-sensitive control problem when the linear dynamics are unknown. See chapter 7 for a description.

   (a) **Solution 1.** Formulated a model-based solution whereby the dynamics are estimated along with an uncertainty set. Using ideas from robust control we devise a controller that incorporates the uncertainty set. The algorithm and the different kinds of uncertainty sets are detailed in section 7.4. We present this algorithm on two examples: one a random example and the other a vehicle example.

   Relevant upcoming publication:

   David Hartman, and John S. Baras. "Linear Exponential Gaussian Control with Unknown Dynamics: A Model-based LMI solution." (In Preparation)

   (b) **Solution 2.** Formulated a variant of the model-free, policy optimization solution. Previously it has been presented for the linear quadratic regulator (LQR) and robust control setting ([5]). Here, we present it for the linear exponential gaussian (LEQG) setting. The gradients in the policy optimization are simulated gradients. Additionally, we showcase a two-point simulated gradient instead of the one-point simulated gradient in ([5]). See section 7.3 for details. Similar to the model-based solution we showcase this algorithm on two examples. Lastly, we compare the performances of the model-

free and model-based solutions.

Chapter 2: Mathematical Background

In Chapter 2, we recall some of the theoretical background that will be needed throughout the thesis and we introduce the foundations for the problems that will be addressed.

2.1 Discrete Time Kalman filter (Recursive)

See [8] for more details on this section and Chapter 3 for this material being referenced. The Kalman filter is a recursive algorithm that uses noisy measurements to estimate the states of a dynamical system. The optimal state estimator is an estimator that minimizes the expected error between the state and the state estimator. This expected estimation error can be calculated apriori before receiving any measurements. There exists discrete time (Kalman filter) and continuous time (Kalman-Bucy filter) versions of the Kalman filter. We present here the equations for the discrete time Kalman filter.

Consider the linear, time-invariant, discrete time system of the form,

$$x_{t+1} = Ax_t + w_t \tag{2.1}$$

for $t = 0 \dots T$, where $x_t \in \mathbb{R}^n$, $x_0 \sim \mathcal{N}(\bar{x}_0, P_0)$ is the initial state, $\{w_t\}_{t=0}^T$ is an i.i.d sequence of Gaussian random variables where $w_t \sim \mathcal{N}(0, W)$, and w_t is independent of x_0 for all t.

The dynamical system is equipped with sensors through the following linear, time-invariant system,

$$y_t = Cx_t + v_t \tag{2.2}$$

for $t = 0, \dots, T$, where $y_t \in \mathbb{R}^m$, $\{v_t\}_{t=0}^T$ is an i.i.d sequence of Gaussian random variable where $v_t \sim \mathcal{N}(0, V)$. Furthermore, for all $t \neq s$: $v_t \perp v_s$, $v_t \perp x_0$, and $v_t \perp w_s$ (independence) for $t, s \in \{0, 1, \dots, T-1\}$.

Let $\hat{x} : \{(t,s) | t, s \in \{0, 1, \dots, T-1\}, s \leq t\} \to \mathbb{R}^n$ be the estimator of the state at time t given measurements $\{y_\tau\}_{\tau=0}^s$. Formally, $\hat{x}(t,s)$ is defined as the conditional expectation,

$$\hat{x}(t,s) := E(x_t | y_0, \dots, y_s) \tag{2.3}$$

Define, $\hat{x}_{t|s} := \hat{x}(t,s)$. Additionally, let $e(t) = x(t) - \hat{x}_{t|t}$ be the estimation error at time t where $e(t) \in \mathbb{R}^n$.

Also, let $P_{t|t}$ be the covariance matrix of the estimation error at time t such that

$$P_{t|t} = E(x(t) - \hat{x}_{t|t})(x(t) - \hat{x}_{t|t})' \tag{2.4}$$

for $t \in \{0, \ldots, T-1\}$ where $P_{t|t} \in \mathbb{R}^{n \times n}$.

Let, $P_{t|s}$ be the conditional covariance of the estimation error at time t given measurements from $t = 0$ to $t = s$ such that

$$P_{t|s} = E(x(t) - \hat{x}_{t|s})(x(t) - \hat{x}_{t|s})' \tag{2.5}$$

where $P_{t|s} \in \mathbb{R}^n$.

From standard filtering theory [8], we have assumed the following definition for the prior mean and covariance, $\hat{x}_0 := \hat{x}_{0|-1}$, $P_0 := P_{0|-1}$.

The estimate updates for the estimated state and covariance matrix are

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + P_{t|t-1}C^T(CP_{t|t-1}C^T + V)^{-1}(y_t - C\hat{x}_{t|t-1}) \tag{2.6}$$

$$P_{t|t} = g_t(P_{t|t-1}) = P_{t|t-1} - P_{t|t-1}C^T(CP_{t|t-1}C^T + V)^{-1}CP_{t|t-1} \tag{2.7}$$

$$\hat{x}_{t+1|t} = A\hat{x}_{t|t} \tag{2.8}$$

$$P_{t+1|t} = h_t(P_{t|t}) = AP_{t|t}A^T + W \tag{2.9}$$

Define the Fisher Information Matrix [9], the inverse of covariance matrices to be $Q_{t|t} : P_{t|t}^{-1}$ and $Q_{t|s} := P_{t|s}^{-1}$. The updates in terms of of $Q_{t|t}$ takes the following form

$$Q_{t|t} = Q_{t|t-1} + C_t^T (V)^{-1} C_t \tag{2.10}$$

$$Q_{t+1|t} = (A Q_{t|t}^{-1} A^T + W)^{-1} \tag{2.11}$$

2.2 Schur Complement

See [10] for more details on this section and Chapters 3, 5, and 7 for this material being used when converting between Algebraic Riccati Inequalities and Linear Matrix Inequalities (LMI).

Consider the symmetric matrix

$$X = \begin{bmatrix} M & Y \\ Y' & L \end{bmatrix} \tag{2.12}$$

The Schur Complement states the condition $X > 0$ is equivalent to $L > 0$, and $M - Y L^{-1} Y' > 0$ if L is invertible. Also, symmetric conditions hold when we replace the $>$ sign with the $<$ sign. See [[11], pg. 11].

(2.13)

2.3 Discrete Time Kalman Filter (Non-Recursive)

See [12] for more details on this section and Chapter 4 for this material being referenced.

For this section, let square brackets such as $[X]$ denote the expectation of the random variable X and let $[X|Y]$ denote the conditional expectation of the random variable X given the random variable Y.

Consider the time-invariant dynamical and measurement equations. Now, define the vector of measurements up until time t, $Y_t = (y_0', y_1', \ldots, y_t')'$ where $Y_t \in \mathbb{R}^{m(t+1)}$. Also, define the vector comprising of initial condition and noise, $r_{t-1} = (x_0', w_0', w_1', \ldots, w_{t-1}')'$ where $r_{t-1} \in \mathbb{R}^{n(t+1)}$.

We can express x_t as a function of r_{t-1},

$$x_t = L_{t-1} r_{t-1} \tag{2.14}$$

where L_{t-1} is the n x $n(t+1)$ matrix such that,

$$L_{t-1} = (A^t, A^{t-1}, \ldots, A, I) \tag{2.15}$$

The vector of measurements, Y_t can be formulated non-recursively as

$$Y_t = \Phi_{t-1} r_{t-1} + V_t \tag{2.16}$$

where the random vector of noise measurements, $V_t = (v_0', v_1', \ldots, v_t')'$ and Φ_{t-1} is a matrix of size $m(t+1) \times n(t+1)$ such that,

$$\Phi_{t-1} = \begin{pmatrix} C & 0 \\ CL_0 & 0 \\ \vdots & \vdots \\ CL_{t-2} & 0 \\ CL_{t-1} & \end{pmatrix}$$

Consider the covariance $(\Omega_{r_{t-1}})$ and conditional covariance $(\Sigma_{r_{t-1}})$ of r_{t-1},

$$\Omega_{r_{t-1}} := (r_{t-1} - [r_{t-1}])(r_{t-1} - [r_{t-1}])' \tag{2.17}$$

$$\Sigma_{r_{t-1}} := [(r_{t-1} - [r_{t-1}|Y_t])(r_{t-1} - [r_{t-1}|Y_t])'] \tag{2.18}$$

Subsequently, as had been defined in equation (2.3) the definition of the conditional estimate of x_t given Y_t is as follows

$$\hat{x}_{t|t} = [x_t|Y_t] \tag{2.19}$$

With definition (2.14)-(2.19), we get an equation for the conditional covariance of r_{t-1} in terms of the covariance of r_{t-1}, $\Omega_{r_{t-1}}$,

$$\Sigma_{r_{t-1}} = \Omega_{r_{t-1}} - \Omega_{r_{t-1}}\Phi_{t-1}'(\Phi_{t-1}\Omega_{r_{t-1}}\Phi_{t-1}' + \Sigma_V)^{-1}\Phi_{t-1}\Omega_{r_{t-1}} \tag{2.20}$$

Subsequently, the equation for the state estimate in terms of r_{t-1} is

$$\hat{x}_{t|t} = L_{t-1}\hat{r}_{t-1}(Y_t) \tag{2.21}$$

Consider equation (2.20) with time subscripts removed so that $\Sigma = \Sigma_{r_{t-1}}$, $\Omega = \Omega_{t-1}$, and $\Phi = \Phi_{t-1}$

$$\Sigma = \Omega - \Omega\Phi'(\Phi\Omega\Phi' + \Sigma_v)^{-1}\Phi\Omega \tag{2.22}$$

Take the conventional covariance of x_t defined in equation (2.21) using the conditional covariance of r_{t-1} defined in equation (2.18). We get the estimation error covariance matrix

$$[(x_t - \hat{x}_{t|t})(x_t - \hat{x}_{t|t})'] = L_{t-1}\Sigma_{r_{t-1}}L'_{t-1} \tag{2.23}$$

Let the mean square error (MSE) at time t be $e_t \in \mathbb{R}^n$ and be equal to the trace of equation (2.23). Using the cyclic property of trace we get the following for the MSE (e_t),

$$\begin{aligned} e_t := & \quad [(x_t - \hat{x}_{t|t})^2] && (2.24)\\ = & \quad tr[(x_t - \hat{x}_{t|t})(x_t - \hat{x}_{t|t})'] && (2.25)\\ = & \quad tr(L_{t-1}\Sigma_{r_{t-1}}L'_{t-1}) && (2.26)\\ = & \quad tr(L'_{t-1}L_{t-1}\Sigma_{r_{t-1}}) && (2.27) \end{aligned}$$

Note, e_0 is by definition the $n \times 1$ vector of zeros. Define the matrix $H_{t-1} := L'_{t-1}L_{t-1}$. Using this definition and equation (2.22) of $\Sigma_{r_{t-1}}$, we get our final result, the MSE (e_t) at time t,

$$e_t = tr(H_{t-1}\Sigma_{r_{t-1}}) \tag{2.28}$$

$$= tr(H_{t-1}(\Omega_{t-1} - \Omega_{t-1}\Phi'_{t-1}(\Phi_{t-1}\Omega_{t-1}\Phi'_{t-1} + \Sigma_V)^{-1}\Phi_{t-1}\Omega_{t-1})) \tag{2.29}$$

We next cite a lemma for a simplified definition of $\Sigma_{r_{t-1}}$, the conditional covariance of r_{t-1}.

Lemma 2.3.1 (Hartman et al. [1])**.** *Assuming $\Sigma_V = \sigma_v^2 I$, the conditional error covariance of r_{t-1} as shown by equation (2.22) can be simplified to*

$$\Sigma_{r_{t-1}} = \sigma^2((\Phi'_{t-1}\Phi_{t-1} + \sigma^2\Omega_{t-1}^{-1})^{-1} \tag{2.30}$$

Proof. For ease of notation, we remove the time subscripts from $\Sigma_{r_{t-1}}$, Φ_{t-1}, and Ω_{t-1} and instead use Σ, Φ, and Ω. From equation (2.22) we have $\Sigma = \Omega - \Omega\Phi'(\Phi\Omega\Phi' + \sigma^2)^{-1}\Phi\Omega$. The Woodbury Matrix Identity states

$$(A + B)^{-1} = A^{-1} - A^{-1}B(B + BA^{-1}B)^{-1}BA^{-1} \tag{2.31}$$

where A, B are conformable matrices.

Apply the Woodbury Matrix Identity to $(\Phi\Omega\Phi' + \sigma^2)^{-1}$ where $A = \Phi\Omega\Phi'$ and $B = \sigma^2 I$ leads to,

$$\begin{gathered}(\Phi\Omega\Phi' + \sigma^2)^{-1} = \Phi^{-1\prime}\Omega^{-1}\Phi^{-1} - \Phi^{-1\prime}\Omega^{-1}\Phi^{-1}\sigma^2 \\ (\sigma^2 I + \sigma^4\Phi^{-1\prime}\Sigma^{-1}\Phi^{-1})^{-1}\sigma^2\Phi^{-1\prime}\Omega^{-1}\Phi^{-1}\end{gathered} \tag{2.32}$$

Therefore, the conditional covariance, Σ equals

$$\begin{gathered}\Sigma = \Omega - \Omega\Phi'(\Phi^{-1\prime}\Omega^{-1}\Phi^{-1} - \Phi^{-1\prime}\Omega^{-1}\Phi^{-1}\sigma^2 \\ (\sigma^2 I + \sigma^4\Phi^{-1\prime}\Omega^{-1}\Phi^{-1})^{-1}\sigma^2\Phi^{-1\prime}\Omega^{-1}\Phi^{-1})\Phi\Omega\end{gathered} \tag{2.33}$$

Distributing $\Omega\Phi'$ and $\Phi\Omega$ into the outer parentheses, we get

$$\Sigma = \Omega - (\Omega - \Phi^{-1}\sigma^2(\sigma^2 I + \sigma^4\Phi^{-1\prime}\Omega^{-1}\Phi^{-1})^{-1}\sigma^2\Phi^{-1\prime} \tag{2.34}$$

$$= \Phi^{-1}\sigma^2(\sigma^2 I + \sigma^4\Phi^{-1\prime}\Omega^{-1}\Phi^{-1})^{-1}\sigma^2\Phi^{-1\prime} \tag{2.35}$$

$$= \Phi^{-1}(I + \sigma^2\Phi^{-1\prime}\Omega^{-1}\Phi^{-1})^{-1}\sigma^2\Phi^{-1\prime} \tag{2.36}$$

Distributing Φ^{-1}, $\Phi^{-1\prime}$ inside the parenthesis, we get

$$\Sigma = \sigma^2(\Phi'\Phi + \sigma^2\Omega^{-1})^{-1} \tag{2.37}$$

□

2.4 Supermodularity

See ([13], [14]) for more details on this section and Chapter 4 for this material being referenced.

First consider set functions, real-valued functions whose domain is a collection of sets. For example, for a finite set V, consider a set function $f : 2^V \to \mathbb{R}$ whose domain is the power set of the finite set V.

Definition 2.4.1 (Intuitive)**.** *The intuitive definition of a submodular function is a set function that has the "diminishing return property." The "diminishing return property" is the property of a set function that says the marginal benefit of adding new elements to the set (the argument of function f) decreases. If the function f is a submodular function, then the function $-f$ is a supermodular function.*

Definition 2.4.2 (Formal)**.** *For finite set V, subsets A and B, and set function f, whose domain is the power set of V, a set function f is supermodular if $\forall A \subseteq B \subseteq V$ and $\forall j \notin B$*

$$f(B + j) - f(B) \geq f(A + j) - f(A) \tag{2.38}$$

where $f(B + j)$ stands for $f(B \cup j)$.

An equivalent definition is

Definition 2.4.3. *For finite set V, subsets A and B, and set function f, whose domain is the*

power set of V, a set function f is supermodular if $\forall A \subseteq V$ and $\forall j \notin A$

$$f(A+i+j) - f(A+i) \geq f(A+j) - f(A) \tag{2.39}$$

where $f(A+j)$ stands for $f(A \cup j)$.

Now, we define the sets P^* and P.

Definition 2.4.4. *Consider the real-valued set function, f and the set P that is a subset of N-element set $\{1, \ldots, N\}$ for any $N \geq 1$. Let P^* be the solution set of the following cardinality-constrained optimization problem*

$$\begin{aligned} &\underset{P \subseteq \{0,1,\ldots,N\}}{\textit{minimize}} && f(P) \\ &\textit{subject to} && |P| \leq p \end{aligned} \tag{2.40}$$

Consider Algorithm 1 which is a suboptimal solution to equation (2.40). Algorithm 1 is referred to as a greedy algorithm for sensor scheduling.

Algorithm 1 Greedy Algorithm for Sensor Scheduling

Input: f, p.

Output: set P

$P \leftarrow \emptyset, i \leftarrow 0$

$i \leq r$ $a_i \leftarrow a' \in \underset{a \notin P}{\operatorname{argmax}}(f(P) - f(P \cup \{a\}))$

$P \leftarrow P \cup \{a_i\}, i \leftarrow i+1$

Let P^{greedy} be the output of Algorithm 1 (the greedy solution) that greedily chooses elements to be added in order to greedily maximize the set function f.

Theorem 2.4.1 (Nemhauser-Wolsey '78 [15])**.** *If the objective function f is monotone, non-decreasing, and supermodular, then the set P^{greedy} returned by the greedy algorithm, Algorithm 1 satisfies*

$$f(P^{greedy}) \leq (1 - \frac{1}{e}) f(P^*) + \frac{1}{e} f(\emptyset) \tag{2.41}$$

2.5 H_∞ Filter

There are several notations of Robust Filtering. Here, we address Robust Filtering in the H_∞ filtering sense.

2.5.1 H_∞ Disturbance Rejection

See [11] for more details on this section and Chapter 5 for the material being referenced.

We present the H_∞ disturbance rejection problem in the H_∞ filtering sense, where we aim to minimize the effect of any deterministic disturbance w_t on the cost function z_t.

Consider the following continuous time state space model,

$$\dot{x_t} = Ax_t + Bw_t, \quad x_0 = 0 \tag{2.42}$$

$$z_t = Cx_t + Dw_t \tag{2.43}$$

Let us use the commonly used notation to represent the state space realization of equations (2.42) and (2.43) via the block matrix G below

$$G = \left(\begin{array}{c|c} A & B \\ \hline C & D \end{array} \right) \tag{2.44}$$

Considering the LTI system described by equations (2.42) and (2.43), the transfer function matrix, $G(s)$ where,

$$G(s) = C(sI - A)^{-1}B + D \tag{2.45}$$

captures the effect of the disturbance, $w(\cdot)$, on the cost function $z(\cdot)$.

In Definition 2.5.1, we define the H_∞ norm on the transfer function matrix, $G(s)$. With a slight abuse of notation, we denote G as the linear operator from $w(\cdot)$ to $z(\cdot)$. In Definition 2.5.2, we define the induced operator norm on the linear operator G. Consult [16] for a further discussion on norms of transfer functions and induced operator norms.

Definition 2.5.1. *The H_∞ norm of the transfer function $G(s)$ is*

$$\|G(s)\|_\infty := sup_{\omega \in \mathbb{R}} \sigma_{max}(G(j\omega)) \tag{2.46}$$

where σ_{max} is the maximum singular value of a matrix.

Definition 2.5.2. *The induced operator 2-norm on the linear operator G,*

$$<< G >>:= sup_{w \neq 0} \frac{\|z\|_2}{\|w\|_2} \tag{2.47}$$

where $\|\cdot\|_2$ denotes the L^2 norm.

Remark 2.5.1. *For signal $z(\cdot)$ and signal $w(\cdot)$ spanning a time horizon of T, the L^2 norms are*

$$\|z\|_2 = \sqrt{\int_0^T \|z_t\|_2^2 dt} \tag{2.48}$$

$$\|w\|_2 = \sqrt{\int_0^T \|w_t\|_2^2 dt} \tag{2.49}$$

And therefore by Definition 2.5.2,

$$<< G >>= sup_{w \neq 0} \frac{\sqrt{\int_0^T \|z_t\|_2^2 dt}}{\sqrt{\int_0^T \|w_t\|_2^2 dt}} \tag{2.50}$$

Proposition 2.5.1. *Let a stable linear system have a transfer matrix $G(s)$, and let G denote the linear map it induces from the L^2 spaces of its inputs to its outputs. Its induced linear operator norm $<< G >>$ satisfies*

$$<< G >>= \|G(s)\|_\infty \tag{2.51}$$

Proof. See reference [16] chapter 1, pg. 13. □

Subsequently, $\|G(s)\|_\infty = sup_{w \neq 0} \frac{\|z\|_2}{\|w\|_2}$

Remark 2.5.2. *The γ threshold bound on $\|G(s)\|_\infty$, i.e. $\|G\|_\infty \leq \gamma$ is equivalent to*

$$\forall w \in L^2, \quad \|z\|_2 \leq \gamma \|w\|_2 \tag{2.52}$$

which is subsequently equivalent to

$$sup_{w \in L^2}(\|z\|_2^2 - \gamma^2 \|w\|_2^2) \leq 0 \tag{2.53}$$

The next lemma, Lemma 2.5.1 is a famous lemma in robust control theory that connects the frequency domain models to conditions on an LMI for thresholding the H_∞ norm of the transfer function $G(s)$. The lemma provides a linear matrix inequality condition that must be met for disturbance rejection.

Assumptions:

We first assume the following for section 2.5.1 :

1. (A,B) is stabilizable. (This is a weaker condition than (A,B) controllable, which is also often used.)

2. (A,C) detectable. (This is a weaker condition than (A,C) observable, which is also often used.)

Lemma 2.5.1 (Bounded-real lemma) [17])**.** *Assume that in the system described by equations (2.42) and (2.43), A is (Hurwitz) stable. Then the following are equivalent*

- $\|G\|_\infty \leq \gamma$

- $\exists P > 0; \begin{bmatrix} PA + A'P & PB & C' \\ B'P & -\gamma^2 I & D' \\ C & D & -I \end{bmatrix} < 0$

Note, the second condition can be written as $\exists P > 0$ such that $\begin{bmatrix} A'P + PA + C'C & PB + C'D \\ B'P + D'C & D'D - \gamma^2 I \end{bmatrix} < 0$. Also, note that if $D = 0$, then the second condition (using the Schur Lemma) can be simplified to a Riccati-like inequality, $A'P + PA + \frac{1}{\gamma^2} PBB'P + C'C < 0$.

2.5.2 H_∞ Estimation

See ([16], [18]) for more details on this section. The problem setup will closely follow ([16]), chapter 7.

Now, consider the H_∞ filter (also known as $minimax$ filter) where there are multi-sensor measurements and we must estimate the state.

First, define the weighted norm of vector $x \in \mathbb{R}^n$ for any positive definite matrix R and for any finite $n \geq 1$, $\|x\|_R := \sqrt{x'Rx}$.

Consider the continuous-time, dynamics model

$$\dot{x_t} = Ax_t + Bw_t, \quad x_0 = 0 \tag{2.54}$$

where $x_t \in \mathbb{R}^n$, $A \in \mathbb{R}^{n\times n}$, $w_t \in \mathbb{R}^m$, and $B \in \mathbb{R}^{n\times m}$. Assume the initial state, x_0 is zero.

Consider the sensor model where the measurements are received from N sensors

$$y_t^i = C^i x_t + v_t^i, \quad i = 1, \ldots, N \tag{2.55}$$

where $y_t^i \in \mathbb{R}^{p_i}, C^i \in \mathbb{R}^{p_i \times n}$, and $v_t^i \in \mathbb{R}^{p_i}$. The disturbances w_t and v_t^i are deterministic, square integrable functions, but unknown. We note here that no stochastic noise appears in the dynamics or measurement equations. Only deterministic disturbances appear in this formulation. This is in contrast to other robust filter formulations such as Kalman filters where stochastic noise appears in the dynamics and measurement equations.

We are interested in obtaining an estimate $\hat{x}_t$ for the system at time = t using measurements up until time t, $y_{[0,t]}$, based on some performance index as specified below (see equation (2.62)).

Notation

Let us introduce the following vertices and matrices,

$$y_t = \begin{bmatrix} y_t^1 \\ \\ y_t^N \end{bmatrix}, \quad C = \begin{bmatrix} C^1 \\ \\ C^N \end{bmatrix}, \quad v_t = \begin{bmatrix} v_t^1 \\ \\ v_t^N \end{bmatrix} \tag{2.56}$$

where $y_t \in \mathbb{R}^{\sum_{i=1}^{N} p_i}$, $C \in \mathbb{R}^{\sum_{i=1}^{N} p_i \times n}$, and $v_t \in \mathbb{R}^{\sum_{i=1}^{N} p_i}$.

Then we can rewrite the sensor model in equation (2.55) as

$$y_t = Cx_t + v_t \tag{2.57}$$

We are interested in obtaining an estimate, $\hat{x}_t$, whose estimate is based on a minimax error (see equation (2.62)) for the system state at time = t, using measurements up to (and including) that time, $y_{[0,t]}$. The error in this estimate is

$$e_t := x_t - \hat{x}_t(y_{[0,t]}) \tag{2.58}$$

which depends on the system and measurement disturbances ($\{w_t, t \geq 0\}$ and $\{v_t, t \geq 0\}$), respectively). The initial state x_0 would normally play a part as well, but here it is assumed zero.

Consider the mapping

$$T_{\hat{x}_t}(w, v) := x_t - \hat{x}_t(y_{[0,t]}) \equiv e_t \tag{2.59}$$

This mapping produces the estimation error at time t.

Consider the following norm of the mapping $T_{\hat{x}} := \{T_{\hat{x}_t}, 0 \leq t \leq T\}$ as

$$\|T_{\hat{x}}(w, v)\| := \sqrt{\int_0^T \|x_t - \hat{x}_t(y_{[0,t]})\|_R^2 dt} \tag{2.60}$$

where $R \geq 0$ is some weighting matrix (see [16] equation (7.8b) on pg. 288).

Also, consider the following norm

$$\|w, v\| := \sqrt{\int_0^T (\|w_t\|_2^2 + \|v_t\|_2^2) dt} \tag{2.61}$$

Then, an H_∞-optimal filter, $\hat{x}^* = \{x_t^*(y_{[0,t]}), t \in [0, T]\}$ is defined by

$$\inf_{\hat{x}} \sup_{w,v} \{\|T_{\hat{x}}(w, v)\| / \|w, v\|\} \tag{2.62}$$

$$= \sup_{w,v} \{T_{\hat{x}^*}(w, v)\| / \|w, v\|\} =: \gamma^* \tag{2.63}$$

where γ^* is the optimum H_∞ performance level for the continuous-time filter.

Assume the infinite-time horizon case and so $T \to \infty$ then equation (2.62) is equivalent to

$$\inf_{\{\hat{x}\}_{t=0}^\infty} \sup_{\{w_t, v_t\}_{t=0}^\infty} \frac{\sqrt{\int_0^\infty (\|x_t - \hat{x}_t\|_R^2) dt}}{\sqrt{\int_0^\infty (\|w_t\|_2^2 + \|v_t\|_2^2) dt}} =: \gamma^* \tag{2.64}$$

For a given $\gamma > \gamma^* > 0$, we look for a minimax estimator $\{\hat{x}_t^*\}_{t=0}^\infty$ such that the following bound

$$\inf_{\{\hat{x}\}_{t=0}^\infty} \sup_{\{w_t, v_t\}_{t=0}^\infty} \frac{\sqrt{\int_0^\infty (\|x_t - \hat{x}_t\|_R^2) dt}}{\sqrt{\int_0^\infty (\|w_t\|_2^2 + \|v_t\|_2^2) dt}} < \gamma \tag{2.65}$$

holds. Note that for simplicity we have dropped in the notation the dependence of $\hat{x}_t$ on $y_{[0,t]}$. Intuitively, $\hat{x}^* = \{\hat{x}_t^*\}_{t=0}^\infty$ is the sequences of variables that minimize the estimation error subject

to a worst-case disturbance. Equation (2.65) is in turn equivalent to

$$\inf_{\{\hat{x}\}_{t=0}^{\infty}} \sup_{\{w_t,v_t\}_{t=0}^{\infty}} \int_0^{\infty} (\|x_t - \hat{x}_t\|_R^2)dt - \gamma^2 \int_0^{\infty} (\|w_t\|_2^2 + \|v_t\|_2^2)dt \leq 0 \tag{2.66}$$

The H_∞ filter solves for the minimax estimator $\{\hat{x}_t^*\}_{t=0}^{\infty}$ by solving the following Linear Quadratic (LQ) zero-sum game

$$\inf_{\{\hat{x}\}_{t=0}^{\infty}} \sup_{\{w_t,v_t\}_{t=0}^{\infty}} \int_0^{\infty} \|x_t - \hat{x}_t\|_R^2 dt - \gamma^2 \int_0^{\infty} (\|w_t\|_2^2 + \|v_t\|_2^2)dt \tag{2.67}$$

subject to

$$\dot{x_t} = Ax_t + Bw_t \tag{2.68}$$

$$y_t = Cx_t + v_t \tag{2.69}$$

Note, the objective in the LQ zero-sum game, (2.67) ensures the threshold condition (2.66) The reason for this is that for a quadratic form under linear constraints, the objective function value will either be zero or positive infinity [16]. Therefore, a solution to the LQ zero-sum game is equivalent to a solution that meets the H_∞ condition (2.66).

Therefore, we consider here the LQ zero-sum game formulation of the H_∞ filter and not the formal definition of the H_∞ filter that uses bounds on transfer functions.

We now present a solution to the zero-sum game of equation (2.67) with constraints (2.68)-(2.69).

Theorem 2.5.1 ([18], [19])**.** *If any of the following conditions hold, then there exists a solution to the LQ zero-sum game (2.67) subject to constraints (2.68)-(2.69),*

1. $\exists P \geq 0 : AP + PA' - P(C'C - \gamma^{-2}R)P + BB' = 0; \quad P(0) = P_0$

2. $\exists Q \geq 0 : QA + A'Q - C'C + \gamma^{-2}R + QBB'Q = 0; \quad Q(0) = Q_0$

3. $\exists P^I > 0 : AP^I + P^IA' - P^I(C'C - \gamma^{-2}R)P^I + BB' < 0$

4. $\exists Q^I > 0 : Q^IA + A'Q^I - C'C + \gamma^{-2}R + Q^IBB'Q^I < 0$

where P, Q are solutions to Riccati equalities and P^I, Q^I are solutions to Riccati inequalites.

Proof. See reference [16] pg. 362-381, reference [18], and reference [19] pg. 87. □

Additionally, the Riccati equation in condition (1) is $AP + PA' + BB' - P\sum_{i=1}^{N} C^{i\prime}C^iP + \gamma^{-2}PRP = 0$.

The update for the state estimate is

$$\dot{\hat{x}}_t = A\hat{x}_t + (PC')(y_t - C\hat{x}_t), \quad x_0 = 0 \tag{2.70}$$

or,

$$\dot{\hat{x}}_t = A\hat{x}_t + P\sum_{i=1}^{N} C^{i\prime}(y_t^i - C^i\hat{x}_t), \quad x_0 = 0 \tag{2.71}$$

which is given in reference [16] equation (7.38b) on pg. 300.

For the interested reader, we note one direction of equivalence in Theorem 2.5.1, namely that condition (2) implies condition (1).

Lemma 2.5.2. *Condition (2) implies condition (1) in Theorem 2.5.1*

Proof. Consider the following form of condition (2). Note, condition (2) assumes the infinite horizon case. Here we set the Riccati equality in condition (2) equal to $\dot{Q}$. As our setting is the infinite horizon case, $\dot{Q} = 0$.

$$-QA - A'Q + \sum_{i=1}^{N} C^{i\prime}C^i - \gamma^{-2}R - QBB'Q = \dot{Q} \tag{2.72}$$

Note, the following equality,

$$\frac{d}{dt}P^{-1} = -P^{-1}\dot{P}P^{-1} \tag{2.73}$$

Substituting $\dot{P^{-1}} = \dot{Q}$ into equality (2.73) and multiplying by P on the left and right we get,

$$-P\dot{Q}P = \dot{P} \tag{2.74}$$

Substituting equation (2.72) for the term $\dot{Q}$ on the left hand side of (2.74) and substituting Q^{-1}

for the term P on the left hand side, we get the following Riccati equality which equals $\dot{P}$ in equation (2.74).

$$-P\dot{Q}P = AP + PA' - \sum_{i=1}^{N} PC^{i\prime}C^{i}P + \gamma^{-2}PRP + BB' = \dot{P}$$

As this is the infinite horizon case, set $\dot{P} = 0$, and get

$$AP + PA' - \sum_{i=1}^{N} PC^{i\prime}C^{i}P + \gamma^{-2}PRP + BB' = 0 \tag{2.75}$$

which is condition (1). □

We note, that an LMI formulation can also be found for the Riccati inequalities in Theorem 2.5.1.

2.6 H_∞ Control

2.6.1 H_∞ Static Full-State Feedback Control

See ([11], [18],[16] and [20]) for more details on this section and Chapter 7 for the material being referenced. See ([11] and [20]) for discussion of of LMIs and ([18] and [16]) for discussion of Riccati equations.

Consider the continuous-time, dynamics model

$$\dot{x_t} = Ax_t + Bu_t + Dw_t, \quad x_0 = 0 \tag{2.76}$$

where $x_t \in \mathbb{R}^n, A \in \mathbb{R}^{n \times n}, u_t \in \mathbb{R}^m, B \in \mathbb{R}^{n \times m}, w_t \in \mathbb{R}^l$, and $D \in \mathbb{R}^{n \times l}$.

Consider the state feedback form of controller u_t,

$$u_t = Kx_t \tag{2.77}$$

where $K \in \mathbb{R}^{m \times n}$.

Also, consider the output function,

$$z_t = C_z x_t + D_{zu} u_t \tag{2.78}$$

where $z_t \in \mathbb{R}^n, C_z \in \mathbb{R}^{n \times n}$, and $D_{zu} \in \mathbb{R}^{n \times m}$.

As in reference [11], consider the following definitions $A_{cl} := A + BK$ and $C_{cl} := C_z + D_{zu}K$,.

The transfer function $w \to z$ as a function of K, T_K can be written as,

$$T_K = C_{cl}(sI - A_{cl})^{-1}D \tag{2.79}$$

The H_∞ static state feedback control (for the fully observable case) is to find a K such that

$$\|T_K\|_\infty < \gamma \tag{2.80}$$

As in reference [18], the H_∞ problem can be formulated as a LQ zero-sum game with state feedback control.

$$\inf_u \sup_w \int_0^\infty (\|z\|_2^2 - \gamma^2 \|w\|_2^2) dt \tag{2.81}$$

Assumptions: (See reference [16] pg. 231)

We assume the following for Section 2.6.1 :

1. (A, D) is stabilizable
2. (A, B) is stabilizable
3. (A, C_z) detectable
4. $R := D'_{zu} D_{zu}$
5. $Q := C'_z C_z$
6. $C'_z D_{zu} = 0$

Derivation of Riccati Equation for H_∞ Static Full State Feedback

Assuming a closed-loop saddle point, a solution to the minimax Problem 2.81 can be obtained via the Hamilton Jacobi Isaacs equation (HJBI) [18]. Because, this is a linear quadratic problem, we assume like in the LQR problem [8], the value function is a quadratic form, $V(t, x) = x(t)' P(t) x(t)$.

The HJBI equation characterizes saddle point equilibria, u^*, w^*. The HJBI equation states the

following about the Hamiltonian H,

$$H(t;x,u^*,w^*)=0 \quad \text{where} \quad H(t;x,u^*,w^*)=\frac{\partial V}{\partial x}(t,x)f(t;x,u,w)+u'Ru-\gamma^2 w'w \tag{2.82}$$

Lemma 2.6.1. *The Hamilton Jacobi Isaac's equation gives the following Riccati equation.*

$$\dot{P}=PA+A'P-P(BR^{-1}B'-\gamma^{-2}DD')P+Q=0 \tag{2.83}$$

$$u^*(t,x)=-R^{-1}B'Px \tag{2.84}$$

$$w^*(t,x)=\gamma^{-2}D'Px \tag{2.85}$$

Proof. Consider, the HJBI equation which lead to equation (2.82).

$$H(t;x,u^*,w^*)=0 \quad \text{where} \quad H(t;x,u^*,w^*)=\frac{\partial V}{\partial x}(t,x)f(t;x,u,w)+u'Ru-\gamma^2 w'w$$

We can compute the following partial derivative,

$$\frac{\partial V}{\partial x}(t,x)f(t;x,u,w)=\dot{x}'Px+x'P\dot{x}+x'\dot{P}x \tag{2.86}$$

$$=(Ax+Bu+Dw)'Px+x'P(Ax+Bu+Dw)+u'Ru-\gamma^2 w'w \tag{2.87}$$

The cost to go is,

$$V(t;x,u,w) \approx x'Px + h\Big(x'Qx + u'Ru - w'w + (Ax+Bu+Dw)'Px + x'P(Ax+Bu+Dw) + x'\dot{P}x \Big) \tag{2.88}$$

Now, take the derivative of the expression (2.88) with respect to u and w.

The derivative with respect to u gives,

$$B'Px + x'PB + 2Ru = 0 \tag{2.89}$$

which leads to $u^* = -R^{-1}B'Px$.

And the derivative with respect to w gives,

$$D'Px + x'PD - 2\gamma^2 w = 0 \tag{2.90}$$

which leads to $w^* = \gamma^{-2}D'Px$. □

We now present, a linear matrix inequality solution (instead of the Riccati equation solution) stemming from the algebraic Riccati inequality solution following [11].

Linear Matrix Inequality Solution [11]

We note here that we solve for the linear matrix inequality solution to the H_∞ static state feedback control problem in a more general setting. Previously, we were concerned with the setting when

the output function did not depend on disturbance w_t, i.e.

$$z_t = C_z x_t + D_{zu} u_t \tag{2.91}$$

as in (2.78). In Lemmas (2.6.2) (2.6.3), we deal with the setting when the output function is modified to be

$$z_t = C_z x_t + D_{zu} u_t + D_{zw} w_t \tag{2.92}$$

Note, the extra term $D_{zw} w_t$. We recover the origianl setting, by setting D_{zw} in the linear matrix inequalities to zero.

Lemma 2.6.2. *A solution to the H_∞ static state feedback control problem exists (namely that there exists a K matrix such that equation $\|T_K\|_\infty < \gamma$ in equation (2.112) holds) if and only if there exists K and P matrices such that*

$$P > 0 \tag{2.93}$$

$$\begin{bmatrix} PA_{cl} + A_{cl}'P & PB_{cl} & C_{cl}' \\ B_{cl}'P & -\gamma I & D_{cl}' \\ C_{cl} & D_{cl} & -\gamma I \end{bmatrix} < 0 \tag{2.94}$$

where,

$$A_{cl} = A + BK \tag{2.95}$$

$$C_{cl} = C_z + D_{zu}K \tag{2.96}$$

$$B_{cl} = D \tag{2.97}$$

$$D_{cl} = D_{zw} \tag{2.98}$$

Proof. The lemma is true by the bounded real lemma which is referenced in [11] and [21] pg. 49. □

We now present a lemma stating an equivalent linear matrix inequality condition for the H_∞ state feedback solution to (2.94) which is presented in [11].

Lemma 2.6.3. *The following linear matrix inequalities are equivalent to linear matrix inequalities* (2.93) *and* (2.94)

$$S > 0 \tag{2.99}$$

$$\begin{bmatrix} AS + BW + SA' + W'B' & D & SC_z' + W'D_{zu}' \\ D' & -\gamma I & D_{zw}' \\ C_zS + D_{zu}W & D_{zw} & -\gamma I \end{bmatrix} < 0 \tag{2.100}$$

where $W = -R^{-1}B'$.

Proof. After a series of transforms including the change of variables $S = P^{-1}, W = KS$ we

show (2.93) and (2.94) are equivalent to (2.99) and (2.100). Consider the change of variables,

$$\begin{bmatrix} P^{-1} & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} PA_{cl} + A'_{cl}P & PB_{cl} & C'_{cl} \\ B'_{cl}P & -\gamma I & D'_{cl} \\ C_{cl} & D_{cl} & -\gamma I \end{bmatrix} \begin{bmatrix} P^{-1} & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} = \begin{bmatrix} A_{cl}P^{-1} + P^{-1}A'_{cl} & B_{cl} & P^{-1}C'_{cl} \\ B'_{cl} & -\gamma I & D'_{cl} \\ C_{cl}P^{-1} & D_{cl} & -\gamma I \end{bmatrix} < 0 \tag{2.101}$$

Using the change of variables $S = P^{-1}, W = KS$,

$$A_{cl}P^{-1} = AP^{-1} + BKP^{-1} = AS + BW \tag{2.102}$$

$$C_{cl}P^{-1} = C_zP^{-1} + D_{zu}KP^{-1} = C_zS + D_{zu}W \tag{2.103}$$

We then get the following LMI,

$$\begin{bmatrix} AS + BW + SA' + W'B' & D & SC'_z + W'D'_{zu} \\ D' & -\gamma I & D'_{zw} \\ C_zS + D_{zu}W & D_{zw} & -\gamma I \end{bmatrix} \tag{2.104}$$

□

Remark 2.6.1 (Optimizing for γ)**.** *We aim to find the smallest γ such that a solution to the linear quadratic zero-sum game is found. To find the optimal H_∞ control, the following optimization problem needs to be solved:*

$$\min_{S,W,\gamma} \gamma \tag{2.105}$$

$$S > 0 \tag{2.106}$$

$$\begin{bmatrix} AS + BW + SA' + W'B' & D & SC_z' + W'D_{zu}' \\ D' & -\gamma I & D_{zw}' \\ C_z S + D_{zu} W & D_{zw} & -\gamma I \end{bmatrix} < 0 \tag{2.107}$$

For the H_∞ static full state feedback control problem, an LMI solution stemming from the algebraic Riccati inequality was provided.

2.7 Policy Optimization for H_∞ Control

See [5] for more details on this section and Chapter 7 for reference of this material.

In this section, we view the H_∞ control problem from the perspective of the linear quadratic (LQ) zero-sum game. Generally, a solution to this zero–sum game is obtained by solving a Generalized Algebraic Riccati Equation (GARE) [16]. In [5] Zhang et al. purposed a policy optimization approach to solve the LQ zero-sum game. Fazel et al. ([22]) showed how to use policy optimization to solve the LQR case with a stabilizing set as a constraint. The reason for the success is that the problem has the property of coercivity. This means the cost has infinite value around the boundary of the constraint set and the iterates remain feasible and strictly separated from the infeasible set as the cost decreases. In other words, the cost of LQR diverges to infinity as the controller K approaches the boundary of the feasible set K. The cost of LQR, therefore,

serves as a barrier function on K. Problems with H_∞-norm constraints, however don't have this property. But there is an implicit regularization property, where the algorithmic iterates preserve the H_∞ robustness constraint as if they are regularized by the algorithm. Reference [5] gives the first results on the implicit regularization property and global convergence of policy optimization methods for robust control.

The solutions to the linear exponential quadratic Gaussian (LEQG) control problem and the zero sum LQ game are equivalent. The solution is also a solution to the H_∞ problem [23]. See also references [24], [25], [26], [27], [28], [29] for this equivalence in other settings such as nonlinear, partially observable and others. In the H_∞ problem, the goal is to find a controller that meets a γ upper bound on the H_∞ norm constraint on the closed loop between disturbance and output functions. The H_∞ problem can be formulated as a LQ zero-sum game.

We summarize below the H_∞ problem and LQ zero-sum game for discrete time.
Consider the discrete-time, dynamics model,

$$x_{t+1} = Ax_t + Bu_t + Dw_t, \quad x_0 = 0 \tag{2.108}$$

where $x_t \in \mathbb{R}^n, A \in \mathbb{R}^{n\times n}, u_t \in \mathbb{R}^m, B \in \mathbb{R}^{n\times m}, w_t \in \mathbb{R}^l$, and $D \in \mathbb{R}^{n\times l}$.

Consider the state feedback form of controller u_t,

$$u_t = Kx_t \tag{2.109}$$

where $K \in \mathbb{R}^{m \times n}$.

Also, consider the output function,

$$z_t = C_z x_t + D_{zu} u_t \tag{2.110}$$

where $z_t \in \mathbb{R}^n, C_z \in \mathbb{R}^{n \times n}$, and $D_{zu} \in \mathbb{R}^{n \times m}$.

Consider the following definitions $A_{cl} := A + BK, C_{cl} := C_z + D_{zu}K$, and $B_{cl} := D$. The transfer function $w \to z$ as a function of K, T_K can be written as,

$$T_K = C_{cl}(zI - A_{cl})^{-1} B_{cl} \tag{2.111}$$

The H_∞ static state feedback control problem (for the fully observable case) is to find a K such that

$$\|T_K\|_\infty < \gamma \tag{2.112}$$

As in reference [18], the H_∞ problem can be formulated as a LQ zero-sum game with state feedback control.

$$\inf_u \sup_w \sum_{t=0}^{\infty} \|z\|_2^2 - \gamma^2 \|w\|_2^2 \tag{2.113}$$

Assumptions: (See reference [16]) pg. 218)

We assume the following for Section 2.7 :

1. (A, D) is stabilizable
2. (A, B) is stabilizable
3. (A, C_z) detectable
4. $R := D'_{zu} D_{zu}$
5. $Q := C'_z C_z$
6. $C'_z D_{zu} = 0$

Theorem 2.7.1. *A solution to the discrete-time, infinite horizon H_∞ control problem (as described by the dynamics model (2.108), the output function (2.110), and the cost function (2.113)) exists if there exists a matrix $P \geq 0$ (where $P \in \mathbb{R}^{n \times n}$) such that the generalized discrete Riccati equation (GARE),*

$$P = Q + A'(\tilde{P} - \tilde{P}B(R + B'\tilde{P}B)^{-1}B'\tilde{P})A \tag{2.114}$$

and the existence condition

$$(I - \gamma^{-2} D'PD) > 0 \tag{2.115}$$

hold, where

$$\tilde{P} = P + PD(\gamma^2 I - D'PD)^{-1}D'P \tag{2.116}$$

And the optimal controller at time t*,* u_t^* *is*

$$u_t^* = K^* x_t \tag{2.117}$$

where

$$K^* = -(R + B'\tilde{P}B)^{-1}B'\tilde{P}A \tag{2.118}$$

Proof. See reference [5], Lemma 2.7. □

Note, the GARE equation in equation (2.114) can be written in terms of matrix K^* in equation (2.118) as

$$P = (A + BK^*)'\tilde{P}(A + BK^*) + Q + K^{*\prime}RK^* \tag{2.119}$$

See reference [30] equations (32) and (33).

Remark 2.7.1. *Another form for the Generalized Algebraic Riccati equation (GARE) in equation*

(2.114) is

$$P = A'(P^{-1} + BR^{-1}B' - \gamma^{-2}DD')^{-1}A + Q \tag{2.120}$$

See reference [18], section 4.2*, equation* (38) *and reference [16], section* 3.4*, equation* (3.52).

The optimal controller at time t*,* u_t^* *and the optimal disturbance at time* t*,* w_t^* *are*

$$u_t^* = K^* x_t \tag{2.121}$$

$$w_t^* = L^* x_t \tag{2.122}$$

where

$$K^* = -R^{-1}B'(P^{-1} + BR^{-1}B' - \gamma^{-2}DD')^{-1}A \tag{2.123}$$

$$L^* = \gamma^{-2}D'(P^{-1} + BR^{-1}B' - \gamma^{-2}DD')^{-1}A \tag{2.124}$$

See reference [18], section 4.2*, equation* (40) *and reference [16], section* 3.4*, equation* (3.51).

Next, we present the work on the gradient of the cost function for the LQ zero-sum game as presented by Zhang et al. [5].

2.7.1 LQ Zero-Sum Game Formulation and Solution

The following discussion is based on reference [5] section 6. Reference [5] presents a solution to the H_∞ discrete time, infinite horizon problem shown in reference [16] in chapter 3, section 3.4. An equivalent solution to the solution in reference [16] is the solution in reference [18], section 4.2.

Consider the discrete time, dynamics model,

$$x_{t+1} = Ax_t + Bu_t + Dw_t \tag{2.125}$$

Define the cost function, $C(u_t, w_t)$ to be the cost in equation (2.113) for the H_∞ control problem. In other words,

$$C(u_t, w_t) := \sum_{t=0}^{\infty} (x_t' Q x_t + u_t' R u_t - \gamma^2 w_t' w_t) \tag{2.126}$$

See reference [5], equation (6.1) in section 6 for a similar formulation. The objective function for the discrete time, H_∞ control problem formulated as a zero-sum LQ game is as follows,

$$\min_u \max_w C(u_t, w_t) \tag{2.127}$$

We consider controller and disturbance functions that are of a linear form (linear with respect to the state)

$$u_t = Kx_t \tag{2.128}$$

$$w_t = Lx_t \tag{2.129}$$

Therefore, we can write $C(u_t, w_t)$ in terms of the matrices K and L and subsequently the cost function can be written as $C(K, L)$ such that

$$C(K, L) := \sum_{t=0}^{\infty} x_t' Q x_t + (Kx_t)' R (Kx_t) - \gamma^2 (Lx_t)'(Lx_t) \tag{2.130}$$

where $C(K, L) = C(u_t, w_t)$. Note, we have imposed the constraint that the controller and disturbance functions are linear. However, it is well-known the optimal controller and disturbance functions are indeed linear functions of the state ([18]).

In fact, the optimal matrices, K^* and L^* (such that $u_t^* = K^* x_t$ and $w_t^* = L^* x_t$) can be written analytically as in equations (2.123) and (2.124). An alternative method as presented in reference [5], section 6.2 is to solve the zero-sum LQ game using policy gradient. For example, the policy gradient updates can be written as,

$$L' = L + \alpha \nabla_L C(K, L) \tag{2.131}$$

$$K' = K - \eta \nabla_K C(K, L) \tag{2.132}$$

where $\nabla_L C(K, L)$ and $\nabla_K C(K, L)$ are the derivatives of the cost function in equation (2.130) with respect to the control parameter K and the disturbance parameter L, respectively where α and η are step sizes.

Now, assume the parameters of a linear dynamical system are unknown (i.e. A, B, D), but a simulator exists for the dynamics.

2.7.2 Model-Free Policy Gradient Solution When Dynamics are Unknown

We present a model-free policy gradient method to solve the zero-sum LQ game formulation of the H_∞ control problem similar to the one presented in [5] and [22]. For the policy gradient for parameters, K and L affiliated with the controller and disturbance, the form of the estimated or simulated gradient is as follows,

$$L' \approx L + \alpha \hat{\nabla}_L C(K, L) \tag{2.133}$$

$$K' \approx K - \eta \hat{\nabla}_K C(K, L) \tag{2.134}$$

where $\hat{\nabla}_K$ and $\hat{\nabla}_L$ are simulated gradients instead of actual gradients. Actual gradients are unknown as the matrices A, B, and D are unknown.

The general process for the policy gradient method for N iterations, is described below starting with initial guesses for matrices, K_0, L_0. This process is detailed in reference [5] (section 6 and appendix D). The problem is to find a suitable simulated gradient for this problem. Possible simulated gradients as detailed in reference ([31]) include finite difference methods, infinitesimal

perturbation analysis (IPA) and likelihood ratio estimators.

1. Simulate the trajectory (episode) for T time steps by using the disturbance $w_t = L_n x_t$, and controller $u_t = K_n x_t$.

2. Solve for $\hat{\nabla}_L C(K_n, L_n)$ using your favorite choice of simulated gradient.

3. Update, $L_{n+1} \approx L_n + \alpha \hat{\nabla}_L C(K, L)$

4. Using the updated, L_{n+1} simulate another T time steps of the trajectory.

5. Solve for $\hat{\nabla}_K C(K_n, L_{n+1})$ using your favorite choice of simulated gradient.

6. Update, $K_{n+1} \approx K_n - \eta \hat{\nabla}_K C(K_n, L_{n+1})$

Part I: Sensor Scheduling in the Kalman Filter Setting

Chapter 3: Sensor Scheduling for Discrete Time Kalman Filter: A Semidefinite Program Approach

In this chapter, we present a mixed-integer semidefinite program (SDP) solution to the sensor scheduling problem in the discrete-time Kalman filter setting. The mixed-integer SDP solution follows the work by Mourikis et al. [32] where a mixed-integer SDP for the steady-state continuous-time Kalman filter is presented. Our work differs in that we deal with a non steady-state, discrete-time Kalman filter [2]. Additionally, we present three relaxation approaches to the mixed-integer SDP. We then showcase the effectiveness of these relaxation approaches on a sensor localization problem. The contribution of this chapter is a formulation of the sensor scheduling problem as a relaxed version of the mixed-integer semidefinite program that can easily be solved with one of the many available SDP solvers such as in [33] and [34].

3.1 Introduction and Literature Review

A whole array of attempts have been proposed to approximately solve the sensor scheduling problem. Meier et al. [35] proposed a solution that checks all possible sensor schedules. Vitus et al. [36] devised a solution that pruned this exponential-size search tree. Joshi et al. [37] relaxed

the problem to a convex optimization problem, a heuristic that often works well in practice. He et al. [38] modeled the sensor scheduling problem as a partially observable Markov decision process (POMDP) and proposed approximate solutions to solve this POMDP. Many greedy solutions have been proposed as well. Chhetri [39] proposed a greedy solution that also employs integer programming. Some of these greedy solutions leverage ideas from submodularity and include optimizing the determinant of the logarithm or the conditional entropy of the covariance matrix as opposed to optimizing the minimum mean square error (MMSE) or equivalently the trace of the covariance matrix. Additionally, Liu et al. [40] proposed an optimal time-periodic sensor schedule. Furthermore, Soleymani et al. [41] devised relaxations that incorporate the cardinality constraints into the objective function. Sensor scheduling has also been studied in continuous-time. Mourikis et al. [32] proposed a mixed-integer semidefinite (SDP) program to solve the steady-state continuous-time sensor scheduling problem.

3.2 Notation and Assumptions

We repeat here some of the notation and assumptions from Chapter 2, Section 2.1.

Consider $x_t \in \mathbb{R}^n$: state of system at time t, $x_0 \sim \mathcal{N}(\bar{x_0}, P_0)$: the initial state, $\{w_t\}_{t=0}^T$: process noise that is an i.i.d sequence of Gaussian random variables where $w_t \sim \mathcal{N}(0, W)$, and w_t is independent of x_0 for all t. We also define the inverse of P_0, $Q_0 := P_0^{-1}$.

Consider $y_t^i \in \mathbb{R}^m$: i^{th} sensor measurement at time t, $\{v_t^i\}_{t=0}^T$: measurement noise for sensor i that is an i.i.d sequence of Gaussian random variables with $v_t^i \sim \mathcal{N}(0, V^i)$. Furthermore, for all

$t \neq s$: $v_t \perp v_s$, $v_t \perp x_0$, and $v_t \perp w_s$ (independence) for $\{0, 1, \ldots, T-1\}$.

Let $\hat{x} : \{(t,s)|t,s \in \{0,1,\ldots,T-1\}, s \leq t\} \rightarrow \mathbb{R}^n$ be the estimator of the state at time t given measurements $\{y_\tau\}_{\tau=0}^{s}$. Formally, $\hat{x}(t,s)$ is defined as the conditional expectation,

$$\hat{x}(t,s) = E(x_t|y_0,\ldots,y_s) \tag{3.1}$$

Define, $\hat{x}_{t|s} := \hat{x}(t,s)$. Additionally, let $e(t) = x(t) - \hat{x}_{t|t}$ be the estimation error at time t. Also, let $P_{t|t} = E(x(t) - \hat{x}_{t|t})(x(t) - \hat{x}_{t|t})'$ be the covariance matrix of the estimation error at time t starting at time = 0 where $P_{t|t} \in \mathbb{R}^{n \times n}$. Let, $P_{t|s} = E(x(t) - \hat{x}_{t|s})(x(t) - \hat{x}_{t|s})'$ be the conditional covariance of the estimation error at time t given measurements from $t = 0$ to $t = s$.

Define $Q_{t|t} := P_{t|t}^{-1}$ and $Q_{t|s} := P_{t|s}^{-1}$.

3.3 Problem Formulation

In this section, we describe a setup of the sensor scheduling problem that has been explored in the literature as summarized in Section 3.1.

Consider the discrete-time, linear, time invariant system

$$x_{t+1} = Ax_t + w_t \quad \forall t \in \{0,\ldots,T\} \tag{3.2}$$

The dynamical system is equipped with N sensors described by

$$y_t^i = C^i x_t + v_t^i, \quad i \in \mathcal{N} = \{1, \ldots, N\}, \tag{3.3}$$

where $w_t \sim \mathcal{N}(0, W)$ and $v_t^i \sim \mathcal{N}(0, V^i)$. Note, since for $t \neq s$: $v_t \perp v_s$, then $V = blkdiag\{V^i\}_{i=1}^N$.

We will assume that only p out of N sensors can be used at any time to obtain measurements in order to estimate the state of a system, where the number p is given.

Let $\theta : \{0, 1, \ldots, T\} \to \mathbb{R}^N$ be a sensor schedule function such that

$$\theta_t^i = \begin{cases} 1 & \text{if sensor } i \text{ is on at time } t, \\ 0 & \text{if sensor } i \text{ is off at time } t, \end{cases} . \tag{3.4}$$

for $i = \{1, \ldots, N\}$ and $\theta_t = (\theta_t^1, \ldots, \theta_t^i, \ldots \theta_t^N)$.

The objective will be to find a sensor schedule θ in order to minimize the cumulative trace of the error covariance, $\sum_{t=0}^T tr(P_{t|t}(\theta(t)))$, or, equivalently, in terms of the inverse of the covariance matrix $\sum_{t=0}^T tr(Q_{t|t}^{-1}(\theta(t)))$ [41]. The sensor scheduling problem can be solved using a brute force method, whereby all N^T schedules are checked. There are some sensor scheduling algorithms that cleverly prune the search space [36].

Note, from standard Kalman filtering theory ([42], [8]), the updates for the inverse of the covariance matrices, $(Q_{t|t} = P_{t|t}^{-1})$, the information filter, as functions of θ_t are,

$$Q_{t|t}(\theta_t) = Q_{t|t-1}(\theta_t) + \sum_{i=1}^{N} \theta_t^i C^{i\prime}(V^i)^{-1} C_t^i \tag{3.5}$$

$$Q_{t+1|t}(\theta_t) = (AQ_{t|t}^{-1}(\theta_t)A' + W)^{-1} \tag{3.6}$$

Consider the set of indices indicating the active sensors at time t, $\mathcal{S}_t$, such that $\mathcal{S}_t := \{i \quad |\theta_t^i = 1\}$. Let $C = vec\{C^i\}_{i=1}^N$ and $V = blkdiag\{V^i\}_{i=1}^N$.

Also, let $C_t(\mathcal{S}_t) = vec\{C^i\}_{i\in\mathcal{S}_t}$ and $V_t(\mathcal{S}_t) = blkdiag\{V^i\}_{i\in\mathcal{S}_t}$ We therefore note, the updates for the covariance matrices, $P_{t|t}$ as functions of θ_t, $C_t(\mathcal{S}_t)$, and $V_t(\mathcal{S}_t)$.

$$P_{t|t}(\theta_t) = g_t(P_{t|t-1}(\theta_t)) = P_{t|t-1}(\theta_t) - P_{t|t-1}(\theta_t)C_t(\mathcal{S}_t)'(C_t(\mathcal{S}_t)P_{t|t-1}(\theta_t)C_t(\mathcal{S}_t)' + V_t(\mathcal{S}_t))^{-1}$$

$$C_t(\mathcal{S}_t)P_{t|t-1}(\theta_t) \tag{3.7}$$

$$P_{t+1|t}(\theta_t) = h_t(P_{t|t}(\theta_t)) = AP_{t|t}(\theta_t)A^T + W \tag{3.8}$$

3.4 Mixed-Integer Semidefinite Program Formulation

In this section, we formulate the sensor scheduling problem as a mixed-integer semidefinite program. In semidefinite programming (SDP), we minimize a linear function subject to the constraint that an affine combination of symmetric matrices is positive semidefinite [43]. In the sensor scheduling problem, the objective function is the mean square error, and the constraints are the update equations from the information filter (equation (3.11)). (Chapter 2, Section 2.1

and reference [9] for more information on information filter) and the restriction on the number of sensors activated per time (equation (3.13)). The mixed-integer SDP formulation for sensor scheduling is presented in Problem 3.4.1.

Note: in this section we use the notation $P_t := P_{t|t}$ and $Q_t := Q_{t|t}$ to simplify subscripts.

Problem 3.4.1. *For any given* $p \leq N, T > 0, P_0(\theta_0) > 0$, *and* $Q_0(\theta_0) := P_0(\theta_0)^{-1}$,

$$\min_{(\theta_t \in \{0,1\}^N, P_t(\theta_t), Q_t(\theta_t))_{t=1}^T} \sum_{t=1}^{T} tr(P_t(\theta_t)) \tag{3.9}$$

$$\textit{subject to } Q_t(\theta_t) = P_t(\theta_t)^{-1} \tag{3.10}$$

$$Q_t(\theta_t) = Q_{t|t-1}(\theta_t) + \sum_{i=1}^{N} \theta_t^i C_t^{iT} (V^i)^{-1} C_t^i \tag{3.11}$$

$$Q_{t+1|t}(\theta_t) = (AQ_t^{-1}(\theta_t)A^T + W)^{-1} \tag{3.12}$$

$$1^T \theta_t = p \tag{3.13}$$

$$t \in \{1, \ldots, T\}.$$

An alternative formulation for Problem 3.4.1 is the mixed-integer SDP formulated in problem 3.4.2.

Problem 3.4.2. *For any given* $p \leq N, T > 0, P_0(\theta_0) > 0$, *and* $Q_0(\theta_0) := P_0(\theta_0)^{-1}$,

$$\min_{(\theta_t \in \{0,1\}^N, P_t, Q_t)_{t=1}^T} \sum_{t=1}^{T} tr(P_t(\theta_t)) \tag{3.14}$$

$$\textit{subject to} \begin{bmatrix} P_t(\theta_t) & I \\ I & Q_t(\theta_t) \end{bmatrix} \geq 0 \tag{3.15}$$

$$Q_t(\theta_t) = Q_{t|t-1}(\theta_t) + \sum_{i=1}^{N} \theta_t^i C_t^{iT} (V^i)^{-1} C_t^i \tag{3.16}$$

$$\begin{bmatrix} W^{-1} - Q_{t+1|t}(\theta_t) & W^{-1}A \\ A'W^{-1} & Q_t(\theta_t) + A'W^{-1}A \end{bmatrix} \geq 0, \tag{3.17}$$

$$1^T \theta_t = p \tag{3.18}$$

$$t \in \{1, \ldots, T\}.$$

Problem 3.4.2 is a mixed-integer semidefinite Program (SDP) as it is an SDP with integral constraints and convex constraints. SDPs are optimization problems with linear cost functions and linear matrix inequalites as constraints (3.15, 3.17). Linear constraints (3.16, 3.18) can also be written as linear matrix inequalities. This sensor scheduling problem can be solved using mixed-integer SDP solvers like YALMIP's BNB or CUTSDP solver. The BNB solver uses branch and bound that has a worse-case complexity of $O(N^T)$. The following proposition shows the connection between the two problems, Problems 3.4.1 and 3.4.2.

Proposition 3.4.1. *The solution for Problem 3.4.1 and the solution for Problem 3.4.2 are equivalent and therefore the sensor scheduling problem can be formulated as a mixed-integer SDP.*

Proof. Converting inequalities to equalities and using Schur complement ([44] and Section 2.2)

to convert equation (3.7) and (3.10) to equation (3.15) and equation (3.12) to equation (3.17), we get the mixed integer SDP in Problem 3.4.2 [32]. The Woodbury matrix identity [10] is used in formulating the linear matrix inequality in equation (3.17). □

We now propose three relaxation approaches to the mixed integer SDP. The three relaxation approaches relax the integral constraint on the variable θ (equation (3.18)). The first and third approach convert the integral constraint to an interval constraint. The second approach uses a technique called lifting and relaxation to relax the integral constraint.

3.5 Semidefinite Program Relaxations

In this section, we present three relaxation approaches to the mixed-integer SDP formulated above. The three relaxation approaches are as follows:

1. **Relaxation Approach 1**

   We can relax the mixed integer SDP by relaxing the integral constraint $\theta_t \in \{0,1\}^N$ to be of the form $0 \leq \theta_t^i \leq 1$ for each sensor i at time t and subsequently we are left with an SDP. Of the outputted θ's, the p largest ones are chosen and the rest of the θ's are set to zero.

2. **Relaxation Approach 2**

   The integral constraint in Problem 3.4.2, $\theta_t \in \{0,1\}^N$ can equivalently be written in

another form by introducting the variable Θ_t as in [45],

$$\iff \Theta_t = \theta_t \theta_t', \quad \Theta_t^{ii} = \theta_t^i \tag{3.19}$$

$$\iff \Theta_t - \theta_t \theta_t' \geq 0, \quad rank(\Theta_t) = 1, \quad \Theta_t^{ii} = \theta_t^i \tag{3.20}$$

The constraint set (3.20) is not a convex set because of the rank constraint. The constraint set (3.20) can be transformed into a convex set by removing the rank constraint. Adding a nuclear norm to the cost function is often a surrogate for the rank constraint as the nuclear norm helps keep the rank small. [46]. Define the nuclear norm for matrix Θ_t,

$$\|\Theta_t\| := trace(\sqrt{\Theta_t' \Theta_t}) \tag{3.21}$$

Therefore, using approach 2, we get the following convex SDP problem with the nuclear norm in the cost function,

Given $p \leq N, T > 0, P_0(\theta_0) > 0$, and $Q_0(\theta_0) := P_0(\theta_0)^{-1}$,

$$\min_{(\{\theta_t\}_{t=1}^T, \Theta_t \in \mathbb{R}^{N\times N}), P_t(\theta_t), Q_t(\theta_t)} \sum_{t=1}^{T} tr(P_t(\theta_t)) + \|\Theta_t\|_* \tag{3.22}$$

$$\text{subject to} \begin{bmatrix} P_t(\theta_t) & I \\ I & Q_t(\theta_t) \end{bmatrix} \geq 0 \tag{3.23}$$

$$Q_t(\theta_t) = Q_{t|t-1}(\theta_t) + \sum_{i=1}^{N} \theta_t^i C_t^{iT} (V_t^i)^{-1} C_t^i \tag{3.24}$$

$$\begin{bmatrix} W^{-1} - Q_{t+1|t}(\theta_t) & W^{-1}A \\ A'W^{-1} & Q_t(\theta_t) + A'W^{-1}A \end{bmatrix} \geq 0, \tag{3.25}$$

$$1^T \theta_t = p \tag{3.26}$$

$$\begin{bmatrix} \Theta_t & \theta_t \\ \theta_t' & 1 \end{bmatrix} \geq 0 \tag{3.27}$$

$$\Theta_t^{ii} = \theta_t^i \tag{3.28}$$

$$t \in \{1, \ldots, T\}.$$

where $\|\cdot\|_*$ is the nuclear norm.

3. **Relaxation Approach 3**

The tracking approach can be viewed as finding the right schedule that optimally tracks the covariance matrix outputted from the relaxed SDP (P_t^o). This is very similar to a trajectory tracking problem, where P_t^o is the reference trajectory. The mixed-integer SDP in Problem 3.4.2 outputs covariance matrices, P_t^o. The aim of the tracking algorithm is to find the

sensor which will lead to an estimation error covariance closest to P_t^o at the corresponding time. The tracking algorithm is shown below (Algorithm 2) and was proposed in [2] by Maity, Hartman, and Baras.

Algorithm 2 Algorithm to compute sensor schedule.

Input $\leftarrow \{P_t^o\}_{t=0}^T$, $P_{0|-1} = P_0$,

for $t = 0 : T$
$M_t(i) \leftarrow g_t(i, P_{t|t-1}), \quad i \in \mathsf{N}$,
$\sigma(t) \leftarrow \text{argmin}_i \, \|P_t^o - M_t(i)\|_F$
$P_t \leftarrow g_t(\sigma(t), P_{t|t-1}), P_{t+1|t} \leftarrow h_t(P_t)$,
end

Output $\leftarrow \sigma$.

The three relaxation approaches to the mixed integer SDP formulation for the sensor scheduling problem are referred to as the max approach, the max-rank approach, and the tracking approach.

3.6 Use Case: Sensor Localization

We test the three relaxation approaches described above on a sensor localization use case. The case is to localize and estimate the position and velocity of a person walking indoors using 6 range-based sensors [47]. Since the prolem is non-linear, a sensor schedule is proposed for the extended Kalman filter [42] instead of the traditional Kalman filter.

Notation. Consider the position and velocity to be the state x where $x = [p_x, p_y, v_x, v_y]^T$.

Dynamics Equation. Consider the dynamics equation to be linear with additive noise such that,

$$x_{k+1} = Ax_k + w_k \tag{3.29}$$

where $A = \begin{bmatrix} I & \triangle t \cdot I \\ 0 & I \end{bmatrix}$ and the process noise w_k is Gaussian with covariance matrix $Q = \sigma_w^2 I$.

Measurement Equations. Lastly, consider the nonlinear sensor equation,

$$y_k^i = h^i(x_k) + v_k^i \tag{3.30}$$

where $h(x_k) = [d_1, d_2, d_3, d_4, d_5, d_6]$ is the output of range between the person and the 6 sensors, ie. $d_i = \sqrt{(p_{xi} - p_x)^2 + (p_{yi} - p_y)^2}$. p_{xi} and p_{yi} are the x, y positions respectively of sensor i. Also, the noise vector v_k is Gaussian with covariance matrix diag($\sigma_{v1}^2, \sigma_{v2}^2, \sigma_{v3}^2, \sigma_{v4}^2, \sigma_{v5}^2, \sigma_{v6}^2$).

As h is a nonlinear function, an extended Kalman filter (EKF) [42] is used in place of the traditional Kalman filter.

The i^{th} row of the Jacobian $H_k = \frac{dh(x_k)}{dx}$ is denoted as

$$[H_k]_i = \begin{bmatrix} \frac{p_x - p_{xi}}{\sqrt{(p_{xi}-p_x)^2+(p_{yi}-p_y)^2}}, & \frac{p_y - p_{yi}}{\sqrt{(p_{xi}-p_x)^2+(p_{yi}-p_y)^2}}, & 0, & 0 \end{bmatrix} \tag{3.31}$$

.

Adaptation of Relaxed Approaches to EKF problem: There are challenges to applying the

three relaxed approaches to an EKF problem. Since, the Jacobian must be recalculated about a new point at every time step, the relaxed SDP must be recalculated for each new time step. The approach is similar to the model predictive control (MPC) approach. In that case, a solution is solved for the remainder of the time span, but only the solution for the next time step is applied. Subsequently, the problem is resolved one step forward in time. The model predictive algorithm is described below in Algorithm 3.

Algorithm 3 Model predictive algorithm to compute sensor schedule

Input $\leftarrow P_0$,

for $t = 0 : T$
$\{\sigma\}_t^T \leftarrow$ RelaxationAlgorithm(P_0),
$\sigma(t) \leftarrow \{\sigma\}_t^t$
$\mathrm{P}_0 \leftarrow h_t(g_t(\sigma(t), P_0))$
end

Output $\leftarrow \{\sigma\}_0^T$.

Figure 3.1 shows the error for the three approaches for a time horizon $T = 30$. The black line indicates the true dynamics and the colored line indicates one of the three approaches. For the sensor localization example for time horizon $T = 30$, the three sensor scheduling approaches produce trajectories that closely follow the true trajectory. The error between each of the approaches and the true trajectory are similar (≈ 0.16).

Also, Table 3.1 presents the mean square errors between the true trajectory and the estimated trajectories coming from one of the three relaxed approaches. The errors are averaged over 30 runs. The errors are fairly close to each other. All three relaxation approaches lead to trajectories that closely track the true trajectory. The mean square error between the true trajectory and the

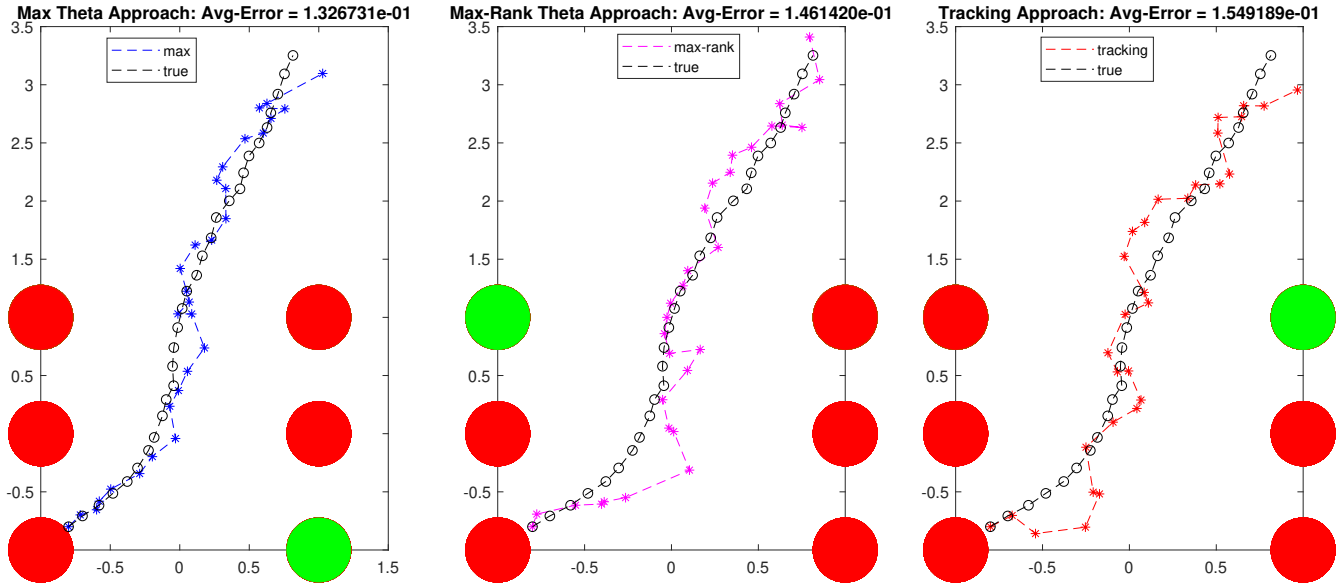

Figure 3.1: Error of 3 different approaches

Max	Max-Rank	Tracking	All Sensors
0.1632	0.1625	0.1680	0.1103

Table 3.1: Error for 3 Different Relaxation Approaches Averaged Over 30 Runs

estimated trajectories coming from an extended Kalman filter that uses all available sensors is 0.1103.

In addition, in the next section we test the tracking approach in a number of numerical evaluations.

3.7 Use Case: Randomly-generated Example

We empirically evaluated the performance of our SDP-based algorithm (that uses the third relaxation approach: tracking) by applying it on a wide range of (randomly generated) scenarios and comparing it to a greedy algorithm [48] and a random search algorithm. The random search algorithm randomly generates 2000 schedules and reports the *best* among these schedules.

We conducted several experiments by varying the dimensions of the system. Four sets of

experiments were conducted for seven different dimensions of $A \in \mathbb{R}^{n \times n}$ for $n = 4, 6, 8$, and 10. For each dimension n, we generated thirty scenarios by randomly generating thirty A matrices with eigenvalues in the range $[1,\ 1.5]$. For each scenario, we considered four randomly chosen sensors with different numbers of dimensions (i.e., different number of rows for matrices C_i). Noise w_t is chosen to have zero mean and unit variance, i.e., $w_t \sim \mathcal{N}(0, I)$. The measurement noises are $v_t^i \sim \mathcal{N}(0, V^i)$ where V^i is a diagonal covariance matrix whose diagonal elements were chosen randomly between 0 and 1. We considered a time horizon of $T = 100$. In Figure 3.2 (left), we illustrate the performance of our algorithm compared to the others. The x-axis represents the dimension of the system and the y-axis represents the cost averaged over the randomly generated scenarios. As can be seen, our algorithm performs better than the other algorithms in terms of the cost $\sum_{t=0}^{T}(P_t)$. We also compare the run-time of these algorithms in Figure 3.2 (right). While the run-times for random search are slightly smaller than ours, their performances are significantly worse than ours. On the other hand, while the performance of [48] is comparable to ours, their run-time is significantly higher compared to ours. In this way the proposed method provides a balanced trade-off between the computation-time and the objective value.

3.7.1 Near-Optimal Performance

While in the last set of experiments we illustrated that our algorithm outperforms the other algorithms, in this section we quantitatively demonstrate how close to the true optimum our solution can be. To compute the optimal schedule, we need to resort to exhaustive searches, and hence, we reduced the time horizon to $T = 9$ in order to maintain tractability of the exhaustive search methods. We randomly picked one of the thirty 10-dimensional scenarios that were used

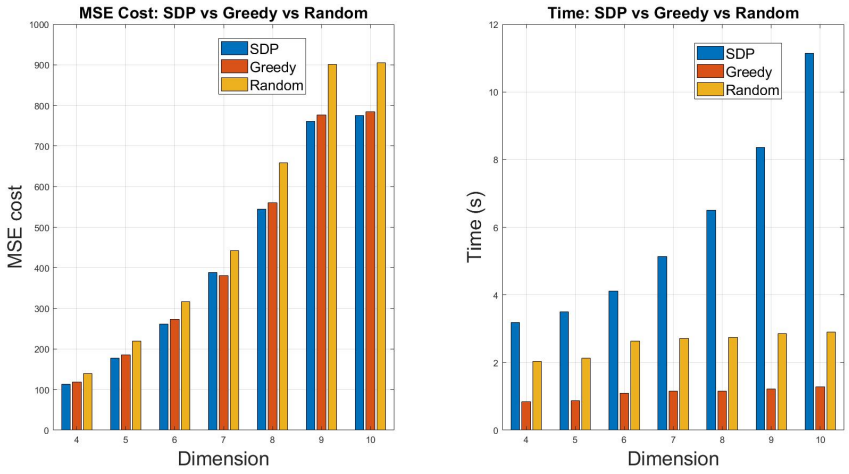

Figure 3.2: Plot of the cost $\sum_{t=0}^{T}(P_t(\sigma))$ and computation times for our SDP algorithm (tracking approach), a greedy algorithm [48] and a random schedule algorithm.

in the previous experiment. We have 4 available sensors, and hence, the total number of possible schedules are 4^{10}. We compare the performances of all permutations of schedules (4^{10} of them) to the performance resulting from our algorithm by plotting a histogram in Figure 3.3. In Figure 3.3, the x-axis represents the cost ($\sum_{t=0}^{T}(P_t)$) and the y-axis represents how many schedules (out of the 4^{10} possible ones) can achieve that cost[1]. Such a histogram represents how likely it is to find a random schedule that will produce a given cost. The histogram is indicated by the red graph and shows the cost for all the different schedule permutations. Superimposed (not part of the histogram) on the red graph, is a gray line that indicates the cost resulting from our tracking algorithm. The cost from our tracking algorithm is smaller than almost all the schedules resulting from the different schedule permutations (red graph) indicating the benefit of our algorithm.

[1]To be precise, we divided the x-axis into intervals of 0.5 (e.g., $[90, 90.5]$ etc.), and evaluated how many schedules produce a cost within each interval.

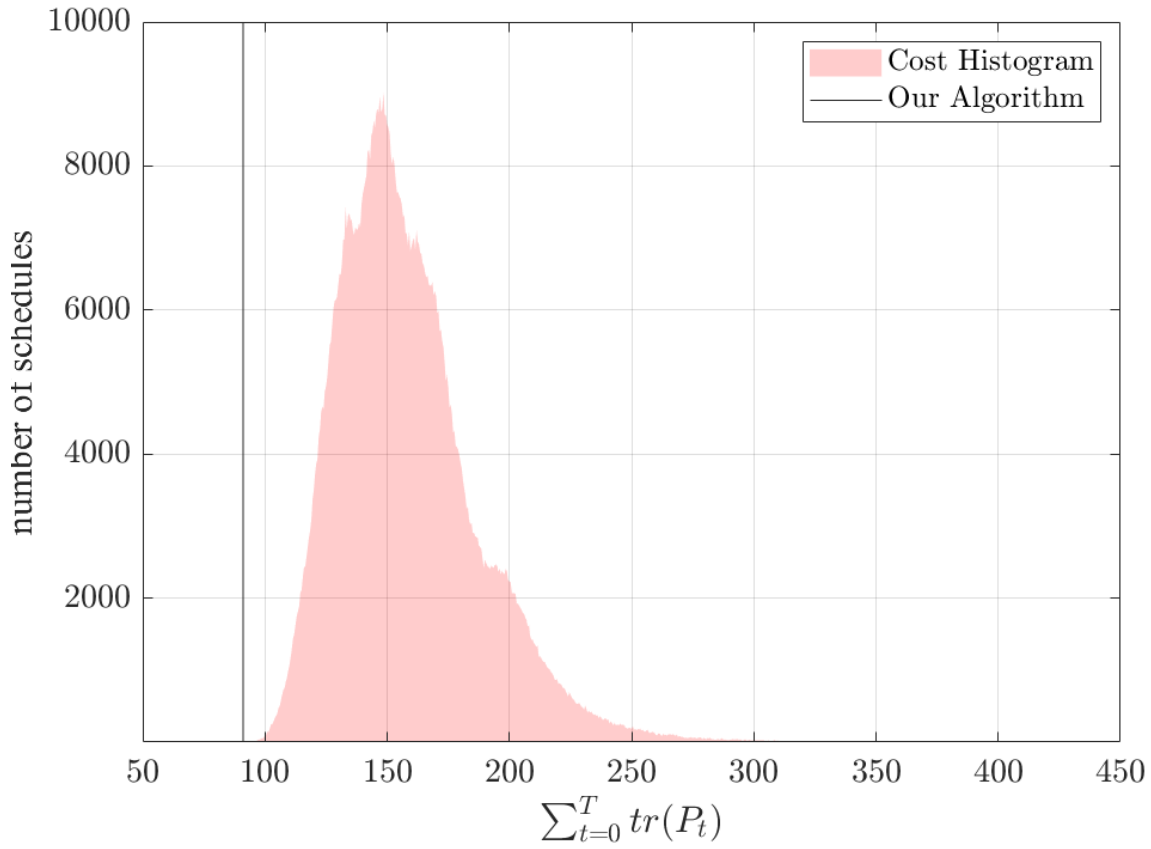

Figure 3.3: The plot in this graph is a histogram of costs for all schedule permutations (red graph). The gray line is superimposed (not a part of the histogram) on the histogram and indicates the cost of the schedule resulting from our tracking algorithm. One can see the cost arising from our algorithm is smaller than almost all the costs coming from the different schedule permutations (red graph).

3.8 Discussion

In this section, we discuss the merits of our SDP-based algorithm and their relaxation approaches by discussing the results from Sections 3.6 and 3.7. In Section 3.6, we compare the results of the different relaxation approaches to a sensor localization case. As the time horizon is too large, we cannot exhaustively search for the optimal schedule. However, we compare the three relaxation approaches to each other. The costs from the schedules arising from the three relaxation approaches are close in value to each other. When all 4 sensors are used instead of a single sensor at each time, there is an improvement of around 30 percent which is not a large improvement considering only a quarter (1 out of 4) of available sensors are used. This is one indication of merit for the SDP-based solutions.

In Section 3.7, we compare the results of the tracking approach to randomly generated examples. Again the time horizon is too large to compare the relaxation approach to the optimal solution. However, we compare our tracking algorithm to a previously used sensor scheduling algorithm [48] as well as a random schedule algorithm. We see that the SDP-based solution (tracking approach) performs better than the greedy algorithm and the random schedule algorithm. This is another indication of merit for the SDP-based tracking algorithm for sensor scheduling.

Lastly, in subsection $3.7.1$, we shrink the time horizon length for the problem in Section 3.7 such that we can compute all the possible schedule permutations. We find that our tracking algorithm results in a schedule that outperforms almost all schedule permutations. This is also an indication of merit for the SDP-based tracking algorithm for sensor scheduling.

3.9 Sensor Sampling

In this section, we show how our algorithm can be extended to sensor sampling, not just sensor scheduling. Our approaches until now are easily adapted to the related sensor sampling problem [49]. In the sensor sampling problem, one selects the number of times the sensor (or sensors) is (are) activated instead of the number of sensors that are activated at a given time (sensor scheduling). At a given time instant, all the sensors are either all active or all not active. We formulate the problem as follows with the relevant information filter equations (3.33) and (3.37 and 3.38), the information filter relation to the covariance matrix (3.32) and (3.36) and the integral constraint (3.34) and (3.39).

Problem 3.9.1. *For the given dynamics and sensor models find a sensor sampling strategy* $\theta = (\theta_1, \ldots, \theta_T)$ *where*

$$\theta_t = \begin{cases} 1 & \textit{if all sensor(s) are active at time } t, \\ & \\ 0 & \textit{if all sensor(s) are not active at time } t, \end{cases}.$$

so that for given $p \leq N, T > 0, P_0(\theta_0) > 0$, *and* $Q_0(\theta_0) := P_0(\theta_0)^{-1}$

$$\min_{(\theta \in \{0,1\}^T, \{P_t(\theta_t)\}_{t=1}^T, \{Q_t(\theta_t)\}_{t=1}^T)} \sum_{t=1}^{T} tr(P_t(\theta_t))$$

$$\textit{subject to } Q_t(\theta_t) = P_t^{-1}(\theta_t) \tag{3.32}$$

$$Q_t(\theta_t) = (AP_{t-1}(\theta_t)A^T + W)^{-1} + \theta_t \sum_{i=1}^{N} C_t^i (V^i)^{-1} C_t^i \tag{3.33}$$

$$t \in \{1, \ldots, T\}$$

$$1^T \theta = p \tag{3.34}$$

An alternative formulation of the sensor sampling problem is as follows:

Problem 3.9.2. *For the given dynamics and sensor models find a sensor sampling strategy* $\theta = (\theta_1, \ldots, \theta_T)$ *so that for given* $p \leq N, T > 0, P_0(\theta_0) > 0$, *and* $Q_0(\theta_0) := P_0(\theta_0)^{-1}$

$$\min_{(\theta \in \{0,1\}^T, \{P_t(\theta_t)\}_{t=1}^T, \{Q_t(\theta_t)\}_{t=1}^T)} \sum_{t=1}^{T} tr(P_t(\theta_t)) \tag{3.35}$$

$$\textit{subject to } \begin{bmatrix} P_t(\theta_t) & I \\ I & Q_t(\theta_t) \end{bmatrix} \geq 0 \tag{3.36}$$

$$Q_t(\theta_t) = Q_{t|t-1}(\theta_t) + \theta_t \sum_{i=1}^{N} C_t^i (V^i)^{-1} C_t^i \tag{3.37}$$

$$\begin{bmatrix} W^{-1} - Q_{t+1|t}(\theta_t) & W^{-1}A \\ A'W^{-1} & Q_t(\theta_t) + A'W^{-1}A \end{bmatrix} \geq 0 \tag{3.38}$$

$$t \in \{1, \ldots, T\}$$

$$1^T \theta = p \tag{3.39}$$

Proposition 3.9.1. *The solutions of Problems 3 and 4 are equivalent.*

Proof. Converting inequalities to equalities and using Schur complement (Section 2.2), we get the mixed-integer SDP. This is the same proof as the equivalence in the sensor scheduling problem provided in Proposition (3.4.1). □

Note: We can use any of the three relaxation approaches provided in Section 3.5. In the example in Section 3.10, we use the first relaxation approach (max approach) to solve Problem 3.9.2. We term a solution to this problem (that uses the first relaxation approach) as our SDP-based solution for sensor sampling.

3.10 Use Case: Randomly-generated Example

We empirically evaluated the performance of our SDP-based solution for sensor sampling (that uses the first relaxation approach in Section 3.5: max approach) by applying it on a wide range of (randomly generated) scenarios and comparing it to a greedy algorithm [48]. We generated thirty scenarios by randomly generating thirty A matrices with eigenvalues in the range $[0.25, 0.85]$. Noise w_t is chosen such that $w_t \sim \mathcal{N}(0, 0.015I)$. The measurement noises are $v_t \sim \mathcal{N}(0, 1.35I)$. We considered two different time horizons, $T = 30$ and $T = 50$. In Figure 3.4, we illustrate the cost of our SDP-based algorithm for sensor sampling (using the first relaxation approach) compared to a greedy algorithm when the time horizon, $T = 30$. In this case, the greedy solution outperforms our SDP-based solution. In Figure 3.5, we do the same but the time horizon is $T = 50$. In this case, our greedy algorithm outperforms our SDP-based solution. In the longer time horizon example ($T = 50$) the greedy algorithm performs worse and our SDP-based solution performs better. The opposite is true in the shorter time horizon example

($T = 30$), although they are fairly close in values. This possibly hints at that as the time horizon becomes larger, our SDP-based solution becomes a better solution than the greedy algorithm. In other words, the two different approaches have their advantages in different situations.

Figure 3.4: Cost in 30 Randomly Generated Examples using a greedy algorithm and our SDP-based solution (max approach) for Time Horizon = 30

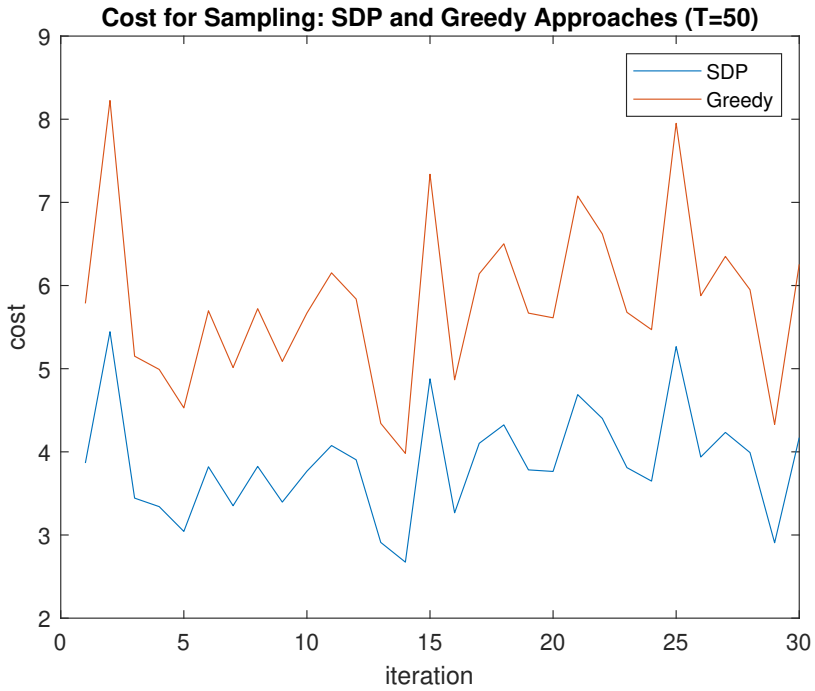

Figure 3.5: Cost in 30 Randomly Generated Examples using a greedy algorithm and our SDP-based solution (max approach) for Time Horizon = 50

3.11 Conclusion

In this chapter, we formulated the sensor scheduling and sensor sampling problem as mixed integer semidefinite programs as in Problems 3.4.2 and 3.9.2. We then relaxed the integral constraints in three different ways leading to a semidefinite program as in Section 3.5. The solution to this semidefinite program serves as the solution to the sensor scheduling problem. We implement our algorithms on a sensor localization case in Section 3.6 and show the viability of all three relaxations. We also show the different costs for different schedule permutations in Section 3.7. Lastly, we show sensor sampling in a randomly-generated example in Section 3.10. Note, one possible future direction is to extend semidefintie program solution to distributed

estimation problems. Another future direction is to extend the tracking Algorithm 2 to a multi-sensor problem.

Chapter 4: Sensor Scheduling for Kalman Filter: A Greedy Approach

We viewed in Chapter 3, multiple approaches to solving the sensor scheduling problem for Kalman filters. The first is brute force and has a complexity of $O(N^T)$ where N is the number of sensors and T is time horizon. The second which uses mixed integer SDP software like YALMIP makes use of branch and bound whose worse complexity can be as high as $O(N^T)$ like the brute force approach. The subsequent approaches all involve some sort of relaxation of a mixed integer SDP to an SDP. SDPs have a polynomial complexity in terms of the number of variables. For example, the complexity of semidefinite programs can be as high as $O(n^{6.5}L)$ [50] for L accurate digits and n variables. The number of variables in the relaxed SDPs is $O(TN^2)$, which gives us a worse-case complexity of $O(T^{6.5}N^{13}L)$. This may present a problem for larger problems. We therefore look for an easier, but related problem in which we can find a lower complexity solution. "Easier" in this case means that we put an additional assumption on the original sensor scheduling problem to make the problem more solvable. The additional assumption is that the covariance of the measurement noise can be written as $\sigma_v^2 I$. This additional assumption allows us to represent the mean square error (MSE) of the Kalman filter at any given time in a non-recursive formulation. The MSE of the Kalman filter at time t is the trace of the covariance at time t. Generally, the covariance matrix at time t is written recursively in terms of the covariance matrix at time $t-1$ (recursive formulation). However, with the additional assumption on the

measurement covariance, the covariance matrix at time t can be written in terms of the covariance matrix at time zero (non-recursive formulation). See Chapter 2, Section 2.3. Additionally, the relaxed SDP approaches shown in the last chapter (Chapter 3) have no guarantees on the gap between the solution of the relaxed problem and the solution of the original problem.

In this chapter, we start by presenting the work by Chamon [51], Tzoumas [48], and others on the formulation of the sensor scheduling problem in terms of the non-recursive Kalman filter. Chamon [51] and Tzoumas [48] leverage the non-recursive formulation to propose a low complexity algorithm for sensor scheduling. They both propose a greedy algorithm as a solution to the sensor scheduling problem. They are justified in their approach by employing a property called supermodularity. The MSE is not generally supermodular. In 2014, Jawaid [52] showed a counter-example when the MSE is not supermodular. Subsequently, Chamon [51] proved the MSE in Kalman filter is approximately supermodular. Tzoumas [48] proved that the metric logdet, (a substitute for MSE) is supermodular. It is this property of supermodularity that allows them to theoretically justify the use of a greedy algorithm [15]. However, both make approximations when dealing with supermodularity, either by replacing MSE with logdet or by showing MSE is approximately supermodular. In our work [1] to be presented in this chapter, we are the first to prove sufficient conditions for when the MSE is supermodular. While these conditions are rather restrictive, we believe the mathematical approach employed in finding these sufficient conditions is useful to others who wish to expand the sufficient conditions criteria. Additionally, we believe the mathematical approach in the proof is useful to others who wish to improve bounds on the approximate supermodularity of the MSE proposed by Chamon [51].

4.1 Literature Review

There is some prior work on supermodularity [53], [54], [55] and sensor scheduling. Jawaid and Smith [52] have shown that logdet of the conditional error covariance of the Kalman filter is not sequence supermodular. Tzoumas et al. have shown that logdet for batch state estimation is supermodular. Additionally, Tzoumas et al. [48], [56] have proven that conditional entropy when the dynamic process is stochastic is supermodular as well. Singh et al. [57] show the mean square estimation error in a single-time step is supermodular under strict conditions of the information and observation matrices. These papers claim that the MSE when scheduling p out of N sensors at each time step is not in general supermodular. However, Chamon, et al. [51] show that the MSE is approximately supermodular. Related to the sensor scheduling problem is the sensor sampling problem. A continuous version of the sensor scheduling problem is discussed in the paper by Baras and Bensoussan [58] in 1989. In 2002, Astrom and Bernhardsson [59] showed that event-driven sampling can outperform periodic sampling with respect to the estimation error. The sampling problem is further discussed in the other papers. Soleymani and Baras [41] solve a relaxed problem with no cardinality constraint but with an augmented cost function that includes estimation error and cost of transmission. In Section 4.4, we take a different approach and show sufficient conditions for when the MSE is supermodular [1].

4.2 Non-Recursive Formulation for Kalman Filter

In this chapter, we repeat the non-recursive formulation of the Kalman filter that appears in Chapter 2, Section 2.3 and is presented in more detail in the Appendix of Bertsekas's Approximate Dynamic Programming and Optimal Control book [12]. The non-recursive formulation requires the added assumption that the covariance of the measurement noise is of the form $\sigma_v^2 I$. This assumption is on top of our original assumption that the measurement noise is independent across sensors (Section 3.2).

First consider the vector $r_{t-1} = (x_0', w_0', w_1', \ldots, w_{t-1}')'$. The non-recursive formulation of the covariance matrix P_t at time t, can be written as a function of two matrices L_{t-1}, which is a vector of powers of the matrix A and $\Sigma_{r_{t-1}}$, the conditional covariance of r_{t-1}. Let,

$$P_t = L_{t-1}\Sigma_{r_{t-1}}L_{t-1}' \tag{4.1}$$

where L_{t-1} is the following n x $n(t+1)$ matrix

$$L_{t-1} = (A^t, A^{t-1}, \ldots, A, I) \tag{4.2}$$

and the conditional covariance

$$\Sigma_{r_{t-1}} = \sigma_v^2((\Phi_{t-1}'\Phi_{t-1} + \sigma_v^2\Omega^{-1})^{-1} \tag{4.3}$$

where Ω_{t-1} is the $n(t+1) \times n(t+1)$ covariance matrix of r_{t-1}

$$\Omega_{t-1} = E[(r_{t-1} - E[r_{t-1}])(r_{t-1} - E[r_{t-1}])'] \tag{4.4}$$

and the $Nm(t+1) \times n(t+1)$ matrix Φ_{t-1} is defined as

$$\Phi_{t-1} = \begin{pmatrix} C_1 & 0 \\ \vdots & \vdots \\ C_N & 0 \\ C_1 L_0 & 0 \\ \vdots & \vdots \\ C_N L_0 & 0 \\ \vdots & \vdots \\ \vdots & \vdots \\ C_1 L_{k-2} & 0 \\ \vdots & \vdots \\ C_N L_{k-2} & 0 \\ C_1 L_{k-1} & \\ \vdots & \\ C_N L_{k-1} & \end{pmatrix}. \tag{4.5}$$

The MSE (e_t) is the trace of P_t

$$\begin{aligned} e_t := & \quad E[(x_t - \hat{x}_{t|t})^2] && (4.6)\\ = & \quad trE[(x_t - \hat{x}_{t|t})(x_t - \hat{x}'_{t|t})] && (4.7)\\ = & \quad tr(L_{t-1}\Sigma_{r_{t-1}}L'_{t-1}) && (4.8)\\ = & \quad tr(L'_{t-1}L_{t-1}\Sigma_{r_{t-1}}) && (4.9) \end{aligned}$$

Note, e_0 is by definition the zero matrix. Define $H_{t-1} := L'_{t-1}L_{t-1}$. We get the MSE ($e_t$) at time t,

$$e_t = \quad tr(H_{t-1}\Sigma_{r_{t-1}}) \tag{4.10}$$

We now turn our attention to the MSE for the Kalman filter example where only a subset of sensors are used by first expanding on the discrete optimization concept of supermodularity.

4.3 Supermodularity in Kalman Filtering

In this section, we discuss the setup for our final analysis of sufficient conditions for supermodularity in Kalman filtering. (Supermodularity is a property of set functions that will be defined in this section; see also Chapter 2, Section 2.4) More precisely, we wish to find conditions under which the mean square error (MSE) is supermodular with respect to the times at which the sensors are activated. In building up to this final analysis, we present both previous

work on supermodularity in Kalman filtering and the criteria that must be met in order to prove the supermodularity property. The usefulness here of supermodularity lies in the fact that it can explain when a greedy algorithm will give theoretically justified good results in combinatorial optimization problems with cardinality constraints. The sensor scheduling problem is one such problem.

In the sensor scheduling problem, the conditional covariance matrix ($\Sigma_{r_{t-1}}$) of r_{t-1}, is a function of the scheduling strategy, θ. We show here a slightly different presentation of the conditional covariance matrix of r_{t-1} as a function of θ than the one shown in [48] and [51]. The presentation follows our work [1]. Let $P(\Theta_t)$ be the set of sensors p out of N for each time instant up until time t and let $S^P(\Theta_t)$ be the selection matrix corresponding to the set of p sensors chosen up until time instant t. The conditional covariance matrix as a function of θ can be written as

$$\Sigma_{r_{t-1}}^{P(\Theta_t)} = \sigma^2((S^P(\Theta_t)\Phi_{t-1})'(S^P(\Theta_t)\Phi_{t-1}) + \sigma^2\Omega_{t-1}^{-1})^{-1} \tag{4.11}$$

such that Θ_t:= $[\theta_0, \ldots, \theta_t]$ and $\theta_t := [\theta_t^1, \ldots, \theta_t^N]^T$ where,

$$\theta_t^i = \begin{cases} 1 & \text{if sensor } i \text{ is on at time } t, \\ 0 & \text{if sensor } i \text{ is off at time } t. \end{cases} . \tag{4.12}$$

The selection matrix $S^P(\Theta_t)$ is composed of $I_{m\times m}$ and $0_{m\times m}$ block matrices. The number of block columns in $S^P(\Theta_t)$ is Nt. The number of block rows in S corresponds to the number of terms in Θ_t that are equal to 1, i.e. $p \times t$. Each block row contains exactly one identity block matrix whose column placement depends on which $\theta_{t'}^i = 1$. The block column of this identity

matrix is in slot $Nt' + i$. For example, consider $N = 2$ sensors, where exactly one sensor is chosen at each time step for a time horizon of length 2 and the corresponding selection set for $t = 1$ is $\Theta_1 = [\theta_0, \theta_1] = \begin{bmatrix} 1 & 0 \\ & \\ 0 & 1 \end{bmatrix}$, which denotes that sensor 1 is chosen at time 1 and sensor 2 is chosen at time 2. Therefore, $S^P(\Theta_2) = \begin{bmatrix} I & 0 & 0 & 0 \\ & & & \\ 0 & 0 & 0 & I \end{bmatrix}$ and $S^P(\Theta_1) = \begin{bmatrix} I & 0 \end{bmatrix}$. If p out of N total sensors are chosen at each time step, then the size of $S(\Theta_t)$ is pmt x Nmt.

Remark 4.3.1. *The discussion in Section 4.4 on sufficient conditions for supermodularity will focus on the less notation heavy problem of sensor sampling instead of sensor scheduling.The big difference between sensor sampling and scheduling is in the definition of Φ_{t-1}. In sensor sampling, Φ_{t-1} is:*

$$\Phi_{t-1} = \begin{pmatrix} C & 0 \\ CL_0 & 0 \\ \vdots & \vdots \\ CL_{t-2} & 0 \\ CL_{t-1} & \end{pmatrix} \tag{4.13}$$

while, in sensor scheduling,

$$\Phi_{t-1} = \begin{pmatrix} C_1 & 0 \\ \vdots & \vdots \\ C_N & 0 \\ C_1 L_0 & 0 \\ \vdots & \vdots \\ C_N L_0 & 0 \\ \vdots & \vdots \\ \vdots & \vdots \\ C_1 L_{k-2} & 0 \\ \vdots & \vdots \\ C_N L_{k-2} & 0 \\ C_1 L_{k-1} & \\ \vdots & \\ C_N L_{k-1} & \end{pmatrix}. \tag{4.14}$$

Definition 4.3.1 (Supermodularity)**.** *The definition of supermodularity is as follows as seen in Section 2.4. For a finite set V, subsets A and B and set function $f : 2^V \longmapsto \mathbb{R}$, supermodularity of set function f can be defined as $\forall A \subseteq V$ and $\forall i, j \in V \backslash A$*

$$f(A + i + j) - f(A + i) \geq f(A + j) - f(A) \tag{4.15}$$

Remark 4.3.2. *For our problem, we let $V = Y_t$ be the set of sampled measurements up until time t, $A = P$, and $f(P) = e_t = tr(H_{t-1}\Sigma^p_{r_{t-1}})$. To prove supermodularity of the MSE (e_t) at time t as a function of the subset of sampled measurements y_t for any $t \in \{1 \dots T\}$ we must show $\forall\ y_j$, $y_l \in Y_t \backslash P$ where $0 \leq j, l \leq t$.*

$$f(P + y_j + y_l) - f(P + y_j) \geq f(P + y_l) - f(P) \tag{4.16}$$

Substituting the definitions for f, V, and A, the proof of supermodularity requires showing

$$\begin{aligned} &tr(H_{t-1}\Sigma^{P+y_j+y_l}_{r_{t-1}}) - tr(H_{t-1}\Sigma^{P+y_j}_{r_{t-1}}) \\ &\geq tr(H_{t-1}\Sigma^{P+y_l}_{r_{t-1}}) - tr(H_{t-1}\Sigma^{P}_{r_{t-1}}) \end{aligned} \tag{4.17}$$

The mean square estimation error at time t (e_t), is supermodular as a function of the sampled measurements or in other words, equation (4.17) holds true if,

$$tr((F + G + H)^{-1}) - tr((F + G)^{-1}) \tag{4.18}$$

$$\geq tr((F + H)^{-1}) - tr((F)^{-1}) \tag{4.19}$$

where $F = H^{-1}_{k-1}\tilde{C}$, $G = H^{-1}_{k-1}\tilde{B}$, $H = H^{-1}_{k-1}\tilde{A}$.

Where the definition of the following matrices $\tilde{C}$, $\tilde{B}$, $\tilde{A}$ is as follows:

$$\tilde{C} := \quad \sigma^{-2}(\Phi'_{t-1}((S^P(\Theta_t))'(S^P(\Theta_t))\Phi_{t-1} + \sigma^2\Omega^{-1}_{t-1}), \tag{4.20}$$

$$\tilde{B} := \qquad \sigma^{-2}(\Phi'_{t-1}(I_{e_j}I'_{e_j})\Phi_{t-1}) \tag{4.21}$$

$$\tilde{A} := \qquad \sigma^{-2}(\Phi'_{t-1}(I_{e_l}I'_{e_l})\Phi_{t-1}) \tag{4.22}$$

4.4 Sufficient Conditions for Supermodularity of MSE

In this section, we develop sufficient conditions for supermodualrity of the mean square error (MSE) in Kalman filtering.

Theorem 4.4.1. *If the following conditions hold*

- *elements of C and A are* ≥ 0
- Ω_{T-1} *is a matrix of the form constant* (σ_w^2) *times identity where the constant is* $1 \leq \sigma_w^2 \leq \frac{1+\sqrt{1+\frac{4}{z}}}{2}$

  $z = max\{x'x|$ *x is column of* $\sigma_v^{2^{-1}}\Phi'_{T-1}\}$
- Σ_v *is a matrix of the form constant* (σ_v^2) *times identity where the constant is* $\sigma_v^2 \geq \frac{\sigma_w^2}{\sigma_w^2-1}\lambda_{max}(\Phi'_{T-1}\Phi_{T-1})$

then the MSE in the Kalman filter as a function of sampled measurements is supermodular.

Proof. To prove supermodularity we must show

$$tr((F+G+H)^{-1}) - tr((F+G)^{-1}) \tag{4.23}$$

$$\geq tr((F+H)^{-1}) - tr((F)^{-1}) \tag{4.24}$$

Writing $\Phi'_{t-1}\Phi_{t-1}$ as $\sum_{i=1}^{t+1} M_i'M_i$ where M_i is the $m(i-1)$ to $m(i)$ rows of Φ_{t-1}, redefine F, G, H

$$F := \sigma^{-2}((S^P(\Theta_t)\Phi_{t-1})'(S^P(\Theta_t)\Phi_{t-1}) + \sigma^2\Omega_{t-1}^{-1}) \tag{4.25}$$

$$G := \sigma^{-2}M_j'M_j = \textstyle\sum_{i=1}^m x_i x_i' \quad \text{where} \quad [x_1,\ldots,x_m] = \sigma^{-1}M_j' \tag{4.26}$$

$$H := \sigma^{-2}M_l'M_l = \textstyle\sum_{i=1}^m y_i y_i' \quad \text{where} \quad [y_1,\ldots,y_m] = \sigma^{-1}M_l' \tag{4.27}$$

By induction, the inequality is true if

$$tr((F+xx')^{-1}) - tr((F+xx'+H)^{-1})$$

$$\leq tr((F)^{-1}) - tr((F+H)^{-1}) \tag{4.28}$$

By the Sherman-Morrison formula ([10], pg. 51) the following holds

$$(F+xx')^{-1} = F^{-1} - \frac{1}{1+ < F^{-1}x, x >} F^{-1}xx'F^{-1} \tag{4.29}$$

$$(F+H+xx')^{-1} = (F+H)^{-1}$$

$$-\frac{1}{1+<(F+H)^{-1}x,x>}(F+H)^{-1}xx'(F+H)^{-1} \tag{4.30}$$

After substituting the Sherman-Morrison formula (4.29, 4.30) into inequality (4.28), inequality (4.28) becomes

$$\frac{x'(F+H)^{-2}x}{1+x'(F+H)^{-1}x} \leq \frac{x'F^{-2}x}{1+x'F^{-1}x} \tag{4.31}$$

Again by induction, inequality (4.31) becomes

$$\frac{x'(F+yy')^{-2}x}{1+x'(F+yy')^{-1}x} \leq \frac{x'F^{-2}x}{1+x'F^{-1}x} \tag{4.32}$$

Using Sherman-Morrison again and rearranging terms, inequality (4.32) becomes

$$\frac{(1+x'F^{-1}x)(x'F^{-1}y)^2(y'F^{-2}y)}{1+y'F^{-1}y}$$

$$\leq 2(x'F^{-2}y)(x'F^{-1}y)(1+x'F^{-1}x)$$

$$-(x'F^{-1}y)^2(x'F^{-2}x) \tag{4.33}$$

Again rearranging we get

$$x'F^{-1}y\{(1+x'F^{-1}x)(y'F^{-2}y)+$$

$$(1+y'F^{-1}y)(x'F^{-2}x)\}$$

$$\leq 2x'F^{-2}y(1+x'F^{-1}x)(1+y'F^{-1}y) \tag{4.34}$$

This is true if

$$x'F^{-1}y(y'F^{-2}y) \leq x'F^{-2}y(1+y'F^{-1}y) \tag{4.35}$$

$$x'F^{-1}y(x'F^{-2}x) \leq x'F^{-2}y(1+x'F^{-1}x) \tag{4.36}$$

The two inequalities are equivalent by exchanging x and y. We are left showing,

$$x'F^{-1}y(x'F^{-2}x) \leq x'F^{-2}y(1+x'F^{-1}x) \tag{4.37}$$

$$\forall \text{x,y columns in } \Phi'_{t-1}$$

F^{-1} is diagonalizable to $P'AP$ and its eigenvalues are $a_{max} = a_1 \geq a_2 \geq \cdots \geq a_{min}$. The above

inequality (4.37) is true if for all x, y columns in Φ'_{t-1} three conditions hold:

$$x'F^{-1}y \leq x'F^{-2}y \tag{4.38}$$

$$(x'F^{-2}x) \leq (1 + x'F^{-1}x) \tag{4.39}$$

$$x'y \geq 0 \tag{4.40}$$

The inequality $x'F^{-1}y \leq x'F^{-2}y$ is true for all x, y if $a_{min} \geq 1$

The inequality $(x'F^{-2}x) \leq (1 + x'F^{-1}x)$ for all x, y is equivalent to

$$z'A^2z \leq z'(A + \tfrac{1}{z'z})z \quad \forall z = Px \tag{4.41}$$

The above inequality (4.41) is true if

$$a_{max}^2 - a_{max} \leq \frac{1}{z'z} \quad \forall z \tag{4.42}$$

or subsequently if,

$$a_{max}^2 - a_{max} \leq \tfrac{1}{max\{x'x|\text{ x is column of } \sigma^{-1}\Phi'\}} = \tfrac{1}{MAX} \tag{4.43}$$

$$\text{or} \quad a_{max} \leq \tfrac{1+\sqrt{1+\frac{4}{MAX}}}{2}$$

In other words,

$$1 \leq a_{min} \leq \cdots \leq a_{max} \leq \frac{1 + \sqrt{1 + \frac{4}{MAX}}}{2} \tag{4.44}$$

The eigenvalues of F are $\lambda_{max}(F) \geq \cdots \geq \lambda_{min}(F)$. The range on the eigenvalues of F is therefore

$$\frac{2}{1+\sqrt{1+\frac{4}{MAX}}} \leq \lambda_{min}(F) \leq \cdots \leq \lambda_{max}(F) \leq 1 \tag{4.45}$$

Note,

$$\lambda_k(F) \leq \sigma^{-2}\lambda_k(\Phi'_{t-1}\Phi_{t-1} + \sigma^2\Omega^{-1}_{t-1}I) \tag{4.46}$$

Assume Ω^{-1}_{t-1} is a diagonal matrix of the form $\frac{1}{\sigma_w^2}I$ then we have

$$\lambda_k(F) \leq \sigma^{-2}\lambda_k(\Phi'_{t-1}\Phi_{t-1}) + \frac{1}{\sigma_w^2} \tag{4.47}$$

If we assume $\frac{1}{\sigma_w^2} \geq \frac{2}{1+\sqrt{1+\frac{4}{MAX}}}$ or $\sigma_w^2 \leq \frac{1+\sqrt{1+\frac{4}{MAX}}}{2}$ then the eigenvalue constraint becomes

$$\lambda_{max}(\Phi'_{t-1}\Phi_{t-1}) \leq \sigma^2(1 - \frac{1}{\sigma_w^2}) \tag{4.48}$$

if $\sigma_w^2 \geq 1$. The last condition that $x'y \geq 0 \quad \forall\, x, y$ columns of Φ'_{t-1} is true if all the elements of matrices C and A are positive. (Note, the requirement of the elements of C and A being positive is a very conservative sufficient condition for $x'y \geq 0$. Other less restrictive conditions could be found.)

In summary, our problem is supermodular if the following sufficient conditions hold:

- elements of C and A are ≥ 0
- $1 \leq \sigma_w^2 \leq \frac{1+\sqrt{1+\frac{4}{MAX}}}{2}$

$MAX = max\{x'x|$ x is column of $\sigma_v^{2^{-1}}\Phi'_{t-1}\}$

- $\sigma_v^2 \geq \frac{\sigma_w^2}{\sigma_w^2-1}\lambda_{max}(\Phi'_{t-1}\Phi_{t-1})$

□

Remark 4.4.1. *The conditions in Theorem 4.4.1 are restrictive. However, as detailed in [51] there is a concept of approximate supermodularity that encompasses problems that are not supermodular but "close" to supermodular (for understanding what "close" means, see [51]). We believe the conditions shown here will help better understand not just the restrictive conditions under which a problem is supermodular but when a problem is "close" (or approximately) supermodular. Like in supermodular problems, in approximately supermodular problems there exists a greedy algorithm that has provable (though not as tight as supermodular) bounds [51].*

4.5 Greedy Solution

We have shown sufficient conditions for supermodularity of the sensor scheduling problem. In this section, we present work by Tzoumas et al. [48] and Chamon et al. [51] who prove bounds on a given greedy algorithm for the sensor scheduling problem leveraging the 1978 theorem by Nemhauser and Wolsey [15]. We then present results of using the greedy algorithm on the sensor localization use case from last chapter (Chapter 3).

Consider, the optimization problem of minimizing f which is a function of the set P, which can have at most p elements in it.

$$\begin{aligned} &\underset{P}{\text{minimize}} \quad f \\ &\text{subject to} \quad |P| \leq p \end{aligned} \tag{4.49}$$

Theorem 4.5.1 (Nemhauser Wolsey '78 [15])**.** *Considering Problem 4.49, if the objective function f is monotone non-increasing, supermodular, and satisfies $f(\emptyset) = 0$, then the set P returned by a greedy algorithm satisfies*

$$f(P) \geq (1 - 1/e) f(P^*) \tag{4.50}$$

Proof. See Nemhauser, Wolsey 1978 paper [15]. □

4.5.1 Optimization Formulation

The sensor scheduling problem for the non-recursive formulation is formulated as the following optimization problem as shown in [48] and [51].

$$\begin{aligned} \underset{\{\{\theta_t^i\}_{i=1}^N\}_{t=1}^T}{\text{minimize}} \quad & tr(H_{t-1}\Sigma_{r_{t-1}}^{P(\Theta_t)}) \\ \text{subject to} \quad & \sum_{i=1}^{N} \theta_t^i \leq s \end{aligned} \tag{4.51}$$

for all $t = 1, \ldots, T$.

Problem 4.51 is NP-hard [48]. We show conditions under which the MSE is supermodular and subsequently describe a greedy algorithm that has provable bounds.

Remark 4.5.1. *We abuse notation here and use P as both a function and set. When we write $P(\Theta_t)$, P is a function that returns a set of indices corresponding to the indices where the*

elements of Θ_t are 1. These indices correspond to the active sensors. When P is a set (with no function arguments) it refers to the indices of active sensors and not the function that returns the set of indices of active sensors.

4.5.2 Bounds

In this section, we show bounds for the optimization Problem 4.51.

The function $f = tr(H_{t-1}\Sigma_{r_{t-1}}^{P(\Theta_t)})$, which is a function of P and equal to the MSE, e_t (as shown in equation (4.10)), is supermodular under the sufficient conditions in Theorem 4.4.1. It can be shown that f is monotone non-increasing. We need our function to be normalized i.e. $f(\emptyset) = 0$. Hence, we redefine our objective function to be $f = tr(H_{t-1}\Sigma_{r_{t-1}}^{P(\Theta_t)}) - tr(H_{t-1}\Sigma_{r_{t-1}}^{\emptyset})$. $\Sigma_{r_{t-1}}^{\emptyset}$ is the (non-conditional) covariance of r_{t-1} and equals $\Omega_{r_{t-1}}$. The function $tr(H_{t-1}\Sigma_{r_{t-1}}^{P(\Theta_t)}) - tr(H_{t-1}\Sigma_{r_{t-1}}^{\emptyset})$ is also supermodular and monotone non-increasing if $tr(H_{t-1}\Sigma_{r_{k-1}}^{P(\Theta_t)})$ is.

Substituting $tr(H_{t-1}\Sigma_{r_{t-1}}^{P(\Theta_t)}) - tr(H_{t-1}\Sigma_{r_{t-1}}^{\emptyset})$ for f in the supermodular theorem we are left with the inequality utilized in [48] and [51].

$$tr(H_{t-1}\Sigma_{r_{t-1}}^{P^{greedy}}) \geq (1 - 1/e)tr(H_{t-1}\Sigma_{r_{t-1}}^{P^*}) + (1/e)tr(\Sigma_{r_{t-1}}^{\emptyset}) \tag{4.52}$$

for the set P^{greedy} returned from greedy algorithm (Algorithm 4) and the set P^* that is the optimal solution to the sensor scheduling problem. Since $tr(H_{t-1}\Sigma_{r_{t-1}}^{\emptyset}) = tr(H_{t-1}\Omega_{r_{t-1}})$, the bound on

the MSE (e_t) at time t is

$$e_t \geq (1 - 1/e){e_t}^* + (1/e)tr(H_{t-1}\Omega_{r_{t-1}}) \tag{4.53}$$

The first term on the right hand side of the inequality includes $e_t^* = tr(H_{t-1}\Sigma_{r_{t-1}}^{P^*})$ which is the "optimal" MSE or the MSE that comes from sampling the sensor at the optimal measurement times.

Algorithm 4 details the greedy solution given by Nemhauser and Wolsey [15]. For the sensor localization use case, we compare the mean errors of the relaxation approaches from last chapter (Chapter 3) and the mean error from the new greedy approach in Table 4.1. We see that the error from the greedy algorithm is more than double the errors from the relaxation approach in Table 4.1 $(0.38$ vs. $0.16)$. However, the relaxation approaches potentially have a high polynomial complexity, while the greedy algorithm has a low polynomial complexity (ie. $O(NT^2)$). The relaxation approaches versus the greedy algorithm highlight the tradeoff between accuracy and complexity. In this instance, the greedy algorithm does not perform well compared to the SDP approaches. This highlights the importance of having multiple approaches to solve the sensor scheduling problem as one approach may not be the best approach in all problems. It shows that the greedy approach can give poor results and possibly should only be used when matched up against the SDP approach or if one needs a faster solution. The analysis in Theorem 4.4.1 potentially can give some indication when the greedy approach apriori is more desirable. Another observation is the cost from the greedy solution is much more variable than the costs from the SDP approaches. This indicates that the greedy solution is affected more by particular

Algorithm 4 Greedy Algorithm for Sensor Scheduling Problem

Input: f, p.
Output: set P that gives approximate solution to problem
$P \leftarrow \emptyset, i \leftarrow 0$
$i \leq r \; a_i \leftarrow a' \in \underset{a \notin P}{\operatorname{argmax}}(f(P) - f(P \cup \{a\}))$
$P \leftarrow P \cup \{a_i\}, i \leftarrow i + 1$

Max (Mean/Var)	Max-Rank (Mean/Var)	Track (Mean/Var)	Greedy (Mean/Var)
0.1618/6.99e-4	0.1632/6.76e-4	0.1605/5.73e-4	0.3815/0.09

Table 4.1: Error for 3 Different Relaxation Approaches Averaged Over 30 Runs

instantiations of the noise variable than the SDP solutions.

4.6 Minimizing Upper Bounds

In this section, we propose another solution to the optimization Problem 4.51. We now turn our attention to finding bounds on the trace of the covariance. The new solution minimizes an upper bound on the MSE based on the condition number and trace of functions of the observability matrix.

Remark 4.6.1 ([44])**.** *If a matrix A is normal then the condition number ($\kappa(A)$) equals*

$$\kappa(A) = \frac{\|\lambda_{max}(A)\|}{\|\lambda_{min}(A)\|} \tag{4.54}$$

Remark 4.6.2. *Recall, the MSE at time t, $e_t = tr(L_{t-1}\Sigma_{r_{t-1}}^{P(\Theta_t)}L_{t-1}')$ as in equation (4.51) where*

$$\Sigma_{r_{t-1}}^{P(\Theta_t)} = (\sigma^{-2}(S^P(\Theta_t)\Phi_{t-1})'(S^P(\Theta_t)\Phi_{t-1}) + \Omega_{t-1}^{-1})^{-1} = \left(\Omega_{t-1}^{-1} + \sum_{t'=0}^{t}\sum_{i=1}^{N}\theta_{t'}^{i}\hat{N}_{t'}^{i}\right)^{-1} \tag{4.55}$$

where $\sigma^{-2}(S^P(\Theta_t)\Phi_{t-1})'(S^P(\Theta_t)\Phi_{t-1}) = \sum_{t'=0}^{t}\sum_{i=1}^{N}\theta_{t'}^{i}\hat{N}_{t'}^{i}$ *where* $\hat{N}_{t'}^{i}$ *is the* $mt(i-1)$ *to* $mt(i)$ *rows of* Φ_{t-1}.

Also,

$$L_{t-1} = (A^t, A^{t-1}, \ldots, A, I) \tag{4.56}$$

Substituting F_{t-1} *for* $\Sigma_{r_{t-1}}^{-1}$*, we get MSE at time t,* $e_t = tr(L_{t-1}F_{t-1}^{-1\,P(\theta_t)}L_{t-1}')$*. The goal is to find an upper bound on* $tr(L_tF_t^{-1}L_t')$ *for* $t = 1, \ldots, T$.

Remark 4.6.3. *Before finding the upper bound, we note that one possible way to minimize MSE,* $e_t = tr(L_{t-1}\Sigma_{r_{t-1}}^{P(\Theta_t)}L_{t-1}')$ *is to maximize* $\left(\Omega_{t-1}^{-1}+\sum_{t'=0}^{t}\sum_{i=1}^{N}\theta_{t'}^{i}\hat{N}_{t'}^{i}\right)$*. The term* $\sum_{t'=0}^{t}\sum_{i=1}^{N}\theta_{t'}^{i}\hat{N}_{t'}^{i}$ *is connected to the observability matrix. We want to maximize the observability matrix. One metric for maximization is to maximize the minimum eigenvalue of the observability matrix. However, what we will see next is that* $e(t)$ *has an upper bound which is in terms of more than just the minimum eigenvalue of the observability matrix. Rather, we will see that the upper bound incorporates the trace and condition number. And forecasting ahead to what we will show, we see that minimizing* $e(t)$ *depends on the ratio of the maximum to minimum eigenvalue of a function of the observability matrix (namely the condition number) along with the sum of eigenvalues of a function of the observability matrix (namely the trace). And so we go beyond just maximizing the eigenvalues of the observability matrix.*

Note: For a given t, we drop here, for easier reading, the subscripts on L_{t-1} and F_{t-1} so

that L_{t-1} becomes L and F_{t-1}^{-1} becomes F^{-1}. To recap then,

$$L = (A^t, A^{t-1}, \ldots, A, I) \tag{4.57}$$

$$F = (\sigma^{-2}(S^P(\Theta_t)\Phi_{t-1})'(S^P(\Theta_t)\Phi_{t-1}) + \Omega_{t-1}^{-1}) = \big(\Omega_{t-1}^{-1} + \sum_{t'=0}^{t}\sum_{i=1}^{N}\theta_{t'}^{i}\hat{N}_{t'}^{i}\big) \tag{4.58}$$

Proposition 4.6.1 (Upper Bound [3])**.** *The following upper bound holds*

$$tr(LF^{-1}L') \leq tr\Bigg(\begin{bmatrix} U_R^{-1} & 0 \\ 0 & I \end{bmatrix} QFQ' \begin{bmatrix} L_R^{-1} & 0 \\ 0 & I \end{bmatrix}\Bigg)^{-1} \tag{4.59}$$

Proof. We first take the QR decomposition of L', so that we get $L' = QR$ where Q is an orthogonal matrix and R is an upper triangular matrix and subsequently

$$LF^{-1}L' = R'Q'F^{-1}QR \tag{4.60}$$

Letting $R = \begin{bmatrix} U_R \\ 0 \end{bmatrix}$ and $R' = \begin{bmatrix} L_R & 0 \end{bmatrix}$ where U_R is an upper triangular matrix and L_R is a lower triangular matrix, we get

$$LF^{-1}L' = \begin{bmatrix} I & 0 \end{bmatrix}\Bigg(\begin{bmatrix} U_R^{-1} & 0 \\ 0 & I \end{bmatrix} QFQ' \begin{bmatrix} L_R^{-1} & 0 \\ 0 & I \end{bmatrix}\Bigg)^{-1} \begin{bmatrix} I \\ 0 \end{bmatrix} \tag{4.61}$$

And then,

$$tr(LF^{-1}L') = tr\Bigg(\begin{bmatrix} U_R^{-1} & 0 \\ 0 & I \end{bmatrix} QFQ' \begin{bmatrix} L_R^{-1} & 0 \\ 0 & I \end{bmatrix} \Bigg)^{-1} \tag{4.62}$$

□

For the next lemma we define the Frobenius matrix norm.

Definition 4.6.1. *For matrix A of size $m \times n$ for any $m, n \geq 1$ with elements a_{ij}*

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2} = \sqrt{trace(A^*A)} \tag{4.63}$$

We utilize the following well-known lemma

Lemma 4.6.1. *[10]*

The following is true for matrices A of size $n \times n$: $\frac{n*cond(A)}{tr(A)} \geq tr(A^{-1})$.

Proof. The result follows from,

$$\frac{n * cond(A) cos(A, I) cos(A^{-1}, I)}{tr(A)} = tr(A^{-1}) \tag{4.64}$$

where $cos(A, I) = \frac{tr(A)}{\|A\|_F \sqrt{n}}$ □

We present our work [3] on calculating an upper bound for the MSE. We then propose two algorithms that minimize this upper bound to find an effective sensor scheduling strategy. Our future work, will be to showcase the effectiveness of these algorithms on the sensor localization

use case.

Theorem 4.6.1. *The following is an upper bound on the MSE in Kalman filtering when the covariance of the measurment noise is of the form* $\sigma_v^2 I$.

$$tr(LF^{-1}L') \leq tr\Bigg(\begin{bmatrix} U_R^{-1} & 0 \\ 0 & I \end{bmatrix} QFQ' \begin{bmatrix} L_R^{-1} & 0 \\ 0 & I \end{bmatrix} \Bigg)^{-1} \leq \frac{n * cond\Bigg(\begin{bmatrix} U_R^{-1} & 0 \\ 0 & I \end{bmatrix} QFQ' \begin{bmatrix} L_R^{-1} & 0 \\ 0 & I \end{bmatrix} \Bigg)}{tr\Bigg(\begin{bmatrix} U_R^{-1} & 0 \\ 0 & I \end{bmatrix} QFQ' \begin{bmatrix} L_R^{-1} & 0 \\ 0 & I \end{bmatrix} \Bigg)} \tag{4.65}$$

Proof. Consider the QR decomposition of L' such that $L' = QR$ where Q is an orthogonal matrix and R is an upper triangular matrix where

$$R = \begin{bmatrix} U \\ 0 \end{bmatrix}, \quad R' = \begin{bmatrix} L & 0 \end{bmatrix} \tag{4.66}$$

Therefore, we get

$$LF^{-1}L' = R'Q'F^{-1}QR \tag{4.67}$$

By substituting equation (4.66) we get,

$$LF^{-1}L' = \begin{bmatrix} I & 0 \end{bmatrix} \left(\begin{bmatrix} U^{-1} & 0 \\ 0 & I \end{bmatrix} QFQ' \begin{bmatrix} L^{-1} & 0 \\ 0 & I \end{bmatrix} \right)^{-1} \begin{bmatrix} I \\ 0 \end{bmatrix} \tag{4.68}$$

And we can further write,

$$tr(LF^{-1}L') \leq tr\left(\begin{bmatrix} U^{-1} & 0 \\ 0 & I \end{bmatrix} QFQ' \begin{bmatrix} L^{-1} & 0 \\ 0 & I \end{bmatrix} \right)^{-1} \tag{4.69}$$

$$\leq \frac{n * cond\left(\begin{bmatrix} U^{-1} & 0 \\ 0 & I \end{bmatrix} QFQ' \begin{bmatrix} L^{-1} & 0 \\ 0 & I \end{bmatrix} \right)}{tr\left(\begin{bmatrix} U^{-1} & 0 \\ 0 & I \end{bmatrix} QFQ' \begin{bmatrix} L^{-1} & 0 \\ 0 & I \end{bmatrix} \right)} \tag{4.70}$$

The first inequality comes by dropping the pre and post matrices, $\begin{bmatrix} I & 0 \end{bmatrix}$ and $\begin{bmatrix} I \\ 0 \end{bmatrix}$ and the second inequality (4.70) comes from Lemma $(4.6.1)$ where $A = \left(\begin{bmatrix} U^{-1} & 0 \\ 0 & I \end{bmatrix} QFQ' \begin{bmatrix} L^{-1} & 0 \\ 0 & I \end{bmatrix} \right)$. □

We propose two algorithms that minimize the upper bound in order to find a sensor schedule.

Algorithm 1

In the first algorithm, we maximize the denominator in Theorem 4.6.1. Solve for the maximum of the denominator

$$b^* = \max_{\Theta} \quad tr\Bigg(\begin{bmatrix} U_R^{-1} & 0 \\ 0 & I \end{bmatrix} QF(\Theta)Q' \begin{bmatrix} L_R^{-1} & 0 \\ 0 & I \end{bmatrix} \Bigg) \tag{4.71}$$

Then the optimality gap is:

$$c(ALG) - c(OPT) \leq \frac{n \times cond\Bigg(\begin{bmatrix} U_R^{-1} & 0 \\ 0 & I \end{bmatrix} QF(\theta_b^*)Q' \begin{bmatrix} L_R^{-1} & 0 \\ 0 & I \end{bmatrix} \Bigg)}{b^*} - a^* \tag{4.72}$$

where,

$$\theta_{b^*} = \arg\max_{\Theta} tr\Bigg(\begin{bmatrix} U_R^{-1} & 0 \\ 0 & I \end{bmatrix} QF(\Theta)Q' \begin{bmatrix} L_R^{-1} & 0 \\ 0 & I \end{bmatrix} \Bigg) \tag{4.73}$$

$$a^* = \min_{\Theta} \frac{1}{tr(LF(\theta)L')} \tag{4.74}$$

Alternatively, another algorithm is as follows where we minimize the numerator.

Algorithm 2

In the second algorithm, we minimize the numerator in Theorem 4.6.1. Solve for the minimum of the numerator.

$$b^* = \min_{\Theta} \quad n \times cond\Bigg(\begin{bmatrix} U_R^{-1} & 0 \\ 0 & I \end{bmatrix} QFQ' \begin{bmatrix} L_R^{-1} & 0 \\ 0 & I \end{bmatrix} \Bigg) \tag{4.75}$$

$$\theta_{b^*} = \arg\min_{\Theta} \quad n \times cond\Bigg(\begin{bmatrix} U_R^{-1} & 0 \\ 0 & I \end{bmatrix} QFQ' \begin{bmatrix} L_R^{-1} & 0 \\ 0 & I \end{bmatrix} \Bigg) \tag{4.76}$$

Then the optimality gap is:

$$c(ALG) - c(OPT) \leq \frac{b^*}{tr\Bigg(\begin{bmatrix} U_R^{-1} & 0 \\ 0 & I \end{bmatrix} QF(\Theta_{b^*})Q' \begin{bmatrix} L_R^{-1} & 0 \\ 0 & I \end{bmatrix} \Bigg)} - a^* \tag{4.77}$$

.

where,

$$\theta_{b^*} = \arg\min_{\Theta} \quad n \times cond\left(\begin{bmatrix} U_R^{-1} & 0 \\ 0 & I \end{bmatrix} QFQ' \begin{bmatrix} L_R^{-1} & 0 \\ 0 & I \end{bmatrix} \right) \tag{4.78}$$

$$a^* = \min_{\Theta} \quad \frac{1}{tr(LFL')} \tag{4.79}$$

A future direction is to devise an algorithm that both maximizes the denominator and minimizes the numerator simultaneously. One possible approach is to use column subset selection as detailed in [60].

4.7 Designing Trusted and Secure Tracking in Wireless Sensor Networks

In this section, we apply sensor scheduling to a new problem involving security that is detailed in [61]. One of the impetuses for sensor scheduling is the heavy communication cost associated with centralized tracking. In the centralized Kalman filter, or any other estimation algorithm, all the sensor nodes must communicate with a central fusion center. This leads to bottlenecks in communications. A distributed Kalman filtering approach is one workaround where sensors only communicate with their neighbors.

Another problem that arises in all wireless sensor networks is the presence of corrupted or malicious sensors. The question becomes how to detect malicious sensors and make productive use of corrupted sensors while at the same time maintaining a distributed Kalman filter scheme. An approach that uses a trusted core at the heart of its algorithm appeared in [61]. There, a two

layer estimation scheme is proposed. The first layer is a hierarchical trust architecture called the trusted core. The trusted particles are a dedicated set of sensors that observe the dynamics of interest. They are installed with higher levels of security. The second layer consists of sensor nodes other than the trusted particles. They are more numerous in number and they have the ability to talk with one or more trust particles. However, there is a cost to communications with trusted particles and such communication may occur infrequently. The trust particles are reference nodes that estimate the trajectory of interest. The remaining sensor nodes can compare their estimation of the trajectory to the reference trajectory. This way, malicious or corrupted sensors can be detected by comparing the sensor node's trajectory estimation to the trusted particle's trajectory estimation. All trusted particles communicate amongst each other and compute a reference estimation. The remaining sensor nodes only communicate with their neighboring nodes and form their own localized estimation trajectories. The localized estimation trajectories and the reference estimation trajectory are compared. This comparison is used to then compute the weights of the distributed Kalman filter scheme.

Here, we add to this approach by designing a sensor network such that the most effective sensors are selected as trusted particles. As mentioned, the trusted core is a core of sensor nodes that serves as reference nodes to which we can compare potentially malicious or corrupted sensors. In designing the trusted core, we may want to be selective about which of the available sensors should serve the role of trusted sensors. We utilize sensor scheduling to select which sensors would serve best as trusted core members. Our criteria is the same criteria as in sensor scheduling. We look for a subset of sensors that would most minimize the estimation error. These selected sensors will then become the trusted core.

Remark 4.7.1. *The steps for using sensor scheduling in designing trusted tracking in wireless sensor networks are as follows*

1. *Select a subset p of sensors using the previously discussed semidefinite programming algorithm for sensor scheduling. Theses sensors represent the trusted core.*

2. *Compute a distributed Kalman filter scheme with trusted core from [61] (See Section 4.8).*

4.8 Trusted and Secure Tracking in Wireless Sensor Localization Use Case

Consider a similar example to the sensor localization problem used in sensor scheduling in Chapter 3.6. There are 21 sensors available to track a trajectory. Firstly, we choose 4 sensors as part of the trusted core or in other words the set of sensors that are considered to be more secure and trustworthy. These 4 sensors are chosen using our sensor scheduling algorithm. Four other sensors are corrupted by more noise. While, 4 different sensors are maliciously manipulated by an adversary. The remaining 9 sensors are referred to as normal sensors and they are less noisy than the "corrupted" sensors and are not manipulated as in the case of the "adversary" sensors. We track the trajectory open loop and closed loop. Open loop means we use all the sensors other than the trusted core to track the trajectory. Other than the trusted core, sensors only communicate with their neighbors. Closed loop means we use the tracking estimate of the trusted core to compare the trustworthiness of the estimates. Based on how the trusted core estimates and the other sensors estimates differ, we weight differently the estimates of the other sensors. The goal is that the estimates from the sensors that have been either corrupted by noise or adversary would be weighted less than the "normal" sensors. There are three figures (4.2, 4.3, 4.4) showing the performance of open and close loop tracking from the perspective of three different sensors -

sensor 1, 12, and 19. Each sensor communicates with their neighbors and the trusted core. Note, the dotted line indicates if the sensors are neighbors. In Figure 4.1, we compare the tracking error with both the open and closed loop approaches. Not surprisingly, closed loop outperforms open loop. And finally, in Figure 4.5 we compare using sensor scheduling to choose the trusted core versus randomly choosing the trusted core. In Figure 4.5, we see the benefit of using sensor scheduling to choose the trusted core.

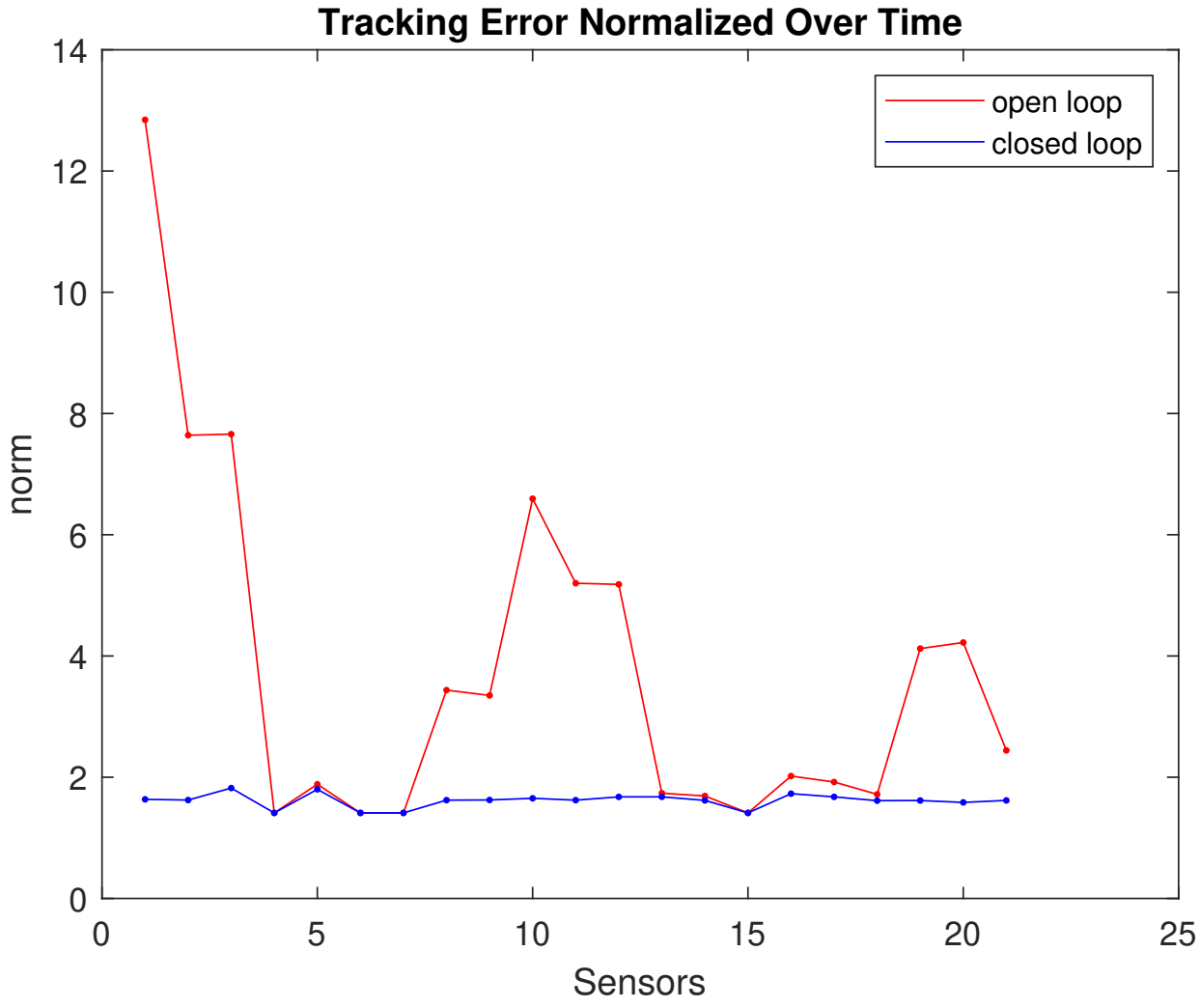

Figure 4.1: Difference between tracking error for open and closed loop

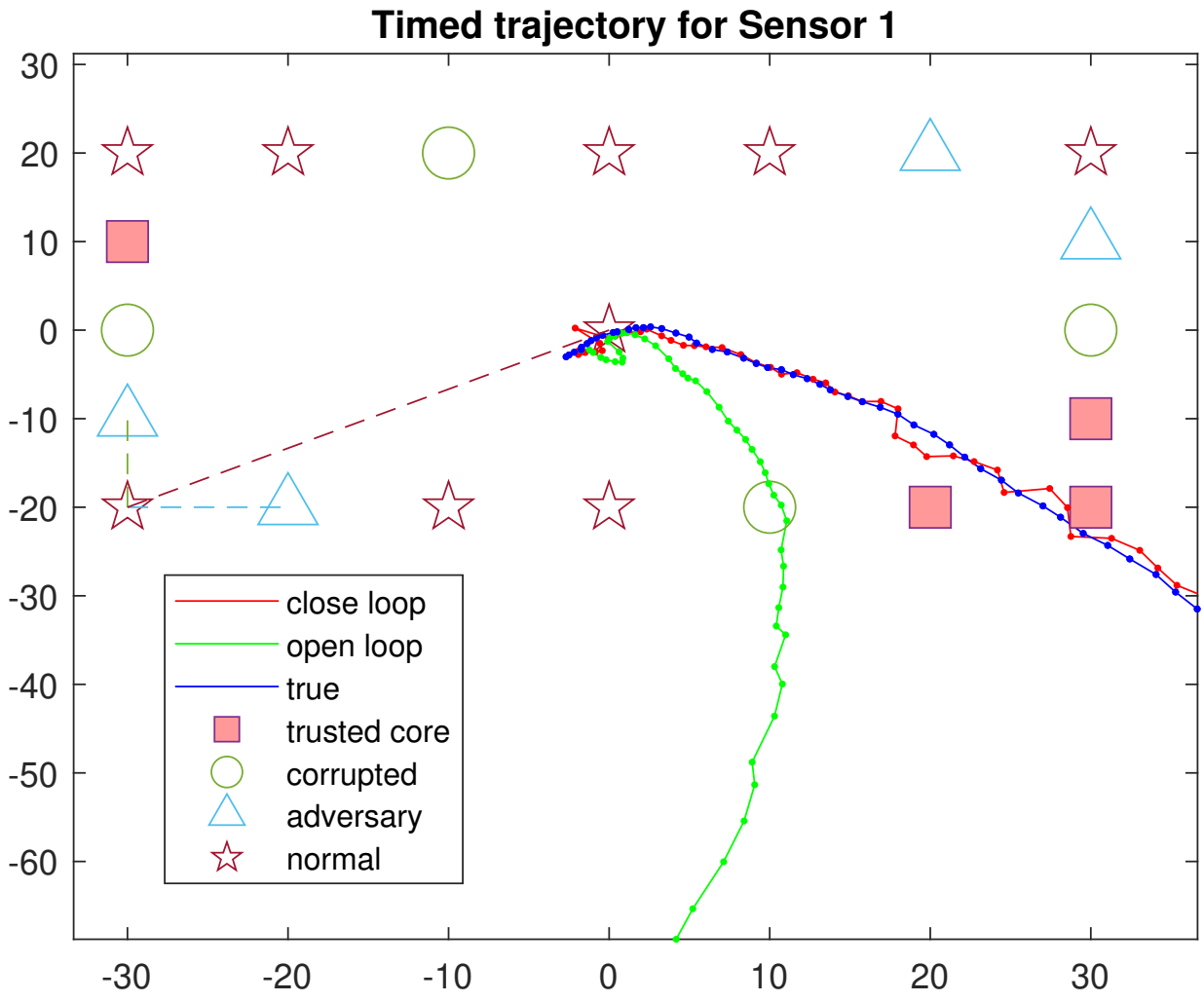

Figure 4.2: Track estimate for sensor 1

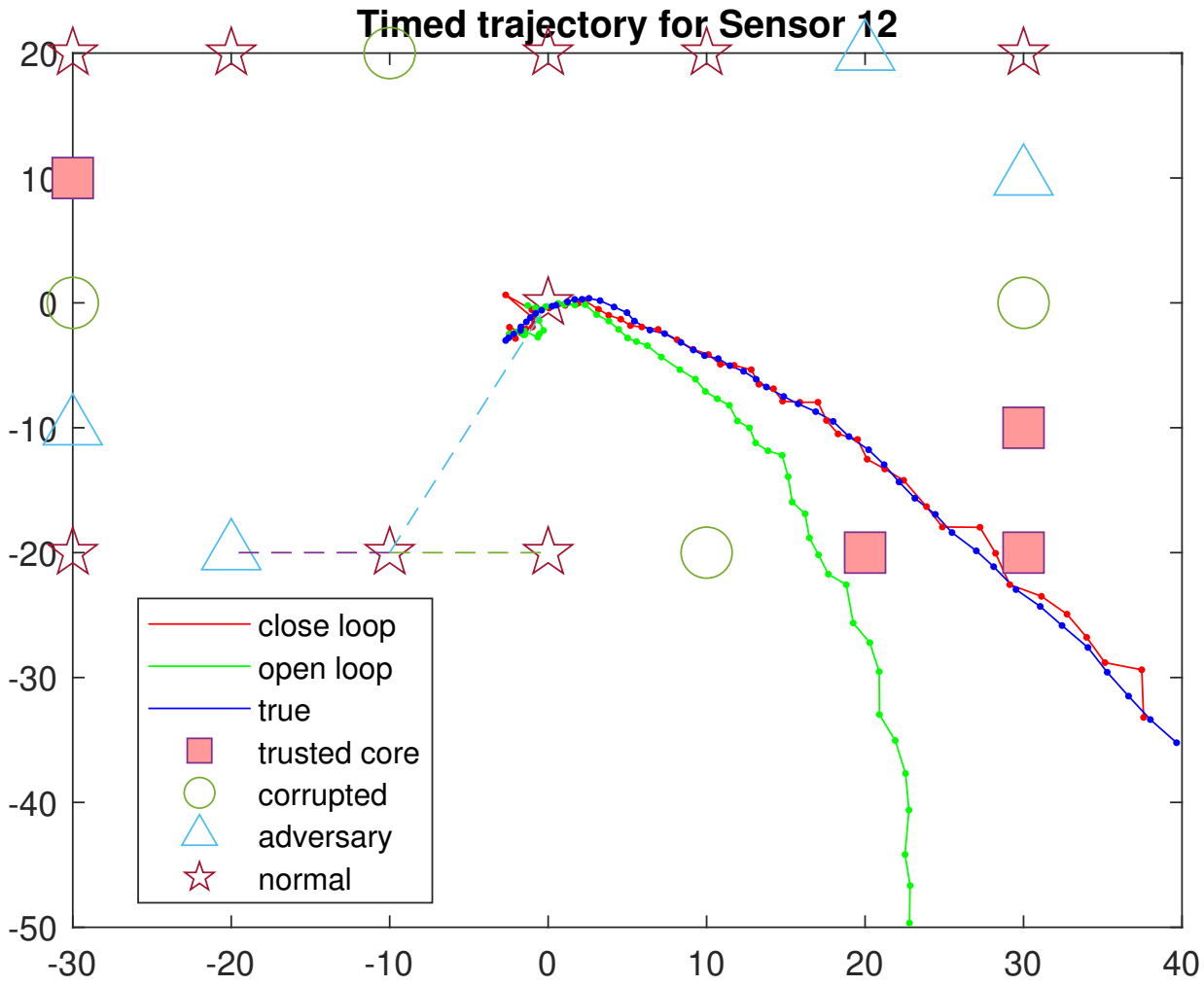

Figure 4.3: Track estimate for sensor 12

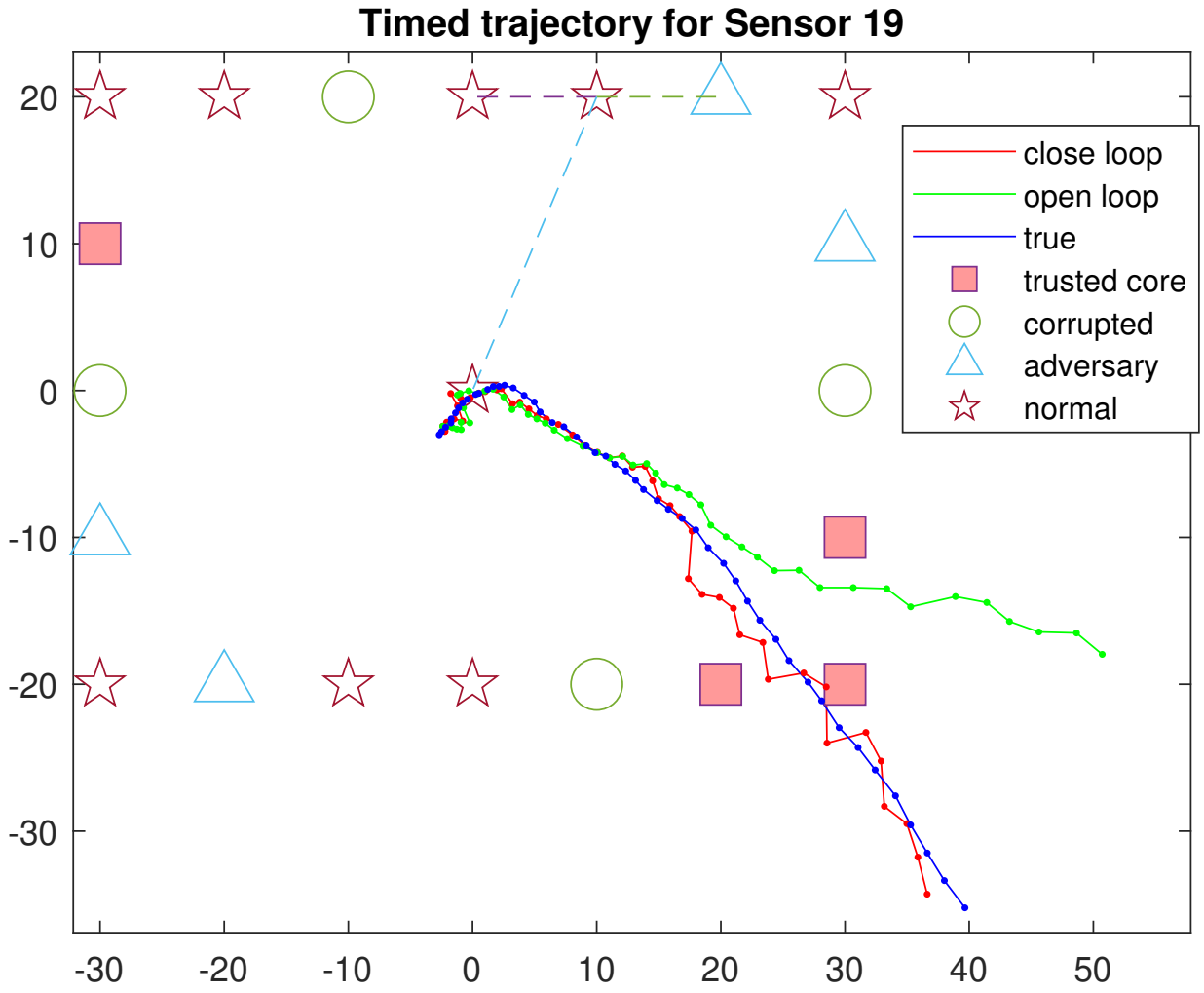

Figure 4.4: Track estimate for sensor 19

It can be seen, as expected, that the closed loop track, namely the one that incorporates estimates from the trusted core to properly weigh the importance of sensors, far outperforms the open loop track, where no such communication with the trusted core occurs. We also compared the results of using sensor scheduling to choose the trusted core versus randomly choosing the trusted core. We see in Figures 4.2, 4.3, 4.4, 4.5 that using sensor scheduling for choosing the trusted core improves the tracking.

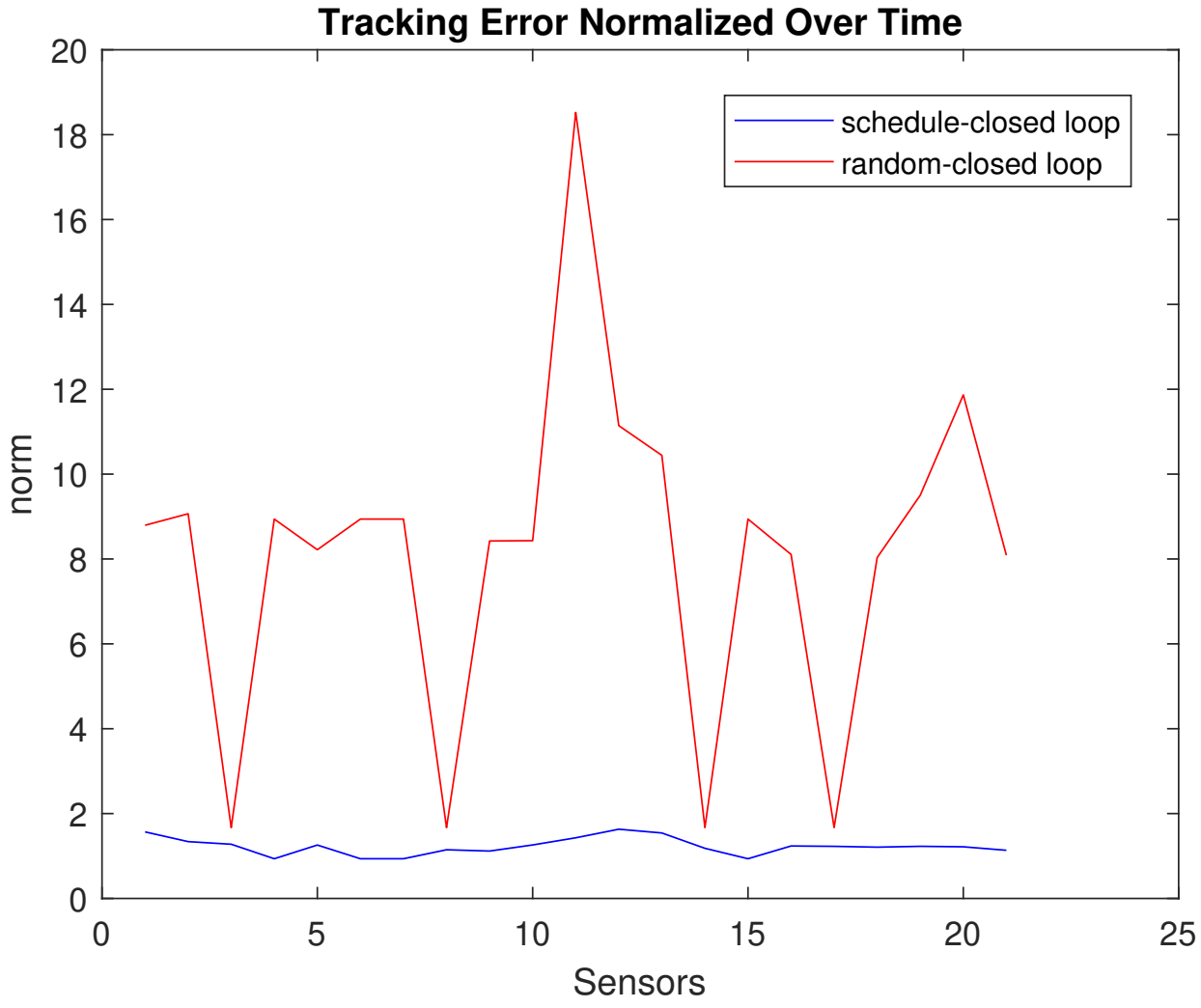

Figure 4.5: Trusted distributed estimation with two approaches to choosing the trusted core: 1. SDP-based sensor selection (Section 3.4) and 2. random sensor selection.

4.9 Conclusion

In this chapter, we proposed algorithms to solve the sensor scheduling (and sensor sampling) problem. The first algorithm is a relaxed version of a mixed-integer semi-definite program. We relax the mixed integer program in three ways. One of the ways is a covariance tracking algorithm. We implement this algorithm on two examples. The first example is a sensor localization

case as discussed in Section 3.6. The second example is to design a trusted wireless network as discussed in Section 4.7 . Additionally, we discuss sufficient conditions (Section 4.4) under which a greedy solution can provide an approximately optimal solution. Lastly, we find an upper bound (Section 4.6) on the estimation error in terms of the condition number and trace of a function of the observability matrix. A proposed sensor scheduling algorithm would minimize this upper bound.

In the future, we can explore algorithms that would optimally minimize the upper bound. The upper bound is a fraction, and so the challenge is maximizing the denominator and minimizing the numerator simultaneously. Additionally, the discussion until now for sensor scheduling has been exclusively in the linear setting. A future direction would be to tackle sensor scheduling for nonlinear systems. One possible approach would be to formulate the sensor scheduling problem as a POMDP and then use Monte-Carlo sampling as in [38]. Monte-Carlo sampling has been studied in MDPs in [62] and POMDPs in [63].

Chapter 5: Sensor Selection for Robust Filtering

In this chapter, we show that sensor scheduling doesn't only apply to Kalman filtering estimation, but it can also apply to robust (H_∞) estimation. There has been much work including our own on the problem of sensor scheduling in the Kalman filtering case. There has been little work on sensor scheduling in the H_∞ filtering case. In 2001, Savkin et al. [64] proposed a Riccati equation solution to the problem of sensor scheduling in the H_∞ setting. In this chapter, we propose a mixed-integer semidefinite program solution to the sensor selection problem in the H_∞ filter continuous-time setting. This follows the same approach as we did for the sensor scheduling problem in the Kalman filtering setting. The sensor selection problem differs from the sensor scheduling problem in that it requires only a one-time choice of sensors as opposed to the sensor scheduling problem where different set of sensors are chosen at different times. Our work on sensor selection for H_∞ filtering is presented in this chapter.

5.1 Problem Formulation

The H_∞ filtering problem differs from the Kalman filtering problem. In Kalman filtering, the goal is to minimize the expected square error where the disturbance is random noise with a

given distribution. In the H_∞ filtering [11], the goal is to minimize the error subject to a worst-case disturbance and the disturbance is deterministic but unknown. See Chapter 2 (Section 2.5.2) for more details on H_∞ filtering.

In the multi-sensor case, consider the dynamics (x) and the N, sensor (y) equations. There are N sensors.

$$\dot{x_t} = Ax_t + Bw_t \tag{5.1}$$

$$y_t^i = C^i x_t + v_t^i \quad \text{for} \quad i \in \{1, \ldots, N\} \tag{5.2}$$

Assume throughout this chapter that A is stable.

Define, the parameter $\theta^i \in \{0, 1\}$ to indicate if sensor i is selected. If $\theta^i = 1$, sensor i is selected and if $\theta^i = 0$, sensor i is not selected. Let $P(\cdot)$ be the function that returns the set of selected sensors. Let $P(\theta)$ be the set of indices (in increasing order) of active sensors or in other words, $P(\theta) = \{j \in \{1, \ldots, N\} \quad \text{s.t} \quad \theta^j = 1\}$.

At most, $p < N$ sensors are selected so that $\sum_{i=1}^{N} \theta^i = p$.

Denote, the sensor selection strategy, $\theta = \begin{bmatrix} \theta^1 \\ \vdots \\ \theta^N \end{bmatrix}$.

Therefore, the relevant sensor equations are the subset of sensor equations corresponding to the

active sensors. We denote the relevant sensor equations as

$$y_t^{P(\theta)} = C^{P(\theta)} x_t + v_t^{P(\theta)} \tag{5.3}$$

where $y_t^{P(\theta)} = vec\{y_t^i\}_{i \in P(\theta)}$, $C^{P(\theta)} = vec\{C^i\}_{i \in P(\theta)}$, and $v_t^{P(\theta)} = vec\{v_t^i\}_{i \in P(\theta)}$.

Consider the weighted norm of vector $x \in \mathbb{R}^n$ for any positive definite matrix R and for any finite $n \geq 1$, $\|x\|_R := \sqrt{x'Rx}$ as in Section 2.5.2. Note, as R is a positive definite matrix, there exists a unique square root, $R^{1/2}$.

Consider the ratio, r of estimation error, $e = x - \hat{x}(\theta)$ to the supremum of all noise (w, v). Such a ratio captures the transfer function from estimation error to noise in the time domain. See Section 2.5.2 for more information.

$$r = \inf_{\hat{x}, \theta \in \{0,1\}^N} \sup_{w,v} \frac{\int_0^\infty \|x - \hat{x}(\theta)\|_R^2 dt}{\int_0^\infty (\|w_t\|_2^2 + \|v_t^{P(\theta)}\|_2^2) dt} \tag{5.4}$$

Thus, our goal is to find an estimator, $\hat{x}(\theta)$, such that

$$r < \gamma^2 \tag{5.5}$$

This is similar to the bound (2.65).

In other words, $\hat{x}$ is the variable that leads to the bound (5.5) being met.

Equation (2.65) can equivalently be written as (exactly the same way as equation (2.66))

$$\inf_{\hat{x},\theta\in\{0,1\}^N} \sup_{w,v} \int_0^\infty \|x-\hat{x}(\theta)\|_R^2 - \gamma^2(\|w_t\|_2^2 + \|v_t^{P(\theta)}\|_2^2)dt \leq 0 \tag{5.6}$$

The scheduler (θ) aims to minimize the cost function while the disturbance functions (w, v) aim to maximize the cost function. The following proposition (Proposition (5.1.1)) is the formulation of the sensor selection H_∞ filtering setting.

Proposition 5.1.1. *The H_∞ filtering sensor selection problem is formulated with the following linear quadratic (LQ) zero-sum game as a performance criteria,*

$$\inf_{\hat{x},\theta\in\{0,1\}^N} \sup_{w,v} \int_0^\infty \|x-\hat{x}(\theta)\|_R^2 - \gamma^2(\|w_t\|_2^2 + \|v_t^{P(\theta)}\|_2^2)dt \tag{5.7}$$

The constraints are as follows,

$$\dot{x_t} = Ax_t + Bw_t \tag{5.8}$$

$$y_t^{(P(\theta))} = C^{(P(\theta)}x_t + v_t^{P(\theta)} \tag{5.9}$$

$$\theta^i \in \{0,1\} \tag{5.10}$$

$$\sum_{i=1}^N \theta^i = p \tag{5.11}$$

Remark 5.1.1. *As mentioned in Section 2.5.2 and in [18] a solution to the H_∞ filtering with performance criteria (5.7) is the same solution that meets the bounds in (5.5) and (5.6). The reason for this is that the performance criteria (5.7) will either be zero or positive infinite because it is a quadratic form under linear constraints (5.8) and (5.9).*

5.2 Riccati Equation Solution

We present in equations (5.17)-(5.21) a related but an alternative Riccati equation solution to the problem defined in Proposition 5.1.1, to the one presented by Savkin et al. [64] and Shaked et al. [65].

We first present the solution to the H_∞ filtering problem when there is no sensor selection (and subsequently no θs). Based on this, we then show the solution to the H_∞ filtering problem with sensor selection.

We begin with the conditions for the H_∞ filtering problem with no sensor selection. As sated in Theorem 2.5.1, the discrete time algebraic (steady-state) Riccati equation solution for the H_∞ filtering problem is

$$AP + PA' - P(\sum_{i=1}^{N} C^{i\prime}C^i - \gamma^{-2}R)P + BB' = 0 \tag{5.12}$$

which states that if there exists a $P \geq 0$ to equation (5.12) then an H_∞ filter can be found.

In the information filter formulation, $Q := P^{-1}$ equation (5.12) can be written as,

$$QA + A'Q - \sum_{i=1}^{N} C^{i\prime}C^i + \gamma^{-2}R + QBB'Q = 0 \tag{5.13}$$

which means that if there exists a $Q \geq 0$ to equation (5.13) then an H_∞ filter can be found.

As stated in reference [11] (Theorem 35), where the bounded real lemma is used (and as stated in Theorem 2.5.1), an equivalent condition for finding a solution to the H_∞ problem (i.e. the bound (5.5) holding) is that there exists a $Q > 0$ such that

$$QA + A'Q - \sum_{i=1}^{N} C^{i\prime}C^i + \gamma^{-2}R + QBB'Q < 0 \tag{5.14}$$

As stated in [65] and equation (2.71), the update for the estimator $\hat{x}$ is

$$\dot{\hat{x}} = A\hat{x} + \sum_{i=1}^{N} K^i(y^i - C^i\hat{x}) \tag{5.15}$$

$$K^i = Q^{-1}C^{i\prime} \tag{5.16}$$

We now consider the H_∞ problem with sensor selection. For our sensor selection problem for a solution to exist to the problem defined in Proposition 5.1.1 there must exist a $Q \geq$ such that

$$QA + A'Q - \sum_{i=1}^{N} \theta^i C^{i\prime}C^i + \gamma^{-2}R + QBB'Q < 0 \tag{5.17}$$

which is a variation of inequality (5.14). The update for the estimator $\hat{x}$ in the sensor selection problem is

$$\dot{\hat{x}} = A\hat{x} + \sum_{i=1}^{N} \theta^i K^i(y^i - C^i\hat{x}) \tag{5.18}$$

$$K^i = Q^{-1}C^{i\prime} \tag{5.19}$$

which is a variation of update equation (5.15). And the following further conditions representing the constraints on active sensors are as follows

$$\theta^i \in \{0,1\} \quad \text{is 1 if active and 0 if inactive} \tag{5.20}$$

$$\sum_{i=1}^{N} \theta^i = p \tag{5.21}$$

5.3 Mixed-Integer SDP Solution

We present, as far as we know, the first mixed-integer semidefinite program (SDP) solution to the sensor selection H_∞ problem, reduced to solving the system of inequalities and equations (5.17), (5.20), and (5.21). The Algebraic Riccati Inequality (ARI) shown in inequality (5.17) can be written as the following linear matrix inequality (LMI) by the Schur complement. See also Section 2.2 and reference [11] pg. 11 for a description of the Schur Complement along with reference [11] for use of the Schur complement with regard to Algebraic Riccati Inequalities. For a recap of the Schur complement, consider the symmetric matrix

$$X = \begin{bmatrix} \hat{A} & \hat{B} \\ \hat{B}' & \hat{C} \end{bmatrix} \tag{5.22}$$

If $\hat{A}$ is invertible, then by the Schur complement X is negative definite if and only if $\hat{A} < 0$ and $\hat{C} - \hat{B}'\hat{A}^{-1}\hat{B} < 0$. Letting $-BB' = \hat{A}, Q = \hat{B}$, and $\hat{C} = QA + A'Q - \sum_{i=1}^{N} \theta^i C^{i\prime} C^i + \gamma^{-2} R$

then we get that (5.17) is equivalent to $Q < 0$ and the LMI (5.23) since $-BB' < 0$,

$$\begin{bmatrix} -(BB')^{-1} & Q \\ Q' & QA + A'Q - \sum_{i=1}^{N} \theta^i C^{i\prime} C^i + \gamma^{-2} R \end{bmatrix} < 0 \tag{5.23}$$

Alternatively, we can write (5.17) as $Q > 0$ and

$$A'Q + QA - \sum_{i=1}^{N} \theta^i C^{i\prime} C^i - \begin{bmatrix} R^{1/2} & QB \end{bmatrix} \begin{bmatrix} -\gamma^2 & 0 \\ 0 & -I \end{bmatrix}^{-1} \begin{bmatrix} R^{1/2} \\ B'Q \end{bmatrix} < 0 \tag{5.24}$$

We can check this out by multiplying the matrices in (5.24) to get (5.17). Equivalently, and by another application of the Schur complement [[11], pg. 11], we can write (5.24) as $Q > 0$ and the LMI (5.25) below

$$\begin{bmatrix} QA + A'Q - \sum_{i=1}^{N} \theta^i C^{i\prime} C^i & R^{1/2} & QB \\ R^{1/2} & -\gamma^2 & 0 \\ B'Q & 0 & -I \end{bmatrix} < 0 \tag{5.25}$$

Indeed, the Schur complement as used in [[11], pg. 11] shows that the inequality,

$$\begin{bmatrix} \hat{A} & \hat{B} \\ \hat{B}' & \hat{C} \end{bmatrix} < 0 \tag{5.26}$$

is equivalent to the following two inequalities $\hat{C} < 0$ and $\hat{A} - \hat{B}\hat{C}^{-1}\hat{B}' < 0$.

If we let $\hat{A} = A'Q + QA - \sum_{i=1}^{N} \theta_i C^{i\prime} C^i$, $\hat{B}' = \begin{bmatrix} R^{1/2} & QB \end{bmatrix}$, and $\hat{C} = \begin{bmatrix} -\gamma^2 & 0 \\ 0 & -I \end{bmatrix}$ in (5.26), we get that $Q > 0$ and (5.24) are equivalent to Q ¿ 0 and (5.25), since the diagonal matrix $\begin{bmatrix} -\gamma^2 & 0 \\ 0 & -I \end{bmatrix}$ is obviously negative definite, we get the LMI in equation (5.25) for the sensor selection problem.

Remark 5.3.1. *To find the optimal (instead of the sub-optimal solution for a given threshold γ) H_∞ filter, the goal is to find the smallest γ. Finding the smallest γ leads to the following optimization problem for the H_∞ sensor selection problem,*

$$\min_{\theta, Q} \quad \gamma \tag{5.27}$$

$$\textit{s.t.} \quad \begin{bmatrix} QA + A'Q - \sum_{i=1}^{N} \theta^i C^{i\prime} C^i & R^{1/2} & QB \\ R^{1/2} & -\gamma^2 & 0 \\ B'Q & 0 & -I \end{bmatrix} < 0, \quad Q > 0, \quad \theta^i \in \{0, 1\} \tag{5.28}$$

$$\sum_{i=1}^{N} \theta^i = p \tag{5.29}$$

5.4 Conclusion

In this chapter, we extended our semidefinite programming solution for sensor scheduling in the Kalman filtering setting to a SDP solution for sensor selection in the robust (H_∞) setting.

In the future, we plan to apply the same previously used relaxation approaches (Section 3.5) to solve the resulting mixed-integer semidefinite program.

Part II: Distributed Risk-Sensitive or Robust Control

Chapter 6: Distributed Risk-Sensitive Control: A Controller Gain Consensus Based Approach

In the previous chapter, we focused on an estimation problem (mostly in the centralized setting). In this chapter, we turn our attention to a control problem in the distributed setting. Additionally, in the previous chapter the criteria for the cost function was the mean of a quadratic function. Here, the criteria is the mean of the *exponential* of a cost function. This is considered a risk-sensitive cost function as taking the exponential accounts for higher moments than just the mean [66]. We address in this chapter an algorithm for distributed risk-sensitive control. We then compare the cost function from the distributed risk-sensitive controller to the cost function from the globally optimal risk-sensitive controller. On the way to distributed risk sensitive control with exponential of integral criteria, we also investigate in this chapter distributed LQG control and develop a novel algorithm based on local agent-neighborhood based computations and consensus.

6.1 Introduction and Literature Review

The most classical and simplest stochastic optimal control problem is the linear quadratic Gaussian (LQG) problem. The LQG setting is one in which Gaussian noise enters the dynamics additively and the cost function is an expectation of the quadratic function of state and control. In risk-sensitive control, we aim to find a controller that not only minimizes the mean of a cost function, but also the higher moments of a cost function. The simplest risk sensitive control problem is the linear exponential quadratic Gaussian (LEQG) problem ([30]). In LEQG, Gaussian noise also enters the dynamics additively, but the cost function is now the expectation of the *exponential* of the LQG cost function. In robust control, we aim to find a controller that minimizes a game-theoretic cost function subject to the maximal disturbance from an exogenous disturbance. A simple robust control problem is the game-theoretic formulation of an H_∞ problem [18]. Finally, in the robust control setting, a deterministic disturbance (instead of stochastic noise) enters the dynamics additively and the goal is to minimize a quadratic cost function of state, control, and maximal disturbance. There is a close connection between LEQG/risk-sensitive control and robust control (ie. H_∞) in that the optimal control solution for one is the optimal solution for the other ([67]).

We now focus on distributed optimal control versus global control. Distributed optimal control has emerged as an area of focus because of communication constraints on the sensors and controllers. In large scale problems such as formation control [68], resource allocation problems, and power grid control problems, multiple agents must work together to minimize a global cost function. Each agent has its own respective local cost function that comprises a part of the global cost

function. The agents are linked by a communication graph that details how different agents can communicate. In our setting, the collaboration graph which designates how different agents interact with each other is the same as the communication graph. The local cost function centered on a given agent depends on the states and controllers of the neighbors of that given agent. In other words, the local cost function of a given agent depends on the same agents that comprise the neighborhood of that agent. Any given agent can be in the neighborhood of multiple agents and therefore there is coupling in the cost function. If each agent would solve for a controller based solely on their local cost function, then there would be a mismatch in controllers for an agent. The neighbors of a given agent would all have their own proposals for the controller of that given agent. Their proposed controllers would not necessarily match. We propose averaging these controllers so we should have one controller for a given agent. Distributed LQG has been studied previously, including for identically dynamically decoupled systems [4]. It has also been studied in non-identical plants in [69] using a bottom-up and top-down approach. In our problem, there is coupling in the cost function. Of course, if there is no coupling at all between the agents whether in the cost function, dynamics equation, or measurement equations, then each agent could formulate their own control action based on their own information. Collectively, this would lead to an optimal controller for the problem. [70].

In the following sections we propose a distributed LQG algorithm where an agent's controller gain is the average of the controller gains computed and suggested by its neighbors. This approach of averaging controller gains is inspired by the work on consensus-based distributed filtering ([7]). We analyze the performance of our consensus-based distributed LQG algorithm compared to a control algorithm that has access to the states and controls of all agents, all models

and all data. This can be found in Section 6.3. Additionally, we propose the same algorithm for the linear exponential quadratic Gaussian (LEQG) problem. Similarly, the algorithm's performance is analyzed in section 6.7. In Section 6.8, we simulate an example for the distributed LQG and LEQG problems to showcase the performance of the proposed distributed algorithm.

6.2 Single Agent LQG

In this section, we summarize the well-known solution to the discrete time, infinite horizon, linear quadratic Gaussian (LQG) control problem.

6.2.1 Problem Setup

First, define the matrix I_v to be the identity matrix in $\mathbb{R}^{v\times v}$ for any $v \in \mathbb{N}$.

Now, let us consider the discrete-time dynamics equation

$$x(t+1) = Ax(t) + Bu(t) + (\sqrt{\epsilon} \cdot I_n)w(t) \tag{6.1}$$

where $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$ and $w(t) \in \mathbb{R}^n$ are the state, control input, and process noise vectors, respectively. The random vector $w(t)$ is i.i.d and for each time is of the form $\mathcal{N}(0, I)$. The initial state, $x_0 \sim \mathcal{N}(\bar{x}_0, P_0)$. Additionally, x_0 and $\{w(t)\}_{t=0}^{T}$ are independent for any value of T.

The time-averaged cost function associated with the infinite horizon LQG problem is

$$J = \lim_{T\to\infty} \frac{1}{T} E[\sum_{t=0}^{T-1} x(t)'Qx(t) + u(t)'Ru(t)] \tag{6.2}$$

$Q \geq 0, R > 0$. We seek control $u(t)$ as a function of $x(t)$ that minimizes the cost function in equation (6.2) constrained to the dynamics in equation (6.1).

6.2.2 Solution

Theorem 6.2.1. *LQG [71]*

Assume, (A, B) is controllable and $(A, Q^{\frac{1}{2}})$ is observable, then the optimal control to the problem in Section 6.2.1 is $u^*(t) = Kx(t)$ where

$$K = -(B'PB + R)^{-1}B'PA \tag{6.3}$$

and where there exists a unique, positive definite solution P to the algebraic Riccati equation

$$P = Q + A'PA - A'PB(R + B'PB)^{-1}B'PA \tag{6.4}$$

Additionally, the optimal average cost is

$$J^* = \epsilon \cdot Tr(P) \tag{6.5}$$

Note that the steady-state feedback gain K does not depend on P_0, the covariance of the initial condition, and also on the covariance of the noise $\epsilon \cdot I_n$. Furthermore, the optimal average cost

does not depend on P_0, the covariance of the initial condition.

Furthermore, the closed loop system with the above control is stable, i.e. all eigenvalues of $(A + BK)$ have magnitude strictly less than 1.

6.3 Multi-Agent LQG

In this section, we formulate a multi-agent LQG problem, in which the agent dynamic equations are dynamically decoupled and the cost functions are coupled. Each agent has their own respective quadratic cost function that depends on the neighbors' states and controllers. Owing to the fact that a given agent is a neighbor to multiple agents, there is coupling in the cost functions. How can we find a sub-optimal solution to this problem where agents communicate and compute using only information from their neighbors (and in this case, as will be seen for our specified proposed algorithm, defacto their neighbor's neighbors)? Next, we detail the problem setup for the multi-agent LQG.

6.3.1 Toy Problem

Consider the toy multi-agent system in Figure 6.1 that will be consulted in this section for ease of understanding the notation. The collaboration network, i.e. the network detailing the interaction of different agents shows which agents (namely their states and controllers) affect the cost of which other agents. In other words, this network details the so-called neighbors of each agent, where the neighbors of an agent i are the agents that affect the cost of agent i. The "local" cost of an agent i is the cost associated with the neighborhood of agent i. Note, that the graph

illustrating these influences is assumed to be undirected, that is if agent i influences j, then also agent j influences i.

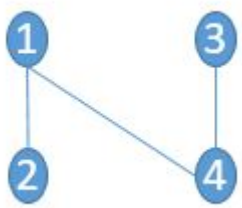

Figure 6.1: Collaboration network of toy example with 4 agents

The following introduce some of the notation we will use, illustrated in the toy example:

- there are 4 agents with states x_1, x_2, x_3, x_4 and controllers u_1, u_2, u_3, u_4
- $n = 1$ (state dimension)
- $m = 1$ (controller dimension)
- the form of A is

$$\begin{bmatrix} A_1 & 0 & 0 & 0 \\ 0 & A_2 & 0 & 0 \\ 0 & 0 & A_3 & 0 \\ 0 & 0 & 0 & A_4 \end{bmatrix} \tag{6.6}$$

where $A_i \in \mathbb{R}$.

- the form of B is

$$\begin{bmatrix} B_1 & 0 & 0 & 0 \\ 0 & B_2 & 0 & 0 \\ 0 & 0 & B_3 & 0 \\ 0 & 0 & 0 & B_4 \end{bmatrix} \tag{6.7}$$

  where $B_i \in \mathbb{R}$.

-

$$Q = \begin{bmatrix} Q_{11} & Q_{12} & Q_{13} & Q_{14} \\ Q_{21} & Q_{22} & Q_{23} & Q_{24} \\ Q_{31} & Q_{32} & Q_{33} & Q_{34} \\ Q_{41} & Q_{42} & Q_{43} & Q_{44} \end{bmatrix}, R = \begin{bmatrix} R_{11} & R_{12} & R_{13} & R_{14} \\ R_{21} & R_{22} & R_{23} & R_{24} \\ R_{31} & R_{32} & R_{33} & R_{34} \\ R_{41} & R_{42} & R_{43} & R_{44} \end{bmatrix} \tag{6.8}$$

  where $Q_{ij}, R_{ij} \in R$. See equation (6.19) for a more precise form for the Q and R matrices in the toy example that indicates which entries are zero.

6.3.2 Single Agent Dynamics

There are N agents in the system. The state of each agent i, $x_i(t) \in R^n$ with control $u_i \in \mathbb{R}^m$ is updated as follows

$$x_i(t+1) = A_i x_i(t) + B_i u_i(t) + \sqrt{\epsilon} \cdot I_n w_i(t) \tag{6.9}$$

for $i \in \{1, \ldots, N\}$ where $A_i \in \mathbb{R}^{n \times n}$, $B_i \in \mathbb{R}^{n \times m}$. Additional assumptions on initial conditions, $x_i(t)$, $w_i(t)$ are detailed in Section 6.2.1. Furthermore, we accommodate different possible dimensions of the state of each agent, and the control of each agent, by embedding them (as needed by adding zeros) in $\mathbb{R}^n$ and $\mathbb{R}^m$ respectively. However for ease of notation and computations we will not explicitly describe these embeddings and we will consider in the sequel the state vectors of each agent as vectors in R^n, and the controls as vectors in R^m.

6.3.3 Notation for Centralized Problem

We consider the following notation to help us in writing the centralized problem.

(A,B) Dynamic Matrices

Consider the matrices $A \in \mathbb{R}^{nN \times nN}$, $B \in \mathbb{R}^{nN \times mN}$.

$$A = blkdiag\big(\{A_i\}_{i=1}^{N}\big) \tag{6.10}$$

$$B = blkdiag\big(\{B_i\}_{i=1}^{N}\big) \tag{6.11}$$

In the toy example, A and B are stated in equations (6.6) and (6.7) in Section 6.3.1.

States, Controllers, and Noise

Consider the vectors $x(t) \in R^{nN}$, $u(t) \in R^{mN}$, and $w(t) \in R^{nN}$.

$$x(t) = vec(\{x_i(t)\}_{i=1}^N) \tag{6.12}$$

$$u(t) = vec(\{u_i(t)\}_{i=1}^N) \tag{6.13}$$

$$w(t) = vec(\{w_i(t)\}_{i=1}^N) \tag{6.14}$$

In the toy example, $x(t) = (x_1'(t), x_2'(t), x_3'(t), x_4'(t))'$, $u(t) = (u_1'(t), u_2'(t), u_3'(t), u_4'(t))'$, and $w(t) = (w_1'(t), w_2'(t), w_3'(t), w_4'(t))'$.

Dynamics

From the block diagonal matrices we get the concatenation of all agent's dynamics and therefore the dynamics of all agents can be captured in the single dynamical system, which is an iterative vector equation with dimension nN.

$$x(t+1) = Ax(t) + Bu(t) + (\sqrt{\epsilon} \cdot I_{nN})w(t) \tag{6.15}$$

Communications and Influence Graphs

Consider the undirected communications graph $\mathcal{G}^{\mathcal{C}} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of sensor nodes, each node represents an agent (or subsystem) and $\mathcal{E}$ is the set of links, such that if $(i, j) \in \mathcal{E}$, then agent i and agent j communicate. (Note, self-edges, (i, i) are included in the set $\mathcal{E}$.) Also, consider the undirected influence graph $\mathcal{G}^{\mathcal{I}}$ where (i, j) is a member of the edge set if agent i influences agent j in cost and vice versa. In other words, there is coupling in the cost. We assume

in this thesis that the influence graph is the same as the communications graph or in other words $\mathcal{G}^{\mathcal{I}} = \mathcal{G}^{\mathcal{C}} = (\mathcal{V}, \mathcal{E})$. In other words, agents influence and communicate with each other if the corresponding $(i, j) \in \mathcal{E}$.

Adjacency Matrix

Consider, the adjacency matrix $\mathcal{L} \in \mathbb{R}^{N \times N}$ of both the communication and influence graphs such that

$$\mathcal{L}_{ij} = \begin{cases} 1 & \text{if} \quad (i,j) \in \mathcal{E} \\ 0 & \text{else} \end{cases} \tag{6.16}$$

In the toy example, $\mathcal{L} \in \mathbb{R}^{4 \times 4} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$.

Q and R Weighting Matrices

Consider the centralized state weighting and control weighting matrices, $Q \in \mathbb{R}^{nN \times nN}$ and

$R \in \mathbb{R}^{mN \times mN}$

$$Q = \begin{bmatrix} Q_{11} & \cdots & Q_{1j} & \cdots Q_{1N} \\ \vdots & \ddots & \vdots & \vdots \\ Q_{j1} & \cdots & Q_{Nj} & \cdots Q_{NN} \end{bmatrix} \tag{6.17}$$

where $Q_{ij} \in \mathbb{R}^{n \times n}$. The $(n \times n)$ block Q_{ij} for $i \neq j$ is nonzero only when $(i, j) \in \mathcal{E}$ and zero otherwise.

$$R = \begin{bmatrix} R_{11} & \cdots & R_{1j} & \cdots R_{1N} \\ \vdots & \ddots & \vdots & \vdots \\ R_{j1} & \cdots & R_{Nj} & \cdots R_{NN} \end{bmatrix} \tag{6.18}$$

where $R_{ij} \in \mathbb{R}^{m \times m}$. The $(m \times m)$ block R_{ij} for $i \neq j$ is nonzero only when $(i, j) \in \mathcal{E}$ and zero otherwise.

In the toy example,

$$Q = \begin{bmatrix} Q_{11} & Q_{12} & 0 & Q_{14} \\ Q_{21} & Q_{22} & 0 & 0 \\ 0 & 0 & Q_{33} & Q_{34} \\ Q_{41} & 0 & Q_{43} & Q_{44} \end{bmatrix}, R = \begin{bmatrix} R_{11} & R_{12} & 0 & R_{14} \\ R_{21} & R_{22} & 0 & 0 \\ 0 & 0 & R_{33} & R_{34} \\ R_{41} & 0 & R_{43} & R_{44} \end{bmatrix} \tag{6.19}$$

We had previously seen the forms for Q and R for the toy example in (6.19). Now, we showed which of those entries in the matrices Q and R are zero.

Centralized Cost

The total cost is

$$J_c = \lim_{T\to\infty} \frac{1}{T} E[\sum_{t=0}^{T-1} x(t)'Qx(t) + u(t)'Ru(t)] \tag{6.20}$$

We seek control $u(t)$ as a function of $x(t)$.

6.3.4 Centralized Control

If there was a central hub for all communications, we would be able to find a global controller that solves the global cost function. Note, this global controller won't necessarily respect the communication structure at each time instance and instead will require that all agents communicate with all other agents, in a path-wise sense (e.g. using multiple hops). This will certainly introduce delays and other synchronization problems between the agents. Additionally, large-scale computation occurs at the central hub instead of being distributed across the agents.

Remark 6.3.1 (LQG Global Centralized Control)**.** *Consider the LQG problem with the dynamics defined in equation* (6.15) *and cost defined in* (6.20)*. Following Theorem 6.2.1if we assume that,* (A, B) *is controllable and* $(A, Q^{\frac{1}{2}})$ *is observable, then the global centralized controller is the following*

$$u_c(t) = Kx(t) \tag{6.21}$$

where

$$K = -(R + B'PB)^{-1}B'PA \tag{6.22}$$

$$P = Q + A'PA - A'PB(R + B'PB)^{-1}B'PA \tag{6.23}$$

where K is an $mN \times nN$ matrix, while P is an $nN \times nN$ square matrix.

From Theorem 6.2.1 we also have that the steady-state feedback gain K does not depend on P_0, the covariance of the initial condition, and also on the covariance of the noise $\epsilon \cdot I_n$. Furthermore, the optimal average cost does not depend on P_0, the covariance of the initial condition.

Furthermore, the closed loop system with the above control is stable, i.e. all eigenvalues of $(A + BK)$ have magnitude strictly less than 1.

6.3.5 Notation for Dynamics of Agent i and its Neighbors

In order to investigate and develop distributed controllers for the multi-agent optimization problem of interest, we first set the notation and expressions for the neighborhoods of each agent.

Neighborhoods

Consider the sets $\mathbb{N}_i$, $i = 1$ to N, which are sets of neighbors of each agent i (we include agent i in the set of its neighbors) or in other words

$$\mathbb{N}_i = \{j| \quad \text{if} \quad (i,j) \in \mathcal{E}\}, i = 1 \dots N \tag{6.24}$$

In the toy example, $\mathbb{N}_1 = \{1, 2, 4\}$, $\mathbb{N}_2 = \{1, 2\}$, $\mathbb{N}_3 = \{3, 4\}$, and $\mathbb{N}_4 = \{1, 3, 4\}$. We denote by $|N_i|$, the cardinality of each neighborhood, that is the number of agents in it.

$(A^{\mathbb{N}_i}, B^{\mathbb{N}_i})$ Dynamics Matrices for Neighborhood Subsystems

We proceed now to describe the dynamics of the states of agents in each neighborhood N_i To this end, consider the matrices $A^{\mathbb{N}_i} \in R^{nN \times nN}$, $B^{\mathbb{N}_i} \in R^{nN \times mN}$, and $\sqrt{\epsilon}^{\mathbb{N}_i} \in R^{nN \times nN}$. (Note, we abuse notation, $\sqrt{\epsilon}^{\mathbb{N}_i}$ is not a mathematical operation but a name for a specific matrix).

First consider the matrix $E_i \in \mathbb{R}^{N \times N}$ such that the j^{th} row of E_i is

$[E_i]_j = \begin{cases} e_j' & \text{if } j \in \mathbb{N}_i \\ 0' & \text{if } j \notin \mathbb{N}_i \end{cases}$ where e_j is the j^{th} canonical basis vector in R^n, that is the vector with all coordinates set to 0, except the j^{th} one which is set to 1.

In the toy example, $E_i \in \mathbb{R}^{4 \times 4}$ for $i = 1, \ldots, N$ is

$$E_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, E_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, E_3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, E_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.25)$$

Then, let us define

$$A^{\mathbb{N}_i} = (E_i \otimes I_n)(A) \tag{6.26}$$

$$B^{\mathbb{N}_i} = (E_i \otimes I_n)(B) \tag{6.27}$$

$$\sqrt{\epsilon}^{\mathbb{N}_i} = (E_i \otimes I_n)(\sqrt{\epsilon} \cdot I_{nN}) \tag{6.28}$$

In other words, $A^{\mathbb{N}_i}, B^{\mathbb{N}_i}$, and $\sqrt{\epsilon}^{\mathbb{N}_i}$ include the j^{th} diagonal sub-block of A, B, and $\sqrt{\epsilon} I_{nN}$ if $j \in \mathbb{N}_i$. All other blocks are zero. We denote by $x^{\mathbb{N}_i}(t)$ the solution of the corresponding dynamic equations for the states of the agents in the neighborhood of agent i, with these block diagonal matrices, after embedding it in a nN vector by adding zeros as needed (see equation (6.47)).

In the toy example, $A^{\mathbb{N}i}$ for $i = 1, \ldots, N$ is

$$A^{\mathbb{N}_1} = \begin{bmatrix} A_1 & 0 & 0 & 0 \\ 0 & A_2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & A_4 \end{bmatrix}, A^{\mathbb{N}_2} = \begin{bmatrix} A_1 & 0 & 0 & 0 \\ 0 & A_2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$A^{\mathbb{N}_3} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & A_3 & 0 \\ 0 & 0 & 0 & A_4 \end{bmatrix}, A^{\mathbb{N}_4} = \begin{bmatrix} A_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & A_3 & 0 \\ 0 & 0 & 0 & A_4 \end{bmatrix}. \tag{6.29}$$

$B^{\mathbb{N}_i}$ for $i = 1, \ldots, N$ is

$$B^{\mathbb{N}_1} = \begin{bmatrix} B_1 & 0 & 0 & 0 \\ 0 & B_2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & B_4 \end{bmatrix}, B^{\mathbb{N}_2} = \begin{bmatrix} B_1 & 0 & 0 & 0 \\ 0 & B_2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$
$$B^{\mathbb{N}_3} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & B_3 & 0 \\ 0 & 0 & 0 & B_4 \end{bmatrix}, B^{\mathbb{N}_4} = \begin{bmatrix} B_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & B_3 & 0 \\ 0 & 0 & 0 & B_4 \end{bmatrix}. \tag{6.30}$$

$\sqrt{\epsilon}^{\mathbb{N}_i}$ for $i = 1, \ldots, N$ is

$$\sqrt{\epsilon}^{\mathbb{N}_1} = \begin{bmatrix} \sqrt{\epsilon}I_n & 0 & 0 & 0 \\ 0 & \sqrt{\epsilon}I_n & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sqrt{\epsilon}I_n \end{bmatrix}, \sqrt{\epsilon}^{\mathbb{N}_2} = \begin{bmatrix} \sqrt{\epsilon}I_n & 0 & 0 & 0 \\ 0 & \sqrt{\epsilon}I_n & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$
$$\sqrt{\epsilon}^{\mathbb{N}_3} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \sqrt{\epsilon}I_n & 0 \\ 0 & 0 & 0 & \sqrt{\epsilon}I_n \end{bmatrix}, \sqrt{\epsilon}^{\mathbb{N}_4} = \begin{bmatrix} \sqrt{\epsilon}I_n & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \sqrt{\epsilon}I_n & 0 \\ 0 & 0 & 0 & \sqrt{\epsilon}I_n \end{bmatrix}. \tag{6.31}$$

Neighborhood States, Controllers, and Noise

Consider the vectors $x^{\mathbb{N}_i}(t) \in R^{nN}$, $u^{\mathbb{N}_i}(t) \in R^{mN}$, and $w^{\mathbb{N}_i}(t) \in R^{nN}$. $x^{\mathbb{N}_i(t)}, u^{\mathbb{N}_i(t)}$, and $w^{\mathbb{N}_i(t)}$ are all composed of N sub-vectors where the j^{th} sub-vector $(x^{\mathbb{N}i}{}_j(t), u^{\mathbb{N}i}{}_j(t), w^{\mathbb{N}i}{}_j(t))$ is as follows

$$x_j^{\mathbb{N}_i}(t) = \begin{cases} \text{local copy of } x_j(t) \in \mathbb{R}^n & \text{if} \quad j \in \mathbb{N}_i \\ 0 & \text{otherwise} \end{cases} \tag{6.32}$$

$$u_j^{\mathbb{N}_i}(t) = \begin{cases} \text{local copy of } u_j(t) \in \mathbb{R}^m & \text{if} \quad j \in \mathbb{N}_i \\ 0 & \text{otherwise} \end{cases} \tag{6.33}$$

$$w_j^{\mathbb{N}_i}(t) = \begin{cases} w_j(t) \in \mathbb{R}^n & \text{if} \quad j \in \mathbb{N}_i \\ 0 & \text{otherwise} \end{cases} \tag{6.34}$$

where $x_j^{\mathbb{N}_i}(t)$ and $u_j^{\mathbb{N}_i}(t)$ are the i^{th} local copies of the j^{th} sub-vectors of $x(t) \in \mathbb{R}^{nN}$ and $u(t) \in \mathbb{R}^{mN}$. Additionally, $w_j(t)$ is the j^{th} sub-vector of $w(t) \in \mathbb{R}^{nN}$.

In the toy example,

$$x^{\mathbb{N}_1}(t) = [x_1^{\mathbb{N}_1\prime}(t), x_2^{\mathbb{N}_1\prime}(t), 0, x_4^{\mathbb{N}_1\prime}(t)]' \tag{6.35}$$

$$x^{\mathbb{N}_2}(t) = [x_1^{\mathbb{N}_2\prime}(t), x_2^{\mathbb{N}_2\prime}(t), 0, 0]' \tag{6.36}$$

$$x^{\mathbb{N}_3}(t) = [0, 0, x_3^{\mathbb{N}_3\prime}(t), x_4^{\mathbb{N}_3\prime}(t)]' \tag{6.37}$$

$$x^{\mathbb{N}_4}(t) = [x_1^{\mathbb{N}_4\prime}(t), 0, x_3^{\mathbb{N}_4\prime}(t), x_4^{\mathbb{N}_4\prime}(t)]' \tag{6.38}$$

$$u^{\mathbb{N}_1}(t) = [u_1^{\mathbb{N}_1\prime}(t), u_2^{\mathbb{N}_1\prime}(t), 0, u_4^{\mathbb{N}_1\prime}(t)]' \tag{6.39}$$

$$u^{\mathbb{N}_2}(t) = [u_1^{\mathbb{N}_2\prime}(t), u_2^{\mathbb{N}_2\prime}(t), 0, 0]' \tag{6.40}$$

$$u^{\mathbb{N}_3}(t) = [0, 0, u_3^{\mathbb{N}_3\prime}(t), u_4^{\mathbb{N}_3\prime}(t)]' \tag{6.41}$$

$$u^{\mathbb{N}_4}(t) = [u_1^{\mathbb{N}_4\prime}(t), 0, u_3^{\mathbb{N}_4\prime}(t), u_4^{\mathbb{N}_4\prime}(t)]' \tag{6.42}$$

$$w^{\mathbb{N}_1}(t) = [w_1'(t), w_2'(t), 0, w_4'(t)]' \tag{6.43}$$

$$w^{\mathbb{N}_2}(t) = [w_1'(t), w_2'(t), 0, 0]' \tag{6.44}$$

$$w^{\mathbb{N}_3}(t) = [0, 0, w_3'(t), w_4'(t)]' \tag{6.45}$$

$$w^{\mathbb{N}_4}(t) = [w_1'(t), 0, w_3'(t), w_4'(t)]' \tag{6.46}$$

Neighborhood Dynamics

In order to develop distributed solutions to the problem, we have to investigate the dynamics of

set of states of each neighborhood of the various agents. State $x^{\mathbb{N}_i}(t) \in R^{nN}$ is updated as follows

$$x^{\mathbb{N}_i}(t+1) = A^{\mathbb{N}_i} x^{\mathbb{N}_i}(t) + B^{\mathbb{N}_i} u^{\mathbb{N}_i}(t) + \sqrt{\epsilon}^{\mathbb{N}_i} w^{\mathbb{N}_i}(t) \tag{6.47}$$

Q^l and R^l Weighting Matrices

To develop a distributed algorithm, we focus on neighborhood dynamics (provided by equation (6.47) and costs and we introduce the state and control cost weighting matrices for the neighborhood of agent l, $Q^{\mathbb{N}_l} \in \mathbb{R}^{nN \times nN}$ and $R^{\mathbb{N}_l} \in \mathbb{R}^{mN \times mN}$. The $(n \times n)$ blocks for $Q^{\mathbb{N}_l}$, $[Q^{\mathbb{N}_l}]_{ii}$ and $[Q^{\mathbb{N}_l}]_{ij}$ (where $i \neq j$) are

$$[Q^{\mathbb{N}_l}]_{ii} = \begin{cases} \frac{Q_{ii}}{\sum_{l=1}^{N} \mathbb{1}_{(i,i) \in \mathbb{N}^l}} & \text{if} \quad (i,i) \in \mathbb{N}^l \\ 0 & \text{else} \end{cases} \tag{6.48}$$

$$[Q^{\mathbb{N}_l}]_{ij} = \begin{cases} \frac{Q_{ij}}{\sum_{l=1}^{N} \mathbb{1}_{(i,j) \in \mathbb{N}^l}} & \text{if} \quad (i,j) \in \mathbb{N}^l \\ 0 & \text{else} \end{cases} \tag{6.49}$$

where the meaning of equation (6.48) is (with slight abuse of notation), "if node i is in the neighborhood of node l, $\mathbb{N}_l$," while the meaning of equation (6.49) is "if both nodes i and j are in the neighborhood of node l, $\mathbb{N}_l$" ; and we have also used the definition below where $\mathbb{1}_{(i,j) \in \mathbb{N}^l}$ =

$$\begin{cases} 1 & \text{if} \quad (i,j) \in \mathbb{N}^l \\ 0 & \text{if} \quad (i,j) \notin \mathbb{N}^l \end{cases}$$

(Note, we normalize the block matrices of $Q^{\mathbb{N}_i}$ with $\sum_{l=1}^{N} \mathbb{1}_{(i,j)\in\mathbb{N}^l}$ in order to have $Q = \sum_{i=1}^{N} Q^{\mathbb{N}_i}$. We likewise do the same with $R^{\mathbb{N}_l}$ so that $R = \sum_{i=1}^{N} R^{\mathbb{N}_i}$.)

In the toy example, $Q^{\mathbb{N}_i}$ for $i = 1, \ldots, N$ is

$$Q^{\mathbb{N}_1} = \begin{bmatrix} \frac{Q_{11}}{3} & \frac{Q_{12}}{2} & 0 & \frac{Q_{14}}{2} \\ \frac{Q_{21}}{2} & \frac{Q_{22}}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{Q_{41}}{2} & 0 & 0 & \frac{Q_{44}}{3} \end{bmatrix}, Q^{\mathbb{N}_2} = \begin{bmatrix} \frac{Q_{11}}{3} & \frac{Q_{12}}{2} & 0 & 0 \\ \frac{Q_{21}}{2} & \frac{Q_{22}}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$Q^{\mathbb{N}_3} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{Q_{33}}{2} & \frac{Q_{34}}{2} \\ 0 & 0 & \frac{Q_{43}}{2} & \frac{Q_{44}}{3} \end{bmatrix}, Q^{\mathbb{N}_4} = \begin{bmatrix} \frac{Q_{11}}{3} & 0 & 0 & \frac{Q_{14}}{2} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{Q_{33}}{2} & \frac{Q_{34}}{2} \\ \frac{Q_{41}}{2} & 0 & \frac{Q_{43}}{2} & \frac{Q_{44}}{3} \end{bmatrix} \tag{6.50}$$

Consider the controller weighting matrices for agent l, $R^{\mathbb{N}_l} \in \mathbb{R}^{nN \times mN}$. The $(n \times m)$ blocks

for $R^{\mathbb{N}_l}$, $[R^{\mathbb{N}_l}]_{ii}$ and $[R^{\mathbb{N}_l}]_{ij}$ (where $i \neq j$) are

$$[R^{\mathbb{N}_l}]_{ii} = \begin{cases} \frac{R_{ii}}{\sum_{l=1}^{N} \mathbb{1}_{(i,i)\in\mathbb{N}^l}} & \text{if} \quad (i,i) \in \mathbb{N}^l \\ 0 & \text{else} \end{cases} \tag{6.51}$$

$$[R^{\mathbb{N}_l}]_{ij} = \begin{cases} \frac{R_{ij}}{\sum_{l=1}^{N} \mathbb{1}_{(i,j)\in\mathcal{N}^l}} & \text{if} \quad (i,j) \in \mathbb{N}^l \\ 0 & \text{else} \end{cases} \tag{6.52}$$

where $\mathbb{1}_{(i,j)\in\mathbb{N}^l} = \begin{cases} 1 & \text{if} \quad (i,j) \in \mathbb{N}^l \\ 0 & \text{if} \quad (i,j) \notin \mathbb{N}^l \end{cases}$

In the toy example, $R^{\mathbb{N}_i}$ for $i = 1, \ldots, N$ is

$$R^{\mathbb{N}_1} = \begin{bmatrix} \frac{R_{11}}{3} & \frac{R_{12}}{2} & 0 & \frac{R_{14}}{2} \\ \frac{R_{21}}{2} & \frac{R_{22}}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{R_{41}}{2} & 0 & 0 & \frac{R_{44}}{3} \end{bmatrix}, R^{\mathbb{N}_2} = \begin{bmatrix} \frac{R_{11}}{3} & \frac{R_{12}}{2} & 0 & 0 \\ \frac{R_{21}}{2} & \frac{R_{22}}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$R^{\mathbb{N}_3} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{R_{33}}{2} & \frac{R_{34}}{2} \\ 0 & 0 & \frac{R_{43}}{2} & \frac{R_{44}}{3} \end{bmatrix}, R^{\mathbb{N}_4} = \begin{bmatrix} \frac{R_{11}}{3} & 0 & 0 & \frac{R_{14}}{2} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{R_{33}}{2} & \frac{R_{34}}{2} \\ \frac{R_{41}}{2} & 0 & \frac{R_{43}}{2} & \frac{R_{44}}{3} \end{bmatrix} \tag{6.53}$$

Neighborhood Cost

Each neighborhood of an agent has a cost function that depends on the agent's state and controller, and its neighbor's states and controllers.

The cost function for the neighborhood subsystem of agent i is

$$J_{nc}^{\mathbb{N}_i} = \lim_{T\to\infty} \frac{1}{T} E[\sum_{t=0}^{T-1} x^{\mathbb{N}_i}(t)' Q^{\mathbb{N}_i} x^{\mathbb{N}_i}(t) + u^{\mathbb{N}_i}(t)' R^{\mathbb{N}_i} u^{\mathbb{N}_i}(t)] \tag{6.54}$$

where $Q^{\mathbb{N}_i} \in \mathbb{R}^{nN\times nN}$ and $R^{\mathbb{N}_i} \in \mathbb{R}^{mN\times mN}$, which are described in equations (6.48), (6.49) and equations (6.51), (6.52), respectively.

The cost function for the collection of all agents' neighborhood subsystems is the sum of the cost functions of all agents' neighborhood systems given by equation (6.54).

$$J_{nc}^{\mathbb{N}} = \lim_{T\to\infty} \frac{1}{T} E[\sum_{i=1}^{N}\sum_{t=0}^{T-1} x^{\mathbb{N}_i}(t)' Q^{\mathbb{N}_i} x^{\mathbb{N}_i}(t) + u^{\mathbb{N}_i}(t)' R^{\mathbb{N}_i} u^{\mathbb{N}_i}(t)] \tag{6.55}$$

$$= \lim_{T\to\infty} \frac{1}{T} E[\sum_{t=0}^{T-1} x^{\mathbb{N}}(t)' Q^{\mathbb{N}} x^{\mathbb{N}}(t) + u^{\mathbb{N}}(t)' R^{\mathbb{N}} u^{\mathbb{N}}(t)] \tag{6.56}$$

where $Q^{\mathbb{N}} = blkdiag(\{Q^{\mathbb{N}_i}\}_{i=1}^{N})$, and $R^{\mathbb{N}} = blkdiag(\{R^{\mathbb{N}_i}\}_{i=1}^{N})$. The size of each block is $nN \times nN$ and $mN \times mN$ respectively (from equations (6.48) to (6.52).

6.3.6 Agent Neighborhood Systems Control

Considering Theorem 6.2.1 and Remark 6.3.1 which detail the global centralized controller, we can similarly write the local (i.e. neighborhood-based) optimal controller for each agent neighborhood (note that an agent can be part of several neighborhoods as noted before). For

example, the local controller from the neighborhood of agent i (note, this is not the distributed controller for agent i, but the contribution to the distributed controller of agent i from the local neighborhood controller of agent i) is shown in Remark 6.3.2.

Remark 6.3.2 (LQG Neighborhood-based Control)**.** *Consider the LQG problem with the dynamics defined in equation* (6.47) *and the cost defined in* (6.54)*. Assume,* $(A^{\mathbb{N}_i}, B^{\mathbb{N}_i})$ *is controllable and* $(A^{\mathbb{N}_i}, (Q^{\mathbb{N}_i})^{\frac{1}{2}})$ *is observable, then the optimal controller for the neighborhood for system of agent* i *is the following*

$$u^{\mathbb{N}_i}(t) = K^{\mathbb{N}_i} x^{\mathbb{N}_i}(t) \tag{6.57}$$

where

$$K^{\mathbb{N}_i} = -(R^{\mathbb{N}_i} + B^{\mathbb{N}_i\prime} P^{\mathbb{N}_i} B^{\mathbb{N}_i})^{-1} B^{\mathbb{N}_i\prime} P^{\mathbb{N}_i} A^{\mathbb{N}_i} \tag{6.58}$$

Riccati equation for the local, neighborhood system of agent i *is*

$$P^{\mathbb{N}_i} = Q^{\mathbb{N}_i} + A^{\mathbb{N}_i\prime} P^{\mathbb{N}_i} A^{\mathbb{N}_i} - A^{\mathbb{N}_i\prime} P^{\mathbb{N}_i} B^{\mathbb{N}_i} (R^{\mathbb{N}_i} + B^{\mathbb{N}_i\prime} P^{\mathbb{N}_i} B^{\mathbb{N}_i})^{-1} B^{\mathbb{N}_i\prime} P^{\mathbb{N}_i} A^{\mathbb{N}_i} \tag{6.59}$$

where the matrices $P^{\mathbb{N}_i}$ *and* $K^{\mathbb{N}_i}$ *are* $\in \mathbb{R}^{nN \times nN}$ *and* $\mathbb{R}^{mN \times nN}$ *respectively.*

From Theorem 6.2.1 we also have that the steady-state feedback gain $K^{\mathbb{N}_i}$ *does not depend on* $P_0^{\mathbb{N}_i}$*, the covariance of the initial condition, and also on the covariance of the noise* $\epsilon \cdot I_n^{\mathbb{N}_i}$*. Furthermore, the optimal average cost does not depend on* $P_0^{\mathbb{N}_i}$*, the covariance of the initial*

condition.

Furthermore, the closed loop system with the above control is stable, i.e. all eigenvalues of $(A^{\mathbb{N}_i} + B^{\mathbb{N}_i} K^{\mathbb{N}_i})$ *have magnitude strictly less than 1.*

Remark 6.3.2 holds true for all $i \in \{1, \ldots, N\}$. Note then, the closed loop system dynamics of the neighborhood $\mathbb{N}_i$ of agent i is given by

$$x^{\mathbb{N}_i}(t+1) = (A^{\mathbb{N}_i} + B^{\mathbb{N}_i} K^{\mathbb{N}_i}) x^{\mathbb{N}_i}(t) + \sqrt{\epsilon}^{\mathbb{N}_i} w^{\mathbb{N}_i}(t) \tag{6.60}$$

6.4 Distributed Control

The goal is to find a distributed controller (in the sense that the controller for each agent i is computed locally where in our scheme local means neighborhood-based) using only locally (neighborhood-based) available information that minimizes the total cost, as given by equation (6.2).

So far we have developed the centralized optimal controller, described in Remark 6.3.1 and equations (6.21), (6.22), (6.23) with the total cost of the multi-agent system given by equation (6.2). We have also developed optimal controllers for each neighborhood of agents, described in Remark 6.3.2 and equations (6.57 (6.58) (6.59), which are block diagonal, with the total cost of the collection of all agents decentralized and decoupled neighborhood systems described by equation (6.56). We emphasize that in each solution we have given explicit expressions for the optimal cost. Furthermore, we have given in each case controllability conditions for the overall multi-agent system, or for each agent neighborhood subsystem, "A-B" matrices that are needed

for the existence of the optimal controllers described. Finally, we have also given observability conditions for the overall multi-agent system, or for each agent neighborhood system, "A-Q" matrices that are needed for the optimal controller to stabilize the closed loop system.

We clarify that the set of agent neighborhood systems, their dynamics, optimal controllers and costs do not constitute a distributed controller for the multi-agent system. Rather they are an intermediate step that we will use to develop our proposed distributed algorithm in what follows in this section. Furthermore the relationship between the controllability and observability assumptions for the set of agent neighborhood systems and those of the overall centralized multi-agent system is left as an open problem. It is clear that the former do not imply the latter. Some further conditions on the system parameters are needed. This investigation is left as a future research problem.

6.4.1 Proposed Distributed Control Algorithm of the Multi-Agent System

We now proceed to develop our proposed distributed algorithm utilizing the agent neighborhood systems controllers developed in sections 6.3.5 and 6.3.6. The following algorithm is our proposed suboptimal, distributed controller. There are two parts to this algorithm. In the first part, each agent i computes a control vector for their respective neighborhood of agents (i.e. for agents j in $\mathbb{N}_i$, from the perspective of agent i). As a given agent i may be in the intersection of multiple neighborhoods of other agents, we compute the distributed control vector for agent i by averaging all the control vectors that the neighboring agents propose for agent i. In other words, agent i is in the cross-hairs of multiple agents with their own respective cost functions (that are functions

of the states of their own neighbors and their own state, and are also influenced by the coupling in the cost) and therefore, each such agent will propose a different control vector for agent i based on their own neighborhood-based local functions and control computations. In part 2, the distributed control vector for agent i is computed by averaging the control vectors proposed by its neighbors.

Algorithm 5 Algorithm for Distributed Control

for $i, k = 1 : N$

For ease of notation we do not include the time argument (i.e. we do not include (t)) in the various state and control functions in this description.

- Solve for $K^{\mathbb{N}_i}$ based on the local, neighborhood-based cost function of agent i using equations (6.57) (6.58), and (6.59). Using $K^{\mathbb{N}_i}$, we pick the k^{th} component of the control vector $u^{\mathbb{N}_i}$ from the local neighborhood-based control of agent i. This k^{th} component is denoted as $u_k^{\mathbb{N}_i}$, and is part of the computation for the control vector for agent k (Note: this is an intermediate step and isn't the end control vector for agent k.)

  In other words,

$$u_k^{\mathbb{N}_i} = [K^{\mathbb{N}_i} x^{\mathbb{N}_i}]_k \tag{6.61}$$

  where $u_k^{\mathbb{N}_i} \in \mathbb{R}^m$.

end

for $l = 1 : N$

- Construct the control vector for agent l by averaging $\{u_l^{\mathbb{N}_i}\}_{\forall i \in \mathbb{N}_l}$. This average is the control vector produced by our distributed scheme for agent l and is denoted as $u_{d,l}$ where

$$u_{d,l} = \sum_{i \in \mathbb{N}_l} \frac{1}{|\mathbb{N}_l|} \sum_{i \in \mathbb{N}_l} u_l^{\mathbb{N}_i} = \sum_{i=1}^{N} \rho_l^{\mathbb{N}_i} u_l^{\mathbb{N}_i} \tag{6.62}$$

Note, $\rho^{\mathbb{N}_i}$ is an $mN \times mN$ block diagonal matrix. The closed loop dynamics of agent l with the feedback control given by equation (6.61) is now given, following equation (6.9), by

$$x_l(t+1) = A_l x_l(t) + B_l u_{d,l}(t) + \sqrt{\epsilon} \cdot I_n w_l(t) \tag{6.64}$$

which is a discrete iteration in R^n.

We note that this dynamics depends on the independent decoupled dynamics of all agents in the neighborhood N_l of agent l, which are described in sections (6.3.5) and (6.3.6). And because of this fact the distributed control signal $u_{d,l}$ in (6.61) cannot be written as a linear function (with some constant gain) of $x_l(t)$. We note that the proposed distributed controller requires some memory to store the values of the control signals computed for the neighborhoods.

Distributed Control algorithm for Toy Example

We now illustrate Algorithm 5, the distributed control algorithm on the toy example.

1. At time 0, nodes $1, 2, 3, 4$ calculate $K^{\mathbb{N}_1}$, $K^{\mathbb{N}_2}$, $K^{\mathbb{N}_3}$, and $K^{\mathbb{N}_4}$ respectively following (6.58) and (6.59).

2. At every time t, nodes $1, 2, 3, 4$ collect state measurements

$$x^{\mathbb{N}_1} = [x_1^{\mathbb{N}_1}(t)', x_2^{\mathbb{N}_1}(t)', 0', x_4^{\mathbb{N}_1}(t)']' \tag{6.65}$$

$$x^{\mathbb{N}_2} = [x_1^{\mathbb{N}_2}(t)', x_2^{\mathbb{N}_2}(t)', 0', 0']' \tag{6.66}$$

$$x^{\mathbb{N}_3} = [0', 0', x_3^{\mathbb{N}_3}(t)', x_4^{\mathbb{N}_3}(t)']' \tag{6.67}$$

$$x^{\mathbb{N}_4} = [x_1^{\mathbb{N}_4}(t)', 0', x_3^{\mathbb{N}_4}(t)', x_4^{\mathbb{N}_4}(t)']' \tag{6.68}$$

3. At every time t, nodes $1, 2, 3, 4$ calculate $u^{\mathbb{N}_1}(t)$, $u^{\mathbb{N}_2(t)}$, $u^{\mathbb{N}_3}(t)$, and $u^{\mathbb{N}_4}(t)$ respectively 6.60.

4. At every time t, node 1 shares with node 2 $u_2^{\mathbb{N}_1}(t)$ and with node 4 $u_4^{\mathbb{N}_1}(t)$.

   Node 2 shares with node 1 $u_1^{\mathbb{N}_2}(t)$.

   Node 3 shares with node 4 $u_4^{\mathbb{N}_3}(t)$.

   Node 4 shares with node 1 $u_1^{\mathbb{N}_4}(t)$ and with node 3 $u_1^{\mathbb{N}_3}(t)$.

5. At every time t, nodes $1, 2, 3, 4$ calculate their controls.

   Node 1 calculates its control $u_{d,1}(t)$ by the averaging scheme: $\frac{1}{3}[u_1^{\mathbb{N}_1}(t) + u_1^{\mathbb{N}_2}(t) + u_1^{\mathbb{N}_4}(t)]$

   Node 2 calculates its control $u_{d,2}(t)$ by the averaging scheme: $\frac{1}{2}[u_2^{\mathbb{N}_1}(t) + u_2^{\mathbb{N}_2}(t)]$

   Node 3 calculates its control $u_{d,3}(t)$ by the averaging scheme: $\frac{1}{2}[u_3^{\mathbb{N}_3}(t) + u_3^{\mathbb{N}_4}(t)]$

   Node 4 calculates its control $u_{d,4}(t)$ by the averaging scheme: $\frac{1}{3}[u_4^{\mathbb{N}_1}(t) + u_4^{\mathbb{N}_3}(t) + u_4^{\mathbb{N}_4}(t)]$

6. At every time t agents (nodes) $1, 2, 3, 4$ execute the iterations described by equation (6.64) above to obtain their closed loop discrete time trajectories with the proposed distributed controller.

7. The values of $x_i(t)$, and of $u_{d,i}(t)$, $i = 1 \dots 4$, are substituted in the cost function (6.2) and the values are averaged over long times and in expectation to get the value of the cost associated with the proposed distributed controller.

In the toy example $\rho^{\mathbb{N}_i} \in \mathbb{R}^{mN \times mN}$ is

$$\rho^{\mathbb{N}_1} = \begin{bmatrix} 1/3 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/3 \end{bmatrix}, \rho^{\mathbb{N}_2} = \begin{bmatrix} 1/3 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{6.69}$$

$$\rho^{\mathbb{N}_3} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1/3 \end{bmatrix} \rho^{\mathbb{N}_4} = \begin{bmatrix} 1/3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1/3 \end{bmatrix}. \tag{6.70}$$

Flow of Information Amongst Agents

We make a brief remark about what is shared between the agents at any given time subject to the communication graph constraints. At any given time t, each agent i will receive measurements of its state and its neighbor's states. It will then compute $K^{\mathbb{N}_i}$ only once and store this (it follows from equations (6.58) and (6.59). Subsequently, it will compute $u^{\mathbb{N}_i}(t)$ and also for all times t. It will then share components of $u^{\mathbb{N}_i}(t)$ with its neighbors at every time. For example, if agent l is in the neighborhood of agent i, agent i will share $u_l^{\mathbb{N}_i}(t)$ with agent l.

New Global Distributed Controller

Define the global, distributed control vector $u^D \in \mathbb{R}^{mN}$ to be (where each $u_{d,l}$, is given by (6.63)).

$$u^D = \begin{bmatrix} u_{d,1} \\ \vdots \\ u_{d,l} \\ \\ \\ u_{d,N} \end{bmatrix} \tag{6.71}$$

Now consider, the block diagonal matrix $\rho^{\mathbb{N}_i} \in \mathbb{R}^{mN \times mN}$ where the block matrices are

$$[\rho^{\mathbb{N}_i}]_{l,l} = \rho_l^{\mathbb{N}_i} \quad \forall \quad l, i = 1, \dots N \tag{6.72}$$

The form of $\rho^{\mathbb{N}_i}$ is

$$\rho^{\mathbb{N}_i} = \begin{bmatrix} \rho_1^{\mathbb{N}_i} & 0 & 0 & 0 \\ 0 & \rho_2^{\mathbb{N}_i} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \rho_N^{\mathbb{N}_i} \end{bmatrix} \tag{6.73}$$

and note that due to (6.63) it holds that $\sum_{i=1}^{N} \rho^{\mathbb{N}_i} = I_{mN}$.

Due, to the impossibility of writing the collection of equations (6.61) (6.63) as a linear feedback control on the state of the multi-agent systems $x(t)$, we introduce here a new distributed control law in Definition 6.4.1. This new distributed control law, u_{DC} is inspired by the computations and equations in (6.61) and (6.63) in Algorithm 5. This new form is subject to analysis as we show later on in this section. In the next remark, we show how the new distributed control law is inspired by (6.61) and (6.63). Where previously the control law comes from averaging the local optimal neighborhood-based controllers at a given agent, in the new distributed control law the control law is derived from averaging the control gain as seen in Definition 6.4.1. This allows us to write the update equations as a function of the global states.

Remark 6.4.1. *The new distributed control law, u_{DC} can be derived by making modifications to equation (6.61) and subsequently equation (6.63). The subsequent replacements we make will show the connection between the original distributed control shown in (6.61) and (6.63) and the newly proposed distributed control law. To get the new distributed control law, we replace the intermediate step for developing the control law in equation (6.61) by substituting the neighborhood states $x^{\mathbb{N}_i}$ (which is updated according to equation (6.60)) for the global states, x^D updated according to the new distributed control law. In other words we replace equation (6.61)*

$$u_k^{\mathbb{N}_i}(t) = [K^{\mathbb{N}_i} x^{\mathbb{N}_i}(t)]_k$$

where $x^{\mathbb{N}_i}(t)$ is updated according to equation (6.60) with

$$\boldsymbol{u_{int,k}^i(t) = [K^{\mathbb{N}_i} x^D(t)]_k} \tag{6.74}$$

where $x^D(t)$ is the global states updated according to the new global, distributed control law which is shown later in equation (6.78) Note, we use here the global state $x^D(t) \in R^{nN}$, but for the purposes of the intermediate step the only relevant part are the states corresponding to neighborhood i. This way, the intermediate step needs to only access the elements in $x^D(t)$ corresponding to neighborhood i. We give this intermediate control (which is just an intermediate step for developing the new distributed control law) the name $u^i_{int,k}(t)$.

Subsequently, we replace the term $u_l^{\mathbb{N}_i}$ in equation (6.63)

$$u_{d,l}(t) = \sum_{i \in \mathbb{N}_l} \frac{1}{|\mathbb{N}_l|} \sum_{i \in \mathbb{N}_l} u_l^{\mathbb{N}_i}(t) = \sum_{i=1}^{N} \rho_l^{\mathbb{N}_i}(t) u_l^{\mathbb{N}_i}(t)$$

with the new intermediate step term $u^i_{int,k}(t)$ (6.74) which gives us,

$$\boldsymbol{u_{new,l} = \sum_{i \in \mathbb{N}_l} \frac{1}{|\mathbb{N}_l|} \sum_{i \in \mathbb{N}_l} u^i_{int,l}(t) = \sum_{i=1}^{N} \rho_l^{\mathbb{N}_i} u^i_{int,l}(t)}$$

Using these modifications, we can now define the new global distributed control.

Definition 6.4.1 (LQG Distributed Control [72])**.** *Define the new global, distributed linear feedback controller (i.e. new global control law), u_{DC} as*

$$u_{DC}(t) = K^{\rho} x(t) = \left(\sum_{i=1}^{N} \rho^{\mathbb{N}_i} K^{\mathbb{N}_i} \right) x(t) \tag{6.75}$$

$$\rho^{\mathbb{N}_i} = \begin{bmatrix} \rho_1^{\mathbb{N}_i} & 0 & 0 & 0 \\ 0 & \rho_2^{\mathbb{N}_i} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \rho_N^{\mathbb{N}_i} \end{bmatrix} \tag{6.76}$$

and for $l, i \in \{1, \ldots, N\}$

$$\rho_l^{\mathbb{N}_i} \in \mathbb{R}^{m \times m} = \begin{cases} \frac{1}{|\mathbb{N}_l|} I_m, & \textit{if} \quad i \in \mathbb{N}_l \\ 0, & \textit{otherwise} \end{cases} \tag{6.77}$$

The global states of the multi-agent system are updated under this new global linear feedback distributed law as follows

$$x^D(t+1) = Ax^D(t) + B \sum_{i=1}^{N} \rho^{\mathbb{N}_i} K^{\mathbb{N}_i} x^D(t) + (\sqrt{\epsilon} \cdot I_{nN}) w(t) \tag{6.78}$$

where $x^D(t)$ is of size $nN \times 1$. $K^\rho = \sum_{i=1}^{N} \rho^{\mathbb{N}_i} K^{\mathbb{N}_i}$ and is of size $mN \times nN$. And, $u_{DC}(t)$ is of size $mN \times 1$.

Proposition 6.4.1. *The controller (in our new distributed control scheme defined in Definition 6.4.1) for agent k is a linear function of the state of agent k, the state of neighbors of agent k, and the state of the neighbors of the neighbors of agent k. In other words, the controller for agent k depends on the 2-hop neighborhood.*

Proof. The controller for agent k in our distributed control scheme depends on the state and

intermediate controllers of the neighbors of agent k. The controllers of the neighbors of agent k in turn depend on the states of its neighbors. □

Assumption 6.4.1. *Assume the following conditions for the remainder of Chapter 6,*

1. $(A^{\mathbb{N}_i}, (Q^{\mathbb{N}_i})^{\frac{1}{2}})$ *is observable for* $i \in \{1, \ldots, N\}$
2. $(A^{\mathbb{N}_i}, B^{\mathbb{N}_i})$ *is controllable for* $i \in \{1, \ldots, N\}$

Definition 6.4.2. *Define the following matrix norm*

$$\|A\| = max_j |\lambda_j| \tag{6.79}$$

where λ_j *are the eigenvalues of* A*.*

Lemma 6.4.1. *If* $((Q^{\mathbb{N}_i})^{\frac{1}{2}}, A^{\mathbb{N}_i})$ *is observable and* $(A^{\mathbb{N}_i}, B^{\mathbb{N}_i})$ *is controllable then following Remark 6.3.2 and Theorem 6.2.1 the local neighborhood-based closed loop matrices* $(A^{\mathbb{N}_i} + B^{\mathbb{N}_i} K^{\mathbb{N}_i})$ *are stable* $\forall\ i \in \{1, \ldots, N\}$ *. That is all their eigenvalues have modulus smaller than 1. Then the closed loop* N *agent system with the new global distributed controller* $u_{DC}(t) = \sum_{i=1}^{N} \rho^{\mathbb{N}_i} K^{\mathbb{N}_i} x(t) = K^{\rho} x(t)$ *results in a global stable closed loop matrix* $A + BK^{\rho}$ *if the following holds true*

$$\|A + B \sum_{i=1}^{N} \rho^{\mathbb{N}_i} K^{\mathbb{N}_i}\| < 1 \tag{6.80}$$

Proof. In general, if $(\sqrt{Q^{\mathbb{N}_i}}, A^{\mathbb{N}_i})$ is observable and $(A^{\mathbb{N}_i}, B^{\mathbb{N}_i})$ is controllable for all i, then the closed loop system is stable with the controller $u^{\mathbb{N}_i} = K^{\mathbb{N}_i} x^{\mathbb{N}_i}$. Global stability of the N agent

system exists if $\|(A + BK^{\rho})\| < 1$.

Remark 6.4.2. *(LMI Stability Conditions for $A + BK^{\rho}$)*

$A + BK^{\rho}$ is Hurwitz (asymptotically stable) if and only if $\exists P > 0$ such that

$$(A + BK^{\rho})' P (A + BK^{\rho}) < P \tag{6.81}$$

This can alternatively be written as

$$P - (A + BK^{\rho})' P (A + BK^{\rho}) > 0 \tag{6.82}$$

Using the Schur Complement we can rewrite condition (6.82) as, $A + BK^{\rho}$ is stable if and only if $\exists P > 0$, Q (where $Q = P^{-1}$), and $\{\rho^{\mathbb{N}_i}\}_{i=1}^{N}$ that solve the following conditions (6.83), (6.84), (6.85).

$$\min tr(P)$$

$$\begin{bmatrix} P & (A + BK^{\rho})' \\ A + BK^{\rho} & Q \end{bmatrix} > 0 \tag{6.83}$$

and

$$\begin{bmatrix} Q & I \\ I & P \end{bmatrix} > 0 \tag{6.84}$$

where $K^{\rho} = \sum_{i=1}^{N} \rho^{\mathbb{N}_i} K^{\mathbb{N}_i}$ *and the structure of* $\rho^{\mathbb{N}_i}$ *is*

$$\begin{bmatrix} * & 0 & 0 & 0 \\ 0 & * & 0 & 0 \\ 0 & 0 & * & 0 \\ 0 & 0 & 0 & * \end{bmatrix} \textit{where} * \textit{is non-zero and} \sum_{i=1}^{N} \rho^{\mathbb{N}_i} = I. \tag{6.85}$$

where $*$ *is non-zero and* $\sum_{i=1}^{N} \rho^{N_i} = I$.

□

Until now, we have discussed the dynamics of the whole system governed by the matrices (A, B) as well as the dynamics of each neighborhood governed by the matrices $(A^{\mathbb{N}_i}, B^{\mathbb{N}_i})$ ((6.26), (6.27)). Now, we will consider the concatenation of all the neighborhood dynamics governed by the matrices $(A^{\mathbb{N}}, B^{\mathbb{N}})$. This will help us bound the cost differences between the global optimal controller and the proposed new distributed controller. To present the dynamics of the concatenation of all neighborhoods, first define $A^{\mathbb{N}}$ and $B^{\mathbb{N}}$ as

$$A^{\mathbb{N}} = blkdiag\big(\{A^{\mathbb{N}_i}\}_{i=1}^{N}\big) \tag{6.86}$$

$$B^{\mathbb{N}} = blkdiag\big(\{B^{\mathbb{N}_i}\}_{i=1}^{N}\big) \tag{6.87}$$

See equations (6.91) and (6.92) for the realization of equations (6.86) and (6.87) in the toy problem in Figure 6.2.

6.5 Concatenation of Neighborhoods

Previously we defined the dynamics when we dealt with the whole system, i.e. all the agents. Here, we deal with the concatenation of all the neighborhoods. A neighborhood includes a certain number of agents and all other agents not included in the neighborhood are padded as zeros. Therefore, in this setting if there are N agents, then the concatenation of all neighborhoods will includes N^2n states. Each neighborhood has N agents (the agents in the whole system not included in a specific neighborhood will be padded with zeros) and there are N neighborhoods. See equation (6.89) for the concatenation of the copies of the agents' states in each neighborhood for the toy problem in Figure 6.2. We present the dynamics when the new global distributed control is used in subsection 6.5.1 and the dynamics when a local optimal control is used in subsection 6.5.2. For both subsections (6.5.1 and 6.5.2), we use the toy example in Figure 6.2 to consider the local (local to a neighborhood) copies of the states, the associated closed loop dynamics, the initial conditions, and the local cost functions.

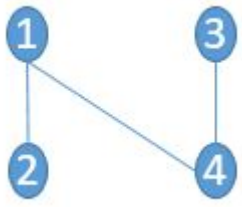

Figure 6.2: Collaboration network of toy example with 4 agents

6.5.1 Dynamics and Cost of Concatenation of Neighborhoods with the New Global Distributed Control

Consider the toy example in Figure 6.2, we look to model the closed loop dynamics and cost for the concatenation or collection of neighborhood systems with the new global distributed control. Each neighborhood system has its own local (local to a neighborhood) copies of the states. In this section (Section 6.5.1) we use the "hat" character ("ˆ") to denote a concatenation or collection of vectors. For example, $\hat{x}^D$ (6.89) is the concatenation of the local copies of the global states for all the neighborhoods (resulting from the new global distributed control law as in equation (6.78)) as opposed to x^D (6.78), which is the global states (resulting from the new distributed control law). We show in this section the dynamics and cost for the neighborhood system with the new global distributed control. In Section 6.5.2, we show the same but for the neighborhood system with the local optimal neighborhood control. We start here with some notation in order to define the dynamics and cost functions.

Notation for Dynamics

For the toy example, the individual agent's, as well as of the collection of neighborhood dynamics, "A and B" matrices are the same as given already in equations (6.29) and (6.30). Similarly the state and control vectors of the neighborhood systems follow the notation of equations (6.38) and (6.42).

Consider the initial condition of concatenation of neighborhoods with the new global distributed

control,

$$x(0) = \begin{bmatrix} x_1(0) \\ x_2(0) \\ x_3(0) \\ x_4(0) \\ \\ x_1(0) \\ x_2(0) \\ x_3(0) \\ x_4(0) \\ \\ x_1(0) \\ x_2(0) \\ x_3(0) \\ x_4(0) \\ \\ x_1(0) \\ x_2(0) \\ x_3(0) \\ x_4(0) \end{bmatrix} \quad \text{(a } 16 \times 1 \text{ vector)} \tag{6.88}$$

Consider the local copies of the states concatenated into one long vector, $\hat{x}^D$ of size $R^{16\times 1}$. For example, we consider three local copies of the state of agent 1, x_1, namely the copy in neighborhood $1, 2,$ and 4. All these local copies are equivalent when using the new global distributed control law, $K^{\rho}x(t)$, defined in Definition 6.4.1. However, the local copy of the state of agent 1 in neighborhood 3 is zero since agent 1 is not a member of neighborhood 3. Consider

the states of the concatenation of neighborhoods with the new global distributed control,

$$\hat{x}^D(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \\ \\ x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \\ \\ x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \\ \\ x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \end{bmatrix} \quad \text{(a } 16 \times 1 \text{ vector)} \tag{6.89}$$

One can construct the global states, x^D from the concatenation of neighborhood states, $\hat{x}^D$ by choosing one of the equivalent copies of the states of each agent. We can create a vector that is a repetition of the global states N times. Consider,

$$\bar{x}^D(t) = \begin{bmatrix} x^D(t) \\ x^D(t) \\ \vdots \end{bmatrix} \tag{6.90}$$

where the global states are repeated N times. Note, there is a difference between $\hat{x}^D$ and $\bar{x}^D$. They are both of the same size. However, some of the terms of $\hat{x}^D(t)$ are zero that are not zero in $\bar{x}^D(t)$.

Consider the block-diagonal matrices for A and B for a concatenation of all the neighborhoods.

$$A^{\mathbb{N}} = \begin{bmatrix} A^{\mathbb{N}_1} & & \\ & \ddots & \\ & & A^{\mathbb{N}_4} \end{bmatrix} = \begin{bmatrix} A_1 & & & & & & & & & & & & & & & \\ & A_2 & & & & & & & & & & & & & & \\ & & 0 & & & & & & & & & & & & & \\ & & & A_4 & & & & & & & & & & & & \\ & & & & \ddots & & & & & & & & & & & \\ & & & & & \ddots & & & & & & & & & & \\ & & & & & & A_1 & & & & & & & & & \\ & & & & & & & 0 & & & & & & & & \\ & & & & & & & & A_3 & & & & & & \\ & & & & & & & & & A_4 & & & & & \end{bmatrix} \quad \text{(a } 16 \times 16 \text{ diagonal matrix)} \tag{6.91}$$

$$B^{\mathbb{N}} = \begin{bmatrix} B^{\mathbb{N}_1} & & \\ & \ddots & \\ & & B^{\mathbb{N}_4} \end{bmatrix} = \begin{bmatrix} B_1 & & & & & & & & & & & & & & & \\ & B_2 & & & & & & & & & & & & & & \\ & & 0 & & & & & & & & & & & & & \\ & & & B_4 & & & & & & & & & & & & \\ & & & & \ddots & & & & & & & & & & & \\ & & & & & \ddots & & & & & & & & & & \\ & & & & & & B_1 & & & & & & & & & \\ & & & & & & & 0 & & & & & & & & \\ & & & & & & & & B_3 & & & & & & & \\ & & & & & & & & & B_4 & & & & & & \end{bmatrix} \tag{6.92}$$

(a 16×16 diagonal matrix)

To solve for the control gain for the concatenated neighborhood with the new distributed control, $\hat{K}^{\rho}$, first consider the modified identity block matrix where some of its block diagonal elements are modified to be zero so that the l^{th} (of N) block diagonal elements follow

$$[I^{\mathbb{N}_i}]_{l,l} = \begin{cases} I_m & \text{if } \mathrm{l} \in \mathbb{N}_i \\ 0 & \text{if } \mathrm{l} \notin \mathbb{N}_i \end{cases} \quad \text{(a } 4 \times 4 \text{ diagonal matrix)} \tag{6.93}$$

where $I^{\mathbb{N}_i}$ is a block diagonal matrix of size 4×4 (or more generally $mN \times mN$).

Define the modified control gain for neighborhood i, $[K^{\rho}]^{\mathbb{N}_i}$, to be such that the rows of K^{ρ} in equation (6.75) are zeroed out corresponding to the agent numbers not in neighborhood i so

that we have the following definition

$$[K^{\rho}]^{\mathbb{N}_i} = I^{\mathbb{N}_i} \times K^{\rho} \quad \text{(a } 4 \times 4 \text{ diagonal matrix)} \tag{6.94}$$

where $[K^{\rho}]^{\mathbb{N}_i}$ is of size 4×4 (or more generally $mN \times nN$). K^{ρ} is defined in Definition 6.4.1 and $I^{\mathbb{N}_i}$ defined in (6.93).

Consider the diagonal matrix, $\hat{K}^{\rho}$, to be the block diagonal matrix consisting of $[K^{\rho}]^{\mathbb{N}_i}$ on the blocks

$$\hat{K}^{\rho} = \begin{bmatrix} [K^{\rho}]^{\mathbb{N}_1} & & \\ & \ddots & \\ & & [K^{\rho}]^{\mathbb{N}_4} \end{bmatrix} \quad \text{(a } 16 \times 16 \text{ diagonal matrix)} \tag{6.95}$$

where $\hat{K}^{\rho}$ is of size 16×16. $[K^{\rho}]^{\mathbb{N}_i}$ is defined in (6.94).

Note, the terms: the global states $x^D(t)$, the repetition of the global states, $\bar{x}^D(t)$ (6.90), and the concatenation of neighborhood states, $\hat{x}^D(t)$. The controller for the concatenation of neighborhoods with the new global distributed control is as follows

$$\hat{u}_{DC}(t) = \hat{K}^{\rho}\bar{x}^D(t) = \begin{bmatrix} [K^{\rho}]^{\mathbb{N}_1} x^D(t) \\ \vdots \\ [K^{\rho}]^{\mathbb{N}_4} x^D(t) \end{bmatrix} \text{(a } 16 \times 1 \text{ vector)} \tag{6.96}$$

where $\hat{u}_{DC}(t)$ is of size 16×1. $[K^{\rho}]^{\mathbb{N}_i}$ is defined in (6.94).

We define a new term for the new distributed controller from the view of neighborhood i

$$[u_{DC}]^{\mathbb{N}_i}(t) = \begin{cases} u_{DC,l}(t) & \text{if } l \in \mathbb{N}_i \\ 0 & \text{else} \end{cases} \quad \text{(a } 4 \times 1 \text{ vector)} \tag{6.97}$$

where $[u_{DC}]^{\mathbb{N}_i}(t)$ is of size 4×1. $u_{DC}(t)$ is defined in Definition (6.4.1).

The i^{th} term in the controller in equation (6.96) can then be written as follows

$$\hat{u}_{DC}(t) = \begin{bmatrix} [u_{DC}]^{\mathbb{N}_1}(t) \\ \vdots \\ [u_{DC}]^{\mathbb{N}_4}(t) \end{bmatrix} \quad \text{(a } 16 \times 1 \text{ vector)} \tag{6.98}$$

so that $[u_{DC}]^{\mathbb{N}_i}(t) = [K^{\rho}]^{\mathbb{N}_i} x^D(t)$ in (6.96). $\hat{u}_{DC}(t)$ is of size 16×1.

In summary, the closed loop dynamics model for the local copies of the states in the concatenated neighborhood setting are given by the block diagonal system,

$$\hat{x}^D(t+1) = A^{\mathbb{N}} \hat{x}^D(t) + B^{\mathbb{N}} \hat{u}_{DC}(t) + (\sqrt{\epsilon} \cdot I_{nN^2}) w(t) \tag{6.99}$$

where $A^{\mathbb{N}}$ and $B^{\mathbb{N}}$ are defined in (6.91) and (6.92), which are all block diagonals with commensurate square blocks with the size of each block being the cardinality of each neighborhood $\mathbb{N}_i$.

In summary, the dynamics model for the controller of the collection of all neighborhood systems is

$$\hat{u}_{DC}(t) = \hat{K}^{\rho}\bar{x}^{D}(t) \tag{6.100}$$

as defined in equation (6.96).

Notation for Cost

We next provide the cost model for the collection of all neighborhood systems.

We present here the cost weighting matrices for the concatenation of neighborhoods. In equations (6.50) and (6.53) we presented the cost weighting matrices for neighborhood i, $Q^{\mathbb{N}_i}$ and $R^{\mathbb{N}_i}$. Here, we present the cost weighting matrices for the concatenation of neighborhoods, referred to

as $Q^{\mathbb{N}}$ and $R^{\mathbb{N}}$.

$$Q^{\mathbb{N}} = \begin{bmatrix} Q^{\mathbb{N}_1} & & \\ & \ddots & \\ & & Q^{\mathbb{N}_4} \end{bmatrix} = \begin{bmatrix} \frac{Q_{11}}{3} & \frac{Q_{12}}{2} & 0 & \frac{Q_{14}}{2} & & & & & & & & \\ \frac{Q_{21}}{2} & \frac{Q_{22}}{2} & 0 & 0 & & & & & & & & \\ 0 & 0 & 0 & 0 & & & & & & & & \\ \frac{Q_{41}}{2} & 0 & 0 & \frac{Q_{44}}{3} & & & & & & & & \\ & & & & \ddots & & & & & & & \\ & & & & & \ddots & & & & & & \\ & & & & & & \frac{Q_{11}}{3} & 0 & 0 & \frac{Q_{14}}{2} & & \\ & & & & & & 0 & 0 & 0 & 0 & & \\ & & & & & & 0 & 0 & \frac{Q_{33}}{2} & \frac{Q_{34}}{2} & & \\ & & & & & & \frac{Q_{41}}{2} & 0 & \frac{Q_{43}}{2} & \frac{Q_{44}}{3} & & \end{bmatrix} \tag{6.101}$$

(a 16×16 block diagonal matrix)

$$R^{\mathbb{N}} = \begin{bmatrix} R^{\mathbb{N}_1} & & \\ & \ddots & \\ & & R^{\mathbb{N}_4} \end{bmatrix} = \begin{bmatrix} \frac{R_{11}}{3} & \frac{R_{12}}{2} & 0 & \frac{R_{14}}{2} & & & & \\ \frac{R_{21}}{2} & \frac{R_{22}}{2} & 0 & 0 & & & & \\ 0 & 0 & 0 & 0 & & & & \\ \frac{R_{41}}{2} & 0 & 0 & \frac{R_{44}}{3} & & & & \\ & & & & \ddots & & & \\ & & & & & \ddots & & \\ & & & & & & \frac{R_{11}}{3} & 0 & 0 & \frac{R_{14}}{2} \\ & & & & & & 0 & 0 & 0 & 0 \\ & & & & & & 0 & 0 & \frac{R_{33}}{2} & \frac{R_{34}}{2} \\ & & & & & & \frac{R_{41}}{2} & 0 & \frac{R_{43}}{2} & \frac{R_{44}}{3} \end{bmatrix} \tag{6.102}$$

(a 16×16 block diagonal matrix)

We can express the cost function and the corresponding dynamics for the new global distributed control in three forms. The first form is the sum of the cost functions for each neighborhood with the new distributed control, $\sum_{i=1}^{N} J_{DC}^{\mathbb{N}_i}$. The second form is the cost function for the concatenation of neighborhoods with the new distributed control, $\hat{J}_{DC}$. And the third form is the cost function for the global states with the new distributed control, J_{DC}.

Cost Function 1 for New Global Distributed Control (sum of local neighborhood cost functions):

Firstly, the cost function of the system can be written as a sum of local, neighborhood cost functions.

Each of the N local, neighborhood cost functions resulting from the new global distributed control can be written in terms of the global states, $x(t)$ and distributed controllers, $[u_{DC}]^{\mathbb{N}_i}(t)$ (6.97) for each neighborhood i.

$$J_{DC}^{\mathbb{N}_i} = \lim_{T\to\infty} \frac{1}{T} E[\sum_{t=0}^{T-1} x(t)' Q^{\mathbb{N}_i} x(t) + [u_{DC}]^{\mathbb{N}_i}(t)' R^{\mathbb{N}_i} [u_{DC}]^{\mathbb{N}_i}(t)] \tag{6.103}$$

The total cost function is the sum

$$\sum_{i=1}^{N} J_{DC}^{\mathbb{N}_i} = \lim_{T\to\infty} \frac{1}{T} E[\sum_{i=1}^{N} \sum_{t=0}^{T-1} x(t)' Q^{\mathbb{N}_i} x(t) + [u_{DC}]^{\mathbb{N}_i}(t)' R^{\mathbb{N}_i} [u_{DC}]^{\mathbb{N}_i}(t)'] \tag{6.104}$$

where $x(t)$ is of size $nN \times 1$ and $[u_{DC}]^{\mathbb{N}_i}(t)$ is of size $mN \times 1$ defined in (6.97). $Q^{\mathbb{N}_i}$ is of size $nN \times nN$ and $R^{\mathbb{N}_i}$ is of size $mN \times mN$ defined in (6.50) and (6.53), respectively.

The following neighborhood-based closed loop dynamics hold

$$[x]^{\mathbb{N}_i}(t+1) = A^{\mathbb{N}_i}[x]^{\mathbb{N}_i}(t) + B^{\mathbb{N}_i}[u_{DC}]^{\mathbb{N}_i}(t) + (\sqrt{\epsilon} \cdot I_{nN}) w(t) \quad \forall i = 1 \ldots N \tag{6.105}$$

$$[u_{DC}]^{\mathbb{N}_i}(t) = [K^{\rho}]^{\mathbb{N}_i} x(t) \tag{6.106}$$

where $[x]^{\mathbb{N}_i}(t) = I^{\mathbb{N}_i} \times x(t)$ ($I^{\mathbb{N}_i}$ is defined in (6.93) and is the global state $x(t)$ with the states not in neighborhoods i zeroed out.) $A^{\mathbb{N}_i}$ is defined in (6.26) and $B^{\mathbb{N}_i}$ is defined in (6.27). $[u_{DC}]^{\mathbb{N}_i}(t)$ is defined in (6.97).

Based on Theorem 6.2.1, the cost function in equation (6.104) can be written as

$$\epsilon(tr(\sum_{i=1}^{N} P_{DC}^{\mathbb{N}_i})) \tag{6.107}$$

where

$$P_{DC}^{\mathbb{N}_i} = (A^{\mathbb{N}_i} + B^{\mathbb{N}_i}K^{\rho})' P_{DC}^{\mathbb{N}_i}(A^{\mathbb{N}_i} + B^{\mathbb{N}_i}K^{\rho}) + (Q^{\mathbb{N}_i} + K^{\rho\prime}R^{\mathbb{N}_i}K^{\rho}) \tag{6.108}$$

Cost Function 2 for New Global Distributed Control (concatenated neighborhood cost function):

The cost can also be written in terms of the concatenated neighborhood states, $\hat{x}(t)$ and the concatenated neighborhood new, global distributed controller $\hat{u}_{DC}(t)$ defined in (6.96),

$$\hat{J}_{DC} = \lim_{T\to\infty} \frac{1}{T} E[\sum_{t=0}^{T-1} \hat{x}(t)' Q^{\mathbb{N}} \hat{x}(t) + \hat{u}_{DC}(t)' R^{\mathbb{N}} \hat{u}_{DC}(t) \tag{6.109}$$

where $\hat{x}(t)$ is of size $nN^2 \times 1$ and $\hat{u}_{DC}(t)$ is of size $mN^2 \times 1$ defined in (6.96). $Q^{\mathbb{N}}$ is of size $nN^2 \times nN^2$ and $R^{\mathbb{N}}$ is of size $mN^2 \times mN^2$ defined in (6.101) and (6.102), respectively.

The following closed loop dynamics for the concatenation of neighborhood systems hold

$$\hat{x}(t+1) = A^{\mathbb{N}}\hat{x}(t) + B^{\mathbb{N}}\hat{u}_{DC}(t) + (\sqrt{\epsilon} \cdot I_{nN^2})\hat{w}(t) \tag{6.110}$$

$$\hat{u}_{DC}(t) = \hat{K}^{\rho}\hat{x}(t) \tag{6.111}$$

$A^{\mathbb{N}}$ is defined in (6.91) and $B^{\mathbb{N}}$ is defined in (6.92). $\hat{u}_{DC}(t)$ is defined in (6.96). $\hat{w}(t)$ is the concatenation of noise $w(t)$ for all neighborhoods and is of size $nN^2 \times 1$.

Based on Theorem 6.2.1, the cost function in equation (6.109) can be written as

$$\epsilon(tr(\sum_{i=1}^{N} \hat{P}_{DC})) \tag{6.112}$$

where

$$\hat{P}_{DC} = (A^{\mathbb{N}} + B^{\mathbb{N}}\hat{K}^{\rho})'\hat{P}_{DC}(A^{\mathbb{N}} + B^{\mathbb{N}}\hat{K}^{\rho}) + (Q^{\mathbb{N}} + \hat{K}^{\rho\prime}R^{\mathbb{N}}K^{\rho}) \tag{6.113}$$

Cost Function 3 for New Global Distributed Control (global state cost function):

Finally, as in the form (6.2), we can express the cost function using global states $x(t)$, global cost weighting matrices Q and R, and global controller $u_{DC}(t)$

$$J_{DC} = \lim_{T\to\infty} \frac{1}{T} E[\sum_{t=0}^{T-1} x(t)'Qx(t) + u_{DC}(t)'Ru_{DC}(t) \tag{6.114}$$

where $x(t)$ is of size $nN \times 1$ and $u_{DC}(t)$ is of size $mN \times 1$ defined in (6.75). Q is of size $nN \times nN$ and R is of size $mN \times mN$.

The following dynamics hold

$$x(t+1) = Ax(t) + Bu_{DC}(t) + (\sqrt{\epsilon} \cdot I_{nN})w(t) \tag{6.115}$$

$$u_{DC}(t) = K^{\rho}x(t) \tag{6.116}$$

$u_{DC}(t)$ is defined in (6.75).

Based on Theorem 6.2.1, the cost function in equation (6.114) can be written as

$$\epsilon tr(P_{DC}) \tag{6.117}$$

where

$$P_{DC} = (A + BK^{\rho})' P_{DC}(A + BK^{\rho}) + (Q + K^{\rho\prime} R K^{\rho}) \tag{6.118}$$

6.5.2 Dynamics of Concatenation of Neighborhoods with Local Optimal Control

As in equation (6.89), consider the local copies of the states concatenated into one long vector, $x^{\mathbb{N}}$ of size $R^{16\times 1}$. Here the controller of the collection of all neighborhood systems is the local optimal control associated with each neighborhood. This is in contrast to Subsection 6.5.1

where the control law is the new global distributed control.

$$x^{\mathbb{N}}(t) = \begin{bmatrix} x^{\mathbb{N}_1}(t) \\ x^{\mathbb{N}_2}(t) \\ x^{\mathbb{N}_3}(t) \\ x^{\mathbb{N}_4}(t) \end{bmatrix} = \begin{bmatrix} x_1^{\mathbb{N}_1}(t) \\ x_2^{\mathbb{N}_1}(t) \\ x_3^{\mathbb{N}_1}(t) \\ x_4^{\mathbb{N}_1}(t) \\ \\ x_1^{\mathbb{N}_2}(t) \\ x_2^{\mathbb{N}_2}(t) \\ x_3^{\mathbb{N}_2}(t) \\ x_4^{\mathbb{N}_2}(t) \\ \\ x_1^{\mathbb{N}_3}(t) \\ x_2^{\mathbb{N}_3}(t) \\ x_3^{\mathbb{N}_3}(t) \\ x_4^{\mathbb{N}_3}(t) \\ \\ x_1^{\mathbb{N}_4}(t) \\ x_2^{\mathbb{N}_4}(t) \\ x_3^{\mathbb{N}_4}(t) \\ x_4^{\mathbb{N}_4}(t) \end{bmatrix} \quad (\text{a } 16 \times 1 \text{ vector}) \tag{6.119}$$

$A^{\mathbb{N}}$ and $B^{\mathbb{N}}$ are the same as in equations (6.91) and (6.92). The controller gain matrices are a concatenation of local neighborhood optimal controller gains.

$$K^{\mathbb{N}} = \begin{bmatrix} K^{\mathbb{N}_1} & & \\ & \ddots & \\ & & K^{\mathbb{N}_4} \end{bmatrix} = \begin{bmatrix} * & * & 0 & * & & & & & & & & & & & & \\ * & * & 0 & * & & & & & & & & & & & & \\ 0 & 0 & 0 & 0 & & & & & & & & & & & & \\ * & * & 0 & * & & & & & & & & & & & & \\ & & & & \ddots & & & & & & & & & & & \\ & & & & & \ddots & & & & & & & & & & \\ & & & & & & & & & & & & & & & \\ & & & & & & & & * & 0 & * & * & & & & \\ & & & & & & & & 0 & 0 & 0 & 0 & & & & \\ & & & & & & & & * & 0 & * & * & & & & \\ & & & & & & & & * & 0 & * & * & & & & \end{bmatrix} \quad (\text{a } 16 \times 16 \text{ matrix}) \tag{6.120}$$

$$u^{\mathbb{N}} = K^{\mathbb{N}} x^{\mathbb{N}} = \begin{bmatrix} K^{\mathbb{N}_1} x^{\mathbb{N}_1} & & \\ & \ddots & \\ & & K^{\mathbb{N}_4} x^{\mathbb{N}_4} \end{bmatrix} = \begin{bmatrix} u^{\mathbb{N}_1} \\ \vdots \\ u^{\mathbb{N}_4} \end{bmatrix} \quad (\text{a } 16 \times 1 \text{ vector}) \tag{6.121}$$

In summary, the closed loop dynamics model for the collection of all neighborhood systems is

$$x^{\mathbb{N}}(t+1) = (A^{\mathbb{N}} + B^{\mathbb{N}} K^{\mathbb{N}}) x^{\mathbb{N}}(t) + (\sqrt{\epsilon} \cdot I_{nN^2}) \hat{w}(t) \quad (\text{a } 16 \times 1 \text{ vector iteration}) \tag{6.122}$$

$\hat{w}(t)$ is the concatenation of noise $w(t)$ for all neighborhoods and is of size $nN^2 \times 1$.

And, in summary we have for the controller the dynamics model for the controller of the collection of all neighborhood systems is

$$u^{\mathbb{N}} = K^{\mathbb{N}} x^{\mathbb{N}} \quad \text{(a } 16 \times 1 \text{ vector)} \tag{6.123}$$

We can consider two equivalent cost functions when using local optimal control. The first is the sum of cost functions for the neighborhoods with local optimal control, $\sum_{i=1}^{N} J^{\mathbb{N}_i}$. The second is the cost function for the concatenation of neighborhoods with local optimal control, $J^{\mathbb{N}}$.

Cost Function 1 for Local Optimal Control (sum of local neighborhood cost functions):

Firstly, we consider the sum of the cost functions for each of the neighborhoods with the local, neighborhood control.

Definition 6.5.1. *The cost function of neighborhood i can be defined as the following when the control gain $K^{\mathbb{N}_i}$ as defined in equation (6.58) is employed*

$$J_{nc}^{\mathbb{N}_i} := \lim_{T\to\infty} \frac{1}{T} E[\sum_{t=0}^{T-1} x^{\mathbb{N}_i}(t)' Q^{\mathbb{N}_i} x^{\mathbb{N}_i}(t) + u^{\mathbb{N}_i}(t)' R^{\mathbb{N}_i} u^{\mathbb{N}_i}(t)] \tag{6.124}$$

The sum of the cost functions for all neighborhoods is

$$\sum_{i=1}^{N} J_{nc}^{\mathbb{N}_i} := \lim_{T\to\infty} \frac{1}{T} E[\sum_{i=1}^{N} \sum_{t=0}^{T-1} x^{\mathbb{N}_i}(t)' Q^{\mathbb{N}_i} x^{\mathbb{N}_i}(t) + u^{\mathbb{N}_i}(t)' R^{\mathbb{N}_i} u^{\mathbb{N}_i}(t)] \tag{6.125}$$

with closed loop dynamics for each neighborhood

$$x^{\mathbb{N}_i}(t+1) = A^{\mathbb{N}_i}x^{\mathbb{N}_i}(t) + B^{\mathbb{N}_i}u^{\mathbb{N}_i}(t) + (\sqrt{\epsilon}\cdot I_{nN})w(t) \quad \forall i = 1\ldots N \tag{6.126}$$

where the controller for each neighborhood is

$$u^{\mathbb{N}_i} = K^{\mathbb{N}_i}x^{\mathbb{N}_i} \tag{6.127}$$

Based on Remark 6.3.2, the cost function in equation (6.125) can be written as

$$\epsilon tr(\sum_{i=1}^{N} P^{\mathbb{N}_i}) \tag{6.128}$$

where $P^{\mathbb{N}_i}$ comes from the Riccati equation

$$P^{\mathbb{N}_i} = (A^{\mathbb{N}_i} + B^{\mathbb{N}_i}K^{\mathbb{N}_i})'P^{\mathbb{N}_i}(A^{\mathbb{N}_i} + B^{\mathbb{N}_i}K^{\mathbb{N}_i}) + (Q^{\mathbb{N}_i} + K^{\mathbb{N}_i\prime}R^{\mathbb{N}_i}K^{\mathbb{N}_i}) \tag{6.129}$$

as in equation (6.59). and $K^{\mathbb{N}_i} = (R^{\mathbb{N}_i} + B^{\mathbb{N}_i\prime}P^{\mathbb{N}_i}B^{\mathbb{N}_i})^{-1}B^{\mathbb{N}_i\prime}P^{\mathbb{N}_i}A^{\mathbb{N}_i}$ as in equation (6.58).

Cost Function 2 for Local Optimal Control (concatenated neighborhood cost function):

We next consider the cost function of the concatenation of neighborhoods with the local, neighborhood control. The cost of the collection of neighborhood systems is as given in equations (6.55) and (6.56), with controllers described by equations (6.57) (6.58) (6.59), as developed in Sections 6.3.5 and 6.3.6.

Definition 6.5.2. *The cost function of the concatenation of neighborhoods can be defined as the following when the control gain $K^{\mathbb{N}}$ as defined in equation (6.120) is employed*

$$J_{nc}^{\mathbb{N}} = \lim_{T\to\infty} \frac{1}{T} E[\sum_{t=0}^{T-1} x^{\mathbb{N}}(t)' Q^{\mathbb{N}} x^{\mathbb{N}}(t) + u^{\mathbb{N}}(t)' R^{\mathbb{N}} u^{\mathbb{N}}(t)] \tag{6.130}$$

with the closed loop system dynamics given by the block diagonal system

$$x^{\mathbb{N}}(t+1) = A^{\mathbb{N}} x^{\mathbb{N}}(t) + B^{\mathbb{N}} u^{\mathbb{N}}(t) + (\sqrt{\epsilon} \cdot I_{nN^2}) \hat{w}(t) \tag{6.131}$$

where the controller for the concatenated neighborhood system based on local optimal control is

$$u^{\mathbb{N}} = K^{\mathbb{N}} x^{\mathbb{N}}(t) \tag{6.132}$$

and where $A^{\mathbb{N}}$, $B^{\mathbb{N}}$, and $K^{\mathbb{N}}$ are in equations (6.91), (6.92), (6.120) which are all block diagonals with commensurate square blocks with the size of each block being the cardinality of each neighborhood $\mathbb{N}_i$.

Based on Remark 6.3.2, the cost function in equation (6.130) can be written as

$$\epsilon tr(P^{\mathbb{N}}) \tag{6.133}$$

where $P^{\mathbb{N}}$ comes from the Riccati equation

$$P^{\mathbb{N}} = (A^{\mathbb{N}} + B^{\mathbb{N}} K^{\mathbb{N}})' P^{\mathbb{N}} (A^{\mathbb{N}} + B^{\mathbb{N}} K^{\mathbb{N}}) + (Q^{\mathbb{N}} + K^{\mathbb{N}\prime} R^{\mathbb{N}} K^{\mathbb{N}}) \tag{6.134}$$

and where, as already explained the gain of the controller is the block diagonal matrix $K^{\mathbb{N}} = (R^{\mathbb{N}} + B^{\mathbb{N}\prime}P^{\mathbb{N}}B^{\mathbb{N}})^{-1}B^{\mathbb{N}\prime}P^{\mathbb{N}}A^{\mathbb{N}}$.

6.5.3 Bound on Cost Difference between Distributed Control and Global, Centralized Control

In this subsection, we bound the difference between the cost from the suboptimal, distributed control and the global, centralized control. We first recap the cost and controller gain for the global, centralized control as in (6.20). Towards this goal of bounding the cost difference between new distributed control and centralized control, we will look at the difference in cost between the cost function of concatenated neighborhoods with the new global distributed control as in Section 6.5.1 and the cost function of concatenated neighborhoods with the local optimal control as in Section 6.5.2.

The cost function of the system when a global, centralized control is employed is (as recapped from (6.20)).

$$J_C = \lim_{T\to\infty} \frac{1}{T} E[\sum_{t=0}^{T-1} x(t)'Qx(t) + u(t)'Ru(t)] \tag{6.135}$$

Closed loop dynamics under centralized control are as follows

$$x(t+1) = Ax(t) + Bu(t) + (\sqrt{\epsilon} \cdot I_{nN})w(t) \tag{6.136}$$

where the centralized control is

$$u(t) = K^C x(t) \tag{6.137}$$

As in Theorem (6.2.1), the cost function in equation (6.135) can be written as

$$\epsilon tr(P^C) \tag{6.138}$$

where P^C comes from the Riccati equation

$$P^C = (A + BK^C)' P^C (A + BK^C) + (Q + K^{C\prime} R K^C) \tag{6.139}$$

and

$$K^C = (R + B' P B)^{-1} B' P^C A \tag{6.140}$$

The controller K^C gives the optimal solution to the costs detailed in equations (6.135) as similarly shown previously in equation (6.20). This is also detailed in Remark 6.3.1.

We now present a lemma from [22] for finding the difference in costs between using two different controllers.

Lemma 6.5.1 (LQR Cost Difference Lemma Fazel '19 [22] - Lemma 6)**.** *The difference in costs between using gain matrices K' and K, $J(K)$ and $J(K')$ satisfies:*

$$J(K') - J(K) = -2Tr(\Sigma_{K'}(K - K')'E_K) + Tr(\Sigma_{K'}(K - K')'(R + B'P_K B)(K - K')) \tag{6.141}$$

where

$$\Sigma_{K'} = \sum_{t=0}^{\infty} x_t x_t' \tag{6.142}$$

$$x_{t+1} = (A + BK')x_t \tag{6.143}$$

$$E_K = \big((R + B'P_K B)K - B'P_K A\big). \tag{6.144}$$

$$\tag{6.145}$$

We use Lemma 6.5.1 to present the difference in cost between the distributed control and the centralized control in Lemma 6.5.2.

Lemma 6.5.2 (Cost Difference between LQR optimal centralized controller and distributed controller)**.** *Consider the centralized controller gain K^C (6.140), the associated Riccati equation solution, P^C (6.139), and the distributed controller gain K^ρ (6.75), then the cost difference between the suboptimal distributed cost, J_{DC} (6.114) (with a distributed solution) and the global optimal cost*

J_C (6.135) (with a centralized solution) is

$$J_{DC} - J_C =$$

$$-2Tr(\Sigma_{K^\rho}(K^C - K^\rho)' E_K^C) + Tr(\Sigma_{K^\rho}(K^C - K^\rho)'(R + B'P^C B)(K^C - K^\rho)) \quad (6.146)$$

where

$$\Sigma_{K^\rho} = \sum_{t=0}^{\infty} (A + BK^\rho)^t x_0 x_0' (A + BK^\rho)^{t\prime} \quad (6.147)$$

$$E_K^C = \left((R + B'P^C B)K^C - B'P^C A\right) \quad (6.148)$$

Proof. Use Lemma 6.5.1 where $K' = K^\rho$, $\Sigma_{K'} = \Sigma_{K^\rho}$, $K = K^C(6.140)$, $E_K = E_K^C$, $\Sigma_{K'} = \Sigma_{K^\rho}$, and $P^K = P^C$ (6.139). Note, we assume condition (6.4.1) holds and K^ρ stabilizes and therefore Σ_{K^ρ} is finite. □

6.6 Single Agent LEQG

In this section, we discuss the linear exponential quadratic gaussian (LEQG), which is a risk-sensitive control problem. The motivation behind LEQG is that risk is an essential aspect in many problems. Through Taylor series, it can be seen that minimizing an exponential of a cost function is the same as minimizing a sum of the different moments of the cost function where the first moment is the expectation and the second moment is the variance, and so on.

Note: as the LEQG setting is a little bit more involved than the LQG setting, for simplicity

we assume the variance of the noise element is 1, or in other words as in the LQG setting, but where $\epsilon = 1$ in equation (6.1).

6.6.1 Problem Setup

Let us consider the discrete-time dynamics equation

$$x(t+1) = Ax(t) + Bu(t) + w(t) \tag{6.149}$$

where $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$ and $w(t) \in \mathbb{R}^n$ are the state, control input, and process noise vectors, respectively. The random vector $w(t)$ is i.i.d and for each time is of the form $\mathcal{N}(0, I)$. The initial state, $x_0 \sim \mathcal{N}(\bar{x}_0, P_0)$. Additionally, x_0 and $\{w(t)\}_{t=0}^{T}$ are independent for any value of T.

The cost function associated with the infinite horizon LEQG problem is the following, J_{risk} (We use an auxiliary function C to help define J_{risk}). Define the following,

$$C = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T} \frac{1}{2}(x(t)'Qx(t) + u(t)'Ru(t)) \tag{6.150}$$

where $Q \geq 0$, $R > 0$. Then the risk sensitive cost associated with LEQG is

$$J_{risk} = \frac{1}{\gamma} log E[exp(\gamma C)] \tag{6.151}$$

with $\gamma > 0$.

LEQG seeks control $u(t)$ as a function of $x(t)$ that minimizes the cost function in equation (6.151) constrained to the dynamics in equation (6.149).

6.6.2 Solution

Theorem 6.6.1. *LEQG [66]*

Assume (A, B) is controllable, and $(A, Q^{\frac{1}{2}})$ is observable, then the optimal control to the problem in Section 6.6.1 is $u^*(t) = Kx(t)$ where

$$K = -(R + B'\tilde{P}B)^{-1}B^T\tilde{P}A \tag{6.152}$$

and where there exists a unique positive definite solution P to the generalized Riccati equation

$$P = Q + K'RK + (A + BK)^T\tilde{P}(A + BK) \tag{6.153}$$

$$\tilde{P} = P + \gamma P(I - \gamma P)^{-1}P \tag{6.154}$$

Additionally, the optimal average cost (6.151) is

$$-(1/\gamma)logdet(I - \gamma P) \tag{6.155}$$

As $\gamma \to 0$, $\tilde{P} \to P$ and $-(\frac{1}{\gamma})logdet(I - \gamma P) \to Tr(P)$. In other words, as $\gamma \to 0$, we get back the LQG setting (which is called the risk-neutral setting).

Remark 6.6.1. *For the global, centralized control setting for LEQG, we use the superscript C*

and denote $K^C_{risk} = K$ and $P^C_{risk} = P$ from Theorem 6.6.1.

Definition 6.6.1. *We define the cost function with the global, centralized control, $J_{C,risk}$ as*

$$C_C = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T} \frac{1}{2}(x(t)'Qx(t) + u(t)'Ru(t)) \tag{6.156}$$

$$J_{C,risk} := \frac{1}{\gamma} log E[exp(\gamma C_C)] \tag{6.157}$$

where $u(t) = K^C_{risk}x(t)$ and $x(t+1) = Ax(t) + Bu(t) + w(t)$.

6.7 Multi-Agent LEQG

This section is very similar to the multi-agent LQG problem in Section 6.3.

6.7.1 Agent Dynamics and Cost

The state of each agent i, $x_i(t) \in \mathbb{R}^n$ is updated as follows

$$x_i(t+1) = A_i x_i(t) + B_i u_i(t) + w_i(t) \tag{6.158}$$

for $i = 1, \ldots, N$. We next define the dynamics and cost function for agent i and its neighbors.

6.7.2 Neighborhood Agents Dynamics and Cost

As in the multi-agent LQG example, the neighborhood dynamics are

$$x^{\mathbb{N}_i}(t+1) = A^{\mathbb{N}_i} x^{\mathbb{N}_i}(t) + B^{\mathbb{N}_i} u^{\mathbb{N}_i}_{risk}(t) + w^{\mathbb{N}_i}(t) \tag{6.159}$$

The local, neighborhood Riccati equations are

$$P_{risk}^{\mathbb{N}_i} = Q^{\mathbb{N}_i} + K_{risk}^{\mathbb{N}_i}{}' R^{\mathbb{N}_i} K_{risk}^{\mathbb{N}_i}{}' + (A^{\mathbb{N}_i} + B^{\mathbb{N}_i} K_{risk}^{\mathbb{N}_i})' \tilde{P}_{risk}^{\mathbb{N}_i} (A^{\mathbb{N}_i} + B^{\mathbb{N}_i} K_{risk}^{\mathbb{N}_i}) \tag{6.160}$$

$$\tilde{P}_{risk}^{\mathbb{N}_i} = P_{risk}^{\mathbb{N}_i} + \gamma P_{risk}^{\mathbb{N}_i} (I - \gamma P_{risk}^{\mathbb{N}_i})^{-1} P_{risk}^{\mathbb{N}_i} \tag{6.161}$$

$$K_{risk}^{\mathbb{N}_i} = -(R^{\mathbb{N}_i} + B^{\mathbb{N}_i}{}' \tilde{P}_{risk}^{\mathbb{N}_i} B^{\mathbb{N}_i})^{-1} B^{\mathbb{N}_i}{}' \tilde{P}_{risk}^{\mathbb{N}_i} A^{\mathbb{N}_i} \tag{6.162}$$

where

$$u_{risk}^{\mathbb{N}_i} = K_{risk}^{\mathbb{N}_i} x^{\mathbb{N}_i} \tag{6.163}$$

We next define the corresponding cost function.

Definition 6.7.1. *We define the cost function, $J_{risk}^{\mathbb{N}_i}$ for the cost of neighborhood i with the neighborhood control defined in (6.163).*

$$C^{\mathbb{N}_i} = \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T} \frac{1}{2} [(x^{\mathbb{N}_i}(t)' Q^{\mathbb{N}_i} x^{\mathbb{N}_i}(t) + u_{risk}^{\mathbb{N}_i}(t)' R^{\mathbb{N}_i} u_{risk}^{\mathbb{N}_i}(t))] \tag{6.164}$$

$$J_{risk}^{\mathbb{N}_i} := \frac{1}{\gamma} log E[exp(\gamma C^{\mathbb{N}_i})] \tag{6.165}$$

6.7.3 Distributed Control

The multi-agent system control is similar to the control for a single agent LEQG. Additionally, the distributed control is similar to the proposed new global distributed control for the multi-

agent LQG problem (See Algorithm 5 and then the modifications in Remark 6.4.1 which is then implemented in Definition 6.4.1 so that the controller can be written as a linear feedback controller) where the parameter for the distributed control, K^{ρ}_{risk} is

$$K^{\rho}_{risk} = \sum_{i=1}^{N} \rho^{\mathbb{N}_i} K^{\mathbb{N}_i}_{risk} \tag{6.166}$$

(as similarly defined in the LQG setting in Definition 6.4.1) with $K^{\mathbb{N}_i}_{risk}$ defined in equation (6.162) and where

$$[\rho^{\mathbb{N}_i}]_{j,j} = \begin{cases} \frac{1}{|\mathcal{N}_j|}, & \text{if} \quad |\mathcal{N}_j| \neq 0 \\ 0, & \text{if} \quad \text{j and i are not connected} \end{cases} \qquad for j = 1, \dots N. \tag{6.167}$$

Note from (6.167), $\sum_{i=1}^{N} \rho^{\mathbb{N}_i} = I_N$.

Definition 6.7.2. *Define the distributed control,*

$$u_{DC,risk}(t) = K^{\rho}_{risk} x(t) \tag{6.168}$$

Definition 6.7.3. *We define the cost of the system when the distributed control is used.*

$$C_{DC} = \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T} \frac{1}{2} (x(t)' Q x(t) + u_{DC,risk}(t)' R u_{DC,risk}(t)) \tag{6.169}$$

$$J_{DC,risk} := \frac{1}{\gamma} log E[exp(\gamma C_{DC})] \tag{6.170}$$

The dynamics are $x(t+1) = Ax(t) + Bu(t) + w(t)$*. This gives us an equation similar to (6.78)*

$$x^D(t+1) = Ax^D(t) + B\sum_{i=1}^{N} \rho^{\mathbb{N}_i} K_{risk}^{\mathbb{N}_i} x^D(t) + w(t) \tag{6.171}$$

with the only difference that $K_{risk}^{\mathbb{N}_i}$ *is defined in equation (6.162) and* $\epsilon = 1$*.* $K_{risk}^{\rho} = \sum_{i=1}^{N} \rho^{\mathbb{N}_i} K_{risk}^{\mathbb{N}_i}$ *is of size* $mN \times nN$*,* $\rho^{\mathbb{N}_i}$ *is of size* $mN \times mN$*, and* $K_{risk}^{\mathbb{N}_i}$ *is of size* $mN \times nN$*.*

We next present the discrete-time bounded real lemma which shows an equivalence between the H_∞ bound and a generalized Riccati equation.

Lemma 6.7.1 (Discrete-Time Bounded Real Lemma)**.** *Consider* $\|T(K)\|_\infty$ *which equals* $\frac{\sum_{t=0}^{\infty} \frac{1}{2}[x_t' Q x_t + u_t' R u_t]}{\sum_{t=0}^{\infty} w_t' w_t}$ *where* $u_t = Kx_t$ *for a given control gain matrix* K *and the state* x_t *updated according to (6.149). Suppose* K *is stabilizing, i.e.* $\|(A+BK)\| < 1$ *then the following conditions are equivalent*

- $\|T(K)\|_\infty < \gamma$
- *there exists some* $P > 0$ *such that* $(I - \gamma P) > 0$ *and*

$$(A+BK)'\tilde{P}(A+BK) + Q + K'RK - P < 0 \tag{6.172}$$

$$\tilde{P} = P + \gamma P(I - \gamma P)^{-1} P \tag{6.173}$$

Remark 6.7.1. *If one of the two conditions in Lemma 6.7.1 hold then*

$$\|(I - \gamma P)'^{-1}(A+BK)\| \leq 1 \tag{6.174}$$

Proof. See Lemma 2.7 in [5]. □

Lemma 6.7.2. *Assuming the conditions in assumption 6.4.1 and the set of matrices* $\{\rho^{\mathbb{N}_i}\}_{i=1}^{N}$ *and* $\{K_{risk}^{\mathbb{N}_i}\}_{i=1}^{N}$ *are such as in the distributed controller in (6.166), leading to* $K_{risk}^{\rho} = \sum_{i=1}^{N} \rho^{\mathbb{N}_i} K_{risk}^{\mathbb{N}_i}$ *being stabilizing. That is* $\|A + BK_{risk}^{\rho}\| \leq 1$. *Then,* $\|T(K_{risk}^{\rho})\| < \gamma$ *if the following two conditions hold:*

Condition 1*: there exists some* $P_{DC,risk} > 0$ *such that the Riccati inequality holds*

$$(A + BK_{risk}^{\rho})' \tilde{P}_{DC,risk} (A + BK_{risk}^{\rho}) + Q + K_{risk}^{\rho}{}' R K_{risk}^{\rho} - P_{DC,risk} < 0 \tag{6.175}$$

$$\tilde{P}_{DC,risk} = P_{DC,risk} + \gamma P_{DC,risk} (I - \gamma P_{DC,risk})^{-1} P_{DC,risk} \tag{6.176}$$

Condition 2*:* $P_{DC,risk}$ *is such that*

$$(I - \gamma P_{DC,risk}) > 0 \tag{6.177}$$

Moreover, these two conditions can be written as the following linear matrix inequality condition. There exists a $P_{DC,risk} > 0$ *such that*

$$\begin{bmatrix} (A + BK_{risk}^{\rho})' P_{DC,risk} (A + BK_{risk}^{\rho}) - P_{DC,risk} + Q + K_{risk}^{\rho}{}' R K_{risk}^{\rho} & (A + BK_{risk}^{\rho})' P_{DC,risk} \\ P_{DC,risk} (A + BK_{risk}^{\rho}) & -(I - \gamma P_{DC,risk}) \end{bmatrix} < 0 \tag{6.178}$$

Proof. Consider Lemma 6.7.1, the discrete-time bounded real lemma and let $K = K^{\rho}_{risk}$ and $P = P_{DC,risk}$. Now, by Schur complement we can represent the two conditions that are equivalent to $\|T(K^{\rho}_{risk})\|_{\infty} < \gamma$ as the linear matrix inequality (6.178). □

Remark 6.7.2. *If the two conditions don't hold we can always raise the value of γ so that $\|T(K)\|_{\infty} < \gamma$ holds for a larger γ. After all, when γ approaches ∞, we get back the LQG setting that we have dealt with previously .*

As in the Lemma in [22], we repeat Lemma 5.1 in [50] here. It is the counterpart to the Cost Difference Lemma in Fazel et al. (2018). [50] established an upper bound for the difference of two matrices P' and P.

Lemma 6.7.3. *(LEQG Cost Difference Lemma Zhang '19 [5] - Lemma 5.1) The cost difference between two costs, one with a controller gain K' and the other with controller gain K is*

$$P' - P$$

$$\leq \sum_{t\geq 0}[(A+BK')'(I-\gamma P')^{-1}]'[-(K-K')'E_k - E_k'(K-K')+ \tag{6.179}$$

$$(K-K')'(R+B'\tilde{P}B)(K-K')][(I-\gamma P')^{-1\prime}(A+BK')]' \tag{6.180}$$

where $E_K = (R+B'\tilde{P}B)K - B'\tilde{P}A$.

Lemma 6.7.4. *(Cost Difference between LEQG distributed controller and optimal centralized controller) The cost difference between the cost with the LEQG distributed controller gain (K^{ρ}_{risk}) (defined in Lemma 6.7.2) and the cost with the optimal centralized controller (K^{C}_{risk}) (defined in*

equation (6.152) is

$$P_{DC,risk} - P^C_{risk} \tag{6.181}$$

$$\leq \sum_{t\geq 0}[(A+BK^\rho_{risk})'(I-\gamma P_{DC,risk})^{-1}]'[-(K^C_{risk}-K^\rho_{risk})'E^C_K - E^{C\prime}_K(K^C_{risk}-K^\rho_{risk}) + \tag{6.182}$$

$$(K^C_{risk}-K^\rho_{risk})'(R+B'\tilde{P}^C_{risk}B)(K^C_{risk}-K^\rho_{risk})][(I-\gamma P_{DC,risk})^{-1\prime}(A+BK^\rho_{risk})]' \tag{6.183}$$

where $E^C_K = (R+B'\tilde{P}^C_{risk}B)K^C_{risk} - B'\tilde{P}^C_{risk}A$.

Proof. Using Lemma 6.7.3, set $P_{DC,risk} = P'$, $P^C = P$, $K^\rho_{risk} = K'$, $E^C_K = E_K$, $K^C_{risk} = K$.

$P_{DC,risk}$ and K^ρ_{risk} are defined in Lemma 6.7.2 and P^C_{risk} and K^C_{risk} are defined in Lemma 6.6.1.

□

Remark 6.7.3. *Denote the LEQG cost (6.156) using the distributed controller (*6.166*) as* $J_{DC,risk}$ *and the LEQG cost (6.156) using the centralized controller (6.152) as* $J_{C,risk}$.

Lemma 6.7.5. *If* $P'_K \geq P_K$, *then the cost difference is*

$$J(K') - J(K) \leq \|I-\gamma P)^{-1}\|[Tr(P_K) - Tr(P'_K)] \tag{6.184}$$

Proof. See Lemma 5.2 in [5]. □

Lemma 6.7.6. *The cost difference between the LEQG distributed controller and global, centralized*

controller, $J_{DC,risk} - J_{C,risk}$ *is upper bounded by*

$$J_{DC,risk} - J_{C,risk} \leq \|I - \gamma P_{DC,risk})^{-1}\| \big[Tr \big(\sum_{t \geq 0} [(A + BK^{\rho}_{risk})'(I - \gamma P_{DC,risk})^{-1}]' \times$$

$$[-(K^{C}_{risk} - K^{\rho}_{risk})' E^{C}_{K} - E^{C\prime}_{K}(K^{C}_{risk} - K^{\rho}_{risk}) + (K^{C}_{risk} - K^{\rho}_{risk})'(R + B' \tilde{P}^{C}_{risk} B)(K^{C}_{risk} - K^{\rho}_{risk})] \times$$

$$[(I - \gamma P_{DC,risk})^{-1\prime}(A + BK^{\rho}_{risk})]' \big) \big] \tag{6.185}$$

where $E^{C}_{K} = (R + B' \tilde{P}^{C}_{risk} B) K^{C}_{risk} - B' \tilde{P}^{C}_{risk} A$.

Proof. Substitute Lemma 6.7.4 into Lemma 6.7.5.

Note, $J_{DC,risk}$ is defined in Definition 6.7.3 and $J_{C,risk}$ is defined in Definition 6.6.1. $P_{DC,risk}$ is defined in Lemma 6.7.2. P^{C}_{risk} and K^{C}_{risk} are defined in Definition 6.6.1. □

6.8 Simulation

We consider the following discrete time version of the double integrator problem where a 10×10 mesh interconnection of $N = 100$ identical, dynamically decoupled agents move in a plane. Consider, the double integrator dynamics in spatial dimensions.

$$\ddot{x}_i = u_{x,i} \quad \ddot{y}_i = u_{y,i} \quad i = 1, \ldots, 100 \tag{6.186}$$

The goal is to move each to a specified location. The optimal control problem associated with this example is

$$\min_{\tilde{u}} J(u, \tilde{x_0}) \tag{6.187}$$

$$\tilde{x}_{k+1} = \tilde{A}\tilde{x}_k + \tilde{B}u_k + \sqrt{\tilde{\epsilon}}\tilde{w}_t \tag{6.188}$$

where

$$J(u, \tilde{x_0}) = \sum_{k=0}^{T} \tilde{x}_k' Q \tilde{x}_k + u_k' R u_k \tag{6.189}$$

Q is made of block matrices:

$$Q_{ii} = diag([5; 0; 5; 0]) \tag{6.190}$$

$$Q_{ij} = diag([-1; 0; -1; 0]); \tag{6.191}$$

R is the identity matrix.A, B, and D matrices are equal to

$$A = \begin{bmatrix} 1 & t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & t \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.192}$$

$$B = \begin{bmatrix} \frac{t^2}{2} & 0 \\ t & 0 \\ 0 & \frac{t^2}{2} \\ 0 & t \end{bmatrix} \tag{6.193}$$

$$D = \begin{bmatrix} \sqrt{\epsilon} & 0 & 0 & 0 \\ 0 & \sqrt{\epsilon} & 0 & 0 \\ 0 & 0 & \sqrt{\epsilon} & 0 \\ 0 & 0 & 0 & \sqrt{\epsilon} \end{bmatrix} \tag{6.194}$$

t is the step size, $\tilde{A} = I_{100} \otimes A$, $\tilde{B} = I_{100} \otimes B$, and $\tilde{D} = I_{100} \otimes D$. Let the state of agent i, $\tilde{x_i} \in \mathbb{R}^4 = [x_i, y_i, v_{xi}, v_{y_i}]$ where x_i is the x^{th} position of agent i, y_i is the y^{th} position of agent i, and v_{xi} and v_{y_i} are the respective velocities. Tildes above a variable denotes the concatenation or block diagonal form for the variables for all N agents. For example, $\tilde{x}_0$ is the concatenation of the N initial states.

We solve for 4 different K control parameters corresponding to 4 different settings. The settings are as follows: global control for LQR (K), distributed control for LQR (K-dist), global control for LEQG (K-risk), and distributed control for LEQG (K-riskdist). Table 6.1 details the cost (averaged over 30 runs) for each of the four settings and the time it took to compute the K matrices. The time horizon is $T = 100$. We chose a time horizon long enough so we can solidly rely on the steady state assumption. In practice, this steady state assumption is not met, but we

	cost	time to compute K
K	13.97	14.71 sec.
K-dist	13.91	1.412 sec.
K-risk	27.25	636.61 sec.
K-riskdist	26.05	11.07 sec.

Table 6.1: Cost and time of different gain matrices

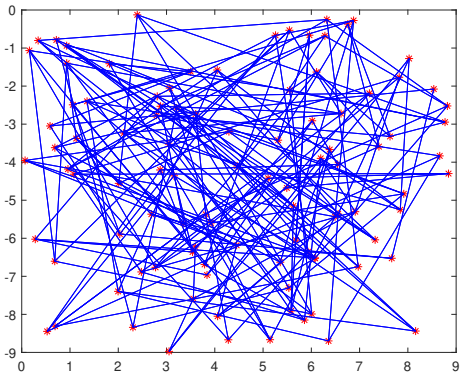

Figure 6.3: The plot is the initial condition of the agents .

rely on it nonetheless because of the length of our time horizon. We see from the table that not only is a distributed controller useful in respecting communication restrictions, but it is also useful in reducing computational cost by distributing the computation into smaller problems. In this way, a large scale linear algebra problem does not have to be solved.

We also show in the figures the performance of the controller in reorganizing the agents to the zeroth state

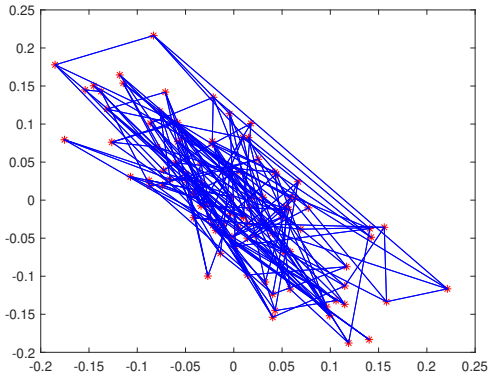

Figure 6.4: The plot is the final condition (T=100) of the agents with distributed LQR control.

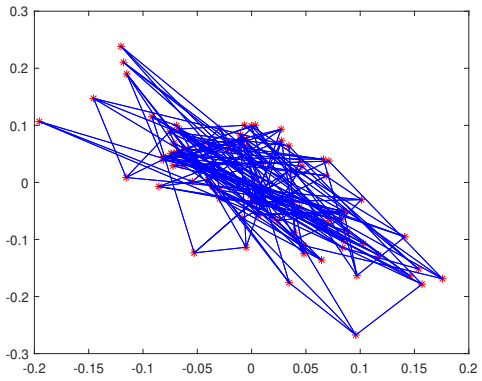

Figure 6.5: The plot is the final condition (T=100) of the agents with distributed LEQG control.

6.9 Conclusion

In this chapter, we have proposed a distributed control algorithm for the LQR and LEQG (risk-sensitive) setting. We analyzed the performance of this distributed algorithm compared to the global control. We showed in LQR that if the R matrix is diagonal, then the cost function associated with the distributed control and the cost function associated with the global control are the same. The LEQG setting is more complicated to deal with mathematically. Our bounds on the performance for the distributed controller for LEQG is not as tight, but does provide some insight on how much worse off one performs by using a distributed LEQG controller versus a global controller.

In the future, we will consider improved upper bounds for the distributed LEQG and better understanding of what the current bounds mean. Additionally, there are other approaches to distributed control that we can consider that include distributed optimization ([73], [74], [75]).

Part III: Reinforcement Learning for Risk-sensitive Control

Chapter 7: Risk-Sensitive Control for Unknown Dynamics

In the previous chapters, we addressed the problem of communications constraints in the setting of estimation and control. Here, we discuss another limitation in the control setting. That is the limitation of not always knowing the dynamics of the system you intend to control. In this third and final part of the thesis, described in this chapter, we discuss a problem in the area of reinforcement learning for risk-sensitive controller (the H_∞ controller, which will be highlighted later as a special case of a risk-sensitive controller). The problem deals with designing a risk-sensitive controller when the dynamics are unknown in the linear setting. The problem of designing H_∞ controllers using policy optimization has been studied in the past by Zhang et al. [5]. However, that paper only has a limited discussion on the case when the dynamics are unknown.

7.1 Introduction and Literature Review

The question presented here is how can we design a risk-sensitive controller when the dynamics are unknown but the structure is known to be linear and the dimensions of the matrices are also known. We take two approaches to this question. The first is a policy gradient (model-free) approach based on the one proposed in [5]. In that paper, an H_∞ controller is used, while here we use a risk-sensitive controller. We discuss this approach in Section 7.3.

The second approach is to use a model-based approach. System identification is used to estimate the unknown matrices in the dynamical equations. A risk-sensitive controller is then computed based on these estimates. This is similar to the approach proposed in [6] where the aim is to compute a linear quadratic regulator (LQR). Here, the aim is to compute a risk-sensitive controller, not LQR. Additionally, in [6], a technique called System Level Synthesis is used to obtain a robust control design. Robust in this context means that the controller is robust to faulty system-identification estimates of the dynamical system's matrices. Here, however, we use a linear matrix inequality approach ([21], [76]) that takes into account uncertainty in the estimates of the unknown dynamics in order to obtain a controller. We deal with both polytopic and interval uncertainty. This is discussed in Section 7.4. Information on particular uncertainty sets is discussed in Section 7.5.

Finally, we showcase the two approaches on two examples. The first is a random linear dynamics example. The second is a linear vehicle dynamics example. We compare the performance of both the model-free and model-based approaches and show the improved performance of incorporating uncertainty sets in the model-based approach. This is all discussed in section 7.6.

7.2 Problem Formulation

Consider the linear time invariant dynamics

$$x_{t+1} = Ax_t + Bu_t + w_t \tag{7.1}$$

where $x(t), w(t) \in \mathbb{R}^n, u_t \in \mathbb{R}^m$, and $w_t \sim \mathcal{N}(O, W)$.

Let the cost function incorporate an exponential function

$$C^{\mu}(u) = \lim_{T\to\infty} \frac{1}{T} E[exp\big(\mu\{\sum_{t=0}^{T} \frac{1}{2} x_t' Q x_t + \frac{1}{2} u_t' R u_t\big)] \tag{7.2}$$

The goal is to minimize the cost function in equation (7.2) as follows

$$\min_{u} C^{\mu}(u) \tag{7.3}$$

Assuming the conditions in assumption 6.4.1, it can be shown that the optimal controller is a linear function of the state such that $u_t = Kx_t$ ([66]). And therefore, the minimization occurs over matrices K and not functions u. We are left with the following problem

$$\min_{K} C^{\mu}(K) = \lim_{T\to\infty} \frac{1}{T} E[exp\big(\mu\{\sum_{t=0}^{T} \frac{1}{2} x_t' Q x_t + \frac{1}{2} x_t' K' R K x_t\big)] \tag{7.4}$$

If the dynamics matrices (A, B) are known we can analytically solve for the optimal gain K^*.

$$K^* = -(R + B'\tilde{P}B)^{-1}B'\tilde{P}A \tag{7.5}$$

with the accompanying modified Riccati equations ([66])

$$\tilde{P} = P + \mu P(W^{-1} - \mu P)^{-1}P \tag{7.6}$$

$$P = Q + K'RK + (A + BK)'\tilde{P}(A + BK) \tag{7.7}$$

Problem 7.2.1. *Consider minimizing the cost function in equation (7.4) when matrices (A, B) in equation (7.1) are unknown.*

We detail two approaches to this problem: model-free and model-based. The first approach can be found in ([5]). They detail their algorithm for a min-max robust control problem. We utilize their algorithm for a risk-sensitive control problem and change their simulated gradient from a one-sided gradient to a two-sided gradient as detailed in ([77]). The second approach, the model-based approach can be found in ([6]). There, a System Level Synthesis (SLS) framework is used to solve for an LQG controller given the uncertainty in the estimator ([78]). Here, we use a linear matrix inequality approach to incorporate the uncertainty set of the estimator to solve for both a LQG and LEQG controller.

Remark 7.2.1. *Note, owing to the exponential in the cost function of the LEQG, there is concern of blow up. In fact, when we implement the model-free and model-based algorithms on the examples in Section 7.6, we use smaller weighting matrices Q and R to avoid this issue.*

7.3 Model-free Approach for Risk-Sensitive Controller Design

In this section, we present the same policy optimization as in [5] now applied to risk-sensitive control. The reason for using a risk-sensitive controller instead of an H_∞ controller as in [5] is two-fold. Firstly, it has previously been shown that the zero-sum LQ game formulation of the H_∞ controller is a limiting case of the risk-sensitive control problem. Also, the controllers for the H_∞ and risk sensitive control problem are the same [67]. Secondly, the risk-sensitive control problem requires only a minimization in its objective function while the H_∞ control problem requires a saddle point solution to a min-max problem. The particular risk-sensitive control problem here is one in which the dynamics are linear and the cost is an exponential of a quadratic function. It is referred to as the linear exponential Gaussian quadratic control problem (LEQG) ([66]) in contrast to the more commonly known linear quadratic Gaussian control problem (LQG) ([71]).

7.3.1 Model-Free Policy Gradients

The model-free approach for the risk-sensitive control problem (with unknown dynamics) as in the case of the H_∞ control problem is to use zeroth order optimization ([79]) to estimate the cost function's gradient by using query access to the cost function. We highlight here the algorithm that uses the zeroth order optimization or sample derivative approach. Algorithm 6 is the same as the algorithm that appears in [5] where a model-free solution is found for the mixed $H_2 - H_\infty$ controller. The only difference is here our cost function is a risk sensitive cost function. Additionally. we use a two-sided gradient estimator instead of a one-sided gradient estimator in algorithm 7. In algorithm 8, we show the gradient update.

Algorithm 6 (One-point Gradient Estimation) Estimating $\nabla_K C(K)$ where C(K) is the risk-sensitive cost

Input: K, number of trajectories m, roll out length l, smoothing parameter r, dimension d

for i=1,. . . , m do

Sample a policy parameterized by a gain matrix such that $\tilde{K}_i = K + U_i$ where U_i is drawn uniformly at random over matrices whose Frobenius norm is r. Simulate the dynamics for T steps according to the equations

$$x_{t+1} = Ax_t + Bu_t + w_t$$

and collect the estimates $\hat{C}_i$ where the cost is the risk sensitive cost

$$\hat{C}_i = \sum_{t=1}^{T} c_t$$

where c_t and x_t are the costs and states on this trajectory.

end

Return the biased estimates

$$\hat{\nabla}_K C(K) = \frac{1}{m} \sum_{i=1}^{m} \frac{d}{r^2} \hat{C}_i U_i$$

Algorithm 7 (Two-point Gradient Estimation) Estimating $\nabla_K C(K)$ where C(K) is the risk-sensitive cost

Input: K, number of trajectories m, roll out length l, smoothing parameter r, dimension d

for i = 1,. . . , m do

Sample a policy parameterized by a gain matrix such that $\tilde{K}_i^F = K + U_i$ where U_i is drawn uniformly at random over matrices whose Frobenius norm is r. We refer to this as the forward policy. Additionally, sample a policy parametrized by a gain matrix such that $\tilde{K}_i^B = K - U_i$. We refer to this as the backward policy. Simulate the dynamics for T steps according to the equations

$$x_{t+1} = Ax_t + Bu_t + w_t$$

and collect the estimates $\hat{C}_i^F$ and $\hat{C}_i^B$ where the cost is the risk sensitive cost and the gain matrices are $\tilde{K}_i^F$ and $\tilde{K}_i^B$ respectively.

$$\hat{C}_i^F = \sum_{t=1}^{T} c_t^F$$

$$\hat{C}_i^B = \sum_{t=1}^{T} c_t^B$$

where c_t^F and c_t^B are the costs on the trajectory.

end

Return the biased estimates

$$\hat{\nabla}_K C(K) = \frac{1}{m} \sum_{i=1}^{m} \frac{d}{2r^2} [\hat{C_i^F} U_i - \hat{C_i^B} U_i]$$

Algorithm 8 Model free risk sensitive control

Input: stabilizing K, number of iterations T for $\tau = 0, \dots T-1$ do
Call Algorithm 7 to obtain the gradient

$$\hat{\nabla}_K C(K)$$

Policy gradient update

$$K_{t+1} = K_t - \eta \hat{\nabla}_K C(K)$$

end

Return K_T

7.4 Model-Based Approach for Risk-Sensitive Controller Design

Consider the same problem as in 7.2.1 where matrices (A, B) are unknown.

7.4.1 LMI solution to LEQG When Matrices are Known

We detail the solution to problem 7.2.1 in the LEQG setting first for the case when the matrices (A,B) are actually known.

Remark 7.4.1. *Consider (A, B) is controllable, $(A, Q^{\frac{1}{2}})$ is observable, then the optimal control gain where $u^* = Kx(t)$ is equal to*

$$K^* = -(R + B'\tilde{P}B)^{-1}\tilde{P}A \tag{7.8}$$

where the following generalized Riccati equations associated with LEQG (considering $W^{-1} - \gamma P > 0$) hold [30]

$$P = Q + A'[\tilde{P} - \tilde{P}B(R + B'\tilde{P}B)^{-1}B'\tilde{P}]A \tag{7.9}$$

$$\tilde{P} = P + \mu P(W^{-1} - \mu P)^{-1}P \tag{7.10}$$

Remark 7.4.2. *The LMI for LEQG controller design when the matrices A and B are known is*

[21]

$$\max_{F,P} \mu \begin{bmatrix} P & AP - BF & I & 0 \\ PA' - F'B' & P & 0 & PC' - F'D' \\ I & 0 & \frac{1}{\sqrt{\mu}} I & 0 \\ 0 & CP - DF & I & \frac{1}{\sqrt{\mu}} I \end{bmatrix} > 0 \tag{7.11}$$

where $K = FP^{-1}$.

Now, we will consider the case when the matrices (A, B) are unknown.

7.4.2 LMI Solution When Matrices are Unknown

In addition to the model-free policy gradient approach to the single agent, fully observable LEQG problem, we detail here a model-based approach. A model-based approach that solves the LQG problem when the dynamics are unknown is detailed in [6]. In that paper, system identification (least-squares) is used to solve for the unknown matrices in the dynamics equations (i.e. A and B) Uncertainty in the estimates of these matrices is quantified and then an optimal controller is solved for incorporating these uncertainty estimates. In other words, instead of just using the estimates for the matrices to compute the optimal LQG controller, they use information about how uncertain these estimates are in order to compute a robust optimal controller. The technique they use is called System Level Synthesis. Here, we do something similar. However, our setting is LEQG instead of LQG. Additionally, we use a linear matrix inequality (LMI) approach to compute the robust optimal controller.

Note, the following helpful definition.

Definition 7.4.1 ([44]). *The convex hull of a set C, denoted $convC$, is the set of all convex combinations of points in C or in other words*

$$convC = \{\theta_1 x_1 + \ + \theta_k x_k | x_i \in C, \quad \theta_i \geq 0, \quad i = 1, \ldots, k, \quad \theta_1 + \cdots + \theta_k = 1\}. \tag{7.12}$$

The model-based approach has two parts to it summarized in Remark 7.4.3 and then further elaborated in the remainder of the section.

Remark 7.4.3. *The following two step approach showcases an LMI model-based approach*

1. *Rollout dynamics for T time steps. Estimate A and B ($\hat{A}$ and $\hat{B}$) by solving least squares problem. Find error perturbations or uncertainties around estimates, $\hat{A}$, $\hat{B}$, namely $\triangle\hat{A}$ and $\triangle\hat{B}$.*

2. *Solve an LMI that does two things: 1. solves LEQG control problem with estimates of A and B. 2. incorporates error perturbations into LMI as interval uncertainties such as $\triangle A(t) = A_1\delta_1(t) + \cdots A_k\delta_k(t)$ and $B(t) = B_1\delta_1(t) + \cdots B_k\delta_k(t)$ or polytopic uncertainties such as $\triangle A(t) = conv(A_1(t), \ldots, A_{2k}(t))$ and $\triangle B(t) = conv(B_1(t), \ldots, B_{2k}(t))$ where conv denotes convex hull as in Definition 7.4.1.*

Remark 7.4.4 (**Estimating matrices A and B when unknown**). *Matrices A and B are estimated by system identification through least-squares [6]. The dynamics are rolled-out for T time steps with a given initial condition x_0 and a given input u. The states x_t ($0 < t < T$) are then collected.*

Estimate A and B through the following least-squares estimation.

$$(\hat{A}, \hat{B}) \in_{(A,B)} \sum_{l=1}^{N} \sum_{t=0}^{T-1} \frac{1}{2} \|Ax_t^{(l)} + Bu_t^{(l)} - x_{t+1}^{(l)}\|_2^2 \tag{7.13}$$

Let $\theta = [A, B]'$ and let z_t:= $\begin{bmatrix} x_t \\ \\ u_t \end{bmatrix}$. The system dynamics can then be written as

$$x'_{t+1} = z'_t\theta + w'_t \tag{7.14}$$

Or in other words,

$$X_N = Z_N\theta + W_N \tag{7.15}$$

where

$$X := \begin{bmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_T \end{bmatrix}, Z := \begin{bmatrix} z'_0 \\ z'_1 \\ \vdots \\ z'_{T-1} \end{bmatrix}, W := \begin{bmatrix} w'_0 \\ w'_1 \\ \vdots \\ w'_{T-1} \end{bmatrix} \tag{7.16}$$

The least squares estimator for θ is

$$\hat{\theta} = (Z'_N Z_N)^{-1} Z'_N X_N \tag{7.17}$$

Now, we discuss the different uncertainty sets in that we use to characterize the uncertainty

in the least squares estimator

Definition 7.4.2 (**Interval Uncertainty**). *We parameterize the uncertainty in the matrices A and B as the sum of matrices, i.e.*

$$\triangle A = A_1\delta_1 + \ldots, A_k\delta_k \tag{7.18}$$

and

$$\triangle B = B_1\delta_1 + \ldots, B_k\delta_k \tag{7.19}$$

where δ lies in the intervals such that $\delta_j \in [\delta_i^-, \delta_i^+]$. And the matrix A_j such that the j^{th} element in the matrix is equal to one and the rest are zeros.

$$A_j = \begin{bmatrix} 0 & \ldots & 0 \\ \vdots & 1 & \vdots \\ 0 & \ldots & 0 \end{bmatrix} \tag{7.20}$$

Remark 7.4.5. *Repeat the least squares estimation M times, where each of the M estimates of the matrices A and B are $\hat{A}_m$ and $\hat{B}_m$ for $m = 1 \ldots M$. Now consider solving for δ_j^- and δ_j^+ for $j = 1 \ldots k$ and $k = n^2$.*

$$\delta_j^- = \min\{[\hat{A}_m]_j | m = 1 \ldots M\} \tag{7.21}$$

and

$$\delta_j^+ = \max\{[\hat{A}_m]_j | m = 1 \dots M\} \tag{7.22}$$

where j refers to the j^{th} location in the matrix.

The uncertainty set is

$$V := \sum_i A_i \delta_i, \tag{7.23}$$

where $\delta_i \in \{\delta^-, \delta^+\}$

Definition 7.4.3 (**Polytopic Uncertainty**). *We parameterize the uncertainty in the matrices A and B as a convex hull of multiple matrices. For example, the uncertainty in the matrix A can be written as $\triangle A = Co(A_1, \dots, A_k, \dots, A_{2k})$ where $k = n^2$ where n is the dimension of the system's states.*

The matrices $\{A_j\}_{j=1}^{2k}$ of which a convex hull is taken are defined as the $n \times n$ matrices that are zero everywhere except at position j or $j - n^2$ (if $j > n^2$).

$$A_j = \begin{bmatrix} 0 & \dots & 0 \\ \vdots & \delta_j^- & \vdots \\ 0 & \dots & 0 \end{bmatrix} \tag{7.24}$$

and

$$A_{2j} = \begin{bmatrix} 0 & \dots & 0 \\ \vdots & \delta_j^+ & \vdots \\ 0 & \dots & 0 \end{bmatrix} \tag{7.25}$$

where the placement for the non-zero element for A_j and A_{2j} is in position j of the matrix (indexed top to bottom).

Remark 7.4.6. *The values δ_j^- and δ_j^+ are obtained in the following way. The dynamics are rolled out for M episodes, each for T time steps. In each episode, least squares is performed and estimates $\hat{A}_m$ and $\hat{B}_m$ for $m = 1, \dots, M$ are obtained. Then, δ_j^- and δ_j^+ are computed for $j = 1 \dots n^2$ in the following manner*

$$\delta_j^- = \min\{[\hat{A}_m]_j | m = 1 \dots M\} \tag{7.26}$$

and

$$\delta_j^+ = \max\{[\hat{A}_m]_j | m = 1 \dots M\} \tag{7.27}$$

where j refers to the j^{th} location in the matrix. Similarly, the same computation can be applied for B_j and B_{2j}.

We now detail a well-known definition and accompanying theorems that will help inform us what the proper linear matrix inequality that incorporates uncertainties in the estimates A and B. The setting for the definition and theorems is continuous time, but could equally be applied

to discrete time.

Definition 7.4.4 (**Quadratic Stability**). *Consider the continuous time system*

$$\dot{x}(t) = (A_0 + \triangle A)x(t)$$

The system is quadratically stable over for all $\triangle A \in \triangle$ if there exists a $P > 0$ s.t.

$$(A + \triangle A)'P + P(A + \triangle A) < 0 \tag{7.28}$$

for all $\triangle A \in \triangle$.

Theorem 7.4.1 (**Quadratic Stability for Interval Uncertainty**). *Consider the interval uncertainty set*

$$\triangle A := \sum_i A_i \delta_i, \tag{7.29}$$

where $\delta_i \in \{\delta^-, \delta^+\}$ for $i = 1, \ldots, k$.

The system is quadratically stable over $\triangle A$ if and only if there exists a $P > 0$ such that

$$\sum_i A_i \delta_i' P + P \sum_i A_i \delta_i < 0 \tag{7.30}$$

for every $\delta_i \in \{\delta^-, \delta^+\}$ for $i = 1, \ldots, k$. There are 2^k LMI constraints.

Theorem 7.4.2 (**Quadratic Stability for Polytopic Uncertainty**)). *Consider the polytopic uncertainty set* $\triangle A = Co(A_1, \ldots, A_k, \ldots, A_{2k})$.

The system is quadratically stable over $\triangle A$ *if and only if there exists a* $P > 0$ *such that*

$$(A + A_i)'P + P(A + A_i) < 0 \tag{7.31}$$

for $i = 1, \ldots, 2k$.

Theorem 7.4.3 (**Quadratic Stability for Polytopic Uncertainty with Control**). *Quadratic Stability for Polytopic Uncertainty can be extended to the system*

$$\dot{x}(t) = (A + \triangle A + (B + \triangle B)K)x(t) \tag{7.32}$$

so that there exists a K *such that the system is quadratically stable for* $(\triangle A, \triangle B) \in Co((A_1, B_1), \ldots, (A_k, B_k))$. *This is true if and only if there exists some* $P > 0$ *and* F *s.t.*

$$(A + A_i)P + P(A + A_i)' + (B + B_i)F + F'(B + B_i)' < 0 \tag{7.33}$$

for $i = 1, \ldots, 2k$ *with* $K = FP^{-1}$.

Definition 7.4.4 and Theorems 7.4.1-7.4.3 are well-known and summarized in [80] and are in fact linear matrix inequalities (LMI). Theorems 7.4.1-7.4.3 allow us to convert an infinite dimensional LMI to a finite dimensional LMI.

Remark 7.4.7. *The main point of Theorems 7.4.1-7.4.3 is that the LMI only needs to hold at the extremal points or vertices of the interval uncertainty or convex hull, namely at most the matrices* $\{A_j\}_{j=1}^{N}$.

Theorem 7.4.4. *The linear matrix inequality that solves problem 7.2.1 for solving a quadratically stable risk-sensitive control that incorporates the interval uncertainty* $\triangle A$ *and* $\triangle B$ *in estimates of* A *and* B *is*

$$\begin{bmatrix} P & (A+A_i)P-(B+B_i)F & I & 0 \\ P(A+A_i)'-F'(B+B_i)' & P & 0 & PC'-F'D' \\ I & 0 & \frac{1}{\sqrt{\mu}}I & 0 \\ 0 & CP-DF & I & \frac{1}{\sqrt{\mu}}I \end{bmatrix} > 0 \quad (7.34)$$

for $i = 1, \ldots, 2k$

The linear matrix inequality for solving a risk-sensitive controller that incorporates the polytopic uncertainty $\triangle A$ *and* $\triangle B$ *in estimate of* A *and* B *is*

$$\begin{bmatrix} P & (A+\sum_i A_i\delta_i)P-(B+\sum_i B_i\delta_i)F & I & 0 \\ P(A+\sum_i A_i\delta_i)'-F'(B+\sum_i B_i\delta_i)' & P & 0 & PC'-F'D' \\ I & 0 & \frac{1}{\sqrt{\mu}}I & 0 \\ 0 & CP-DF & I & \frac{1}{\sqrt{\mu}}I \end{bmatrix} > 0 \quad (7.35)$$

for $\delta_i \in \{\delta^-, \delta^+\}$ for $i = 1, \ldots, k$. This is 2^k LMI constraints.

A gain matrix is then recovered as $K = FP^{-1}$. We refer to this gain matrix as K_{Robust} owing to the fact that it is the gain matrix associated with a controller robust and resilient to uncertainties in the estimates of A and B.

Proof. The LMI solution when the matrices (A, B) are known is found in Remark 7.4.2. Here we incorporate the LMIs with uncertainty in the matrices (A, B). We utilize Theorem that dictates that one needs to only check the vertices of a polytope for stability. This leads to only needing to check the LMI constraints a the vertices which amounts to 2^k constraints. □

Remark 7.4.8. *We would like to compare the performance of K_{Robust}, the control gain resulting from the model based approach in Remark 7.4.3 to the gain matrix K obtained when estimating matrices A and B without incorporating any uncertainties in their estimates (namely $\hat{A}$ and $\hat{B}$) and subsequently the linear matrix inequality is*

$$\begin{bmatrix} P & \hat{A}P - \hat{B}F & I & 0 \\ P\hat{A}' - F'\hat{B}' & P & 0 & PC' - F'D' \\ I & 0 & \frac{1}{\sqrt{\mu}}I & 0 \\ 0 & CP - DF & I & \frac{1}{\sqrt{\mu}}I \end{bmatrix} > 0 \tag{7.36}$$

7.5 Uncertainty Sets

The following steps are used to incorporate uncertainty in system identification using polytopic uncertainty

1. Rollout dynamics for T time steps and for N episodes.

2. Compute least squares on the rolled-out system's states to estimate matrices A and B. Call these estimates $\hat{A}$ and $\hat{B}$.

3. Using estimates $\hat{A}$ and $\hat{B}$, rollout the dynamics to obtain new system states. Use these new system states to solve for new estimates, $\tilde{A}$ and $\tilde{B}$. Repeat this M times.

4. Calculate the uncertainty set.

Remark 7.5.1. *We characterize four different uncertainty sets*

1. *Interval uncertainty set where intervals are defined by maximum/minimum or a percentile of sampled data*

2. *Polytopic uncertainty set where vertices are defined by maximum/minimum or a percentile of sampled data*

3. *Interval uncertainty set where intervals are defined by sample variance of data*

4. *Polytopic uncertainty set where intervals are defined by sample variance of data*

Remark 7.5.2 (**Difference between polytopic and interval uncertainty**)**.** *The following is a short note about the differences between the polytopic and interval uncertainty. Instead of computing a convex hull, we can compute an uncertainty set with* 2^{n^2} *vertices. This uncertainty set encompasses the polytopic uncertainty set. We refer to this as the "conservative robust sys-id" approach. Unfortunately, this approach which uses interval uncertainty requires* 2^{n^2} *linear matrix inequalities. On the other hand, the "robust sys-id" approach which uses polytopic uncertainty requires* $2n^2$ *linear matrix inequalities.*

We show how each of these uncertainty sets are built.

7.5.1 Interval Uncertainty Set based on Maximum/Minimum

Compute the differences between $\hat{A}$ and the M $\tilde{A}$'s element by element in the matrices. Likewise do the same for $\hat{B}$ and $\tilde{B}$. Capture, the maximum and minimum of the M differences for each slot in the matrix. Take the maximum and minimum at each slot in the matrix. Compute the interval uncertainty set where δ_j^+ and δ_j^- refer to maximum and minimum, respectively. Alternatively, we can take a percentile of the range. So instead of the maximum, we can take the 68^{th} percentile.

7.5.2 Polytopic Uncertainty Set Based on Maximum/Minimum

Compute the convex hull of matrices that are all zero except for a single element which contains the maximum or minimum at that position in the matrix.

7.5.3 Interval Uncertainty Set Based on Sample Variance

Another option is to use the sample variance of the sampled matrices and use that as bounds for our uncertainty set. We can do this two ways. The first way uses an interval uncertainty set. We compute the one dimensional sample variance of each element in the sampled matrices. The question becomes how close is the sample variance from the actual variance of the estimator. The estimator is a normal distribution with a mean and variance and so if we take as the uncertainty set one unit of variance in each direction then the uncertainty set covers around 68% of the distribution of sampled matrices. Considering this, we still need to show the sample variance and

actual variance of a least squares estimator are close in some sense. Let σ^2 be the variance of the estimator and μ be the mean of the estimator. Additionally, let S be the sample variance of the estimator. Consider Chebyshev's inequality for the random variable X and mean m

$$P(|X - m| \geq c) \leq \frac{Var(X)}{c^2} \tag{7.37}$$

Now, let X be the sample variance variable. If μ is the mean of the estimator, then the mean of the sample variance is $\frac{N}{N-1}\sigma^2$. Therefore, using Chebyshev's inequality we get

$$P(S - \frac{N}{N-1}\sigma^2 \geq c) \leq \frac{var(S)}{c^2} \tag{7.38}$$

The question is what is the variance of the sample variance (var(S)). It has been shown previously that var(S) $\sim (\mathcal{O}(\frac{1}{N}))$ This means that the distance between the sample variance and population variances varies inversely (linearly) with the number of samples. We do this for each dimension of the sampled matrices. This is an interval uncertainty set where a sample variance is computed for each element of the sampled matrices for a total of n^2 sample variances. As we choose more samples to build up our uncertainty set, we can be more confident using the sample variance as bounds for an uncertainty set. We obtain a proper uncertainty set that encompasses around 68 percent of samples of the matrices, which is a fairly robust uncertainty set to encompass.

7.5.4 Polytopic Uncertainty Set Based on Sample Variance

The second way to use sample variance is to compute the covariance matrix of the entire sampled matrix set and then use the eigenvectors and eigenvalues of that set to find boundary points. We then take the convex hull of these boundary points. We assemble the sampled matrices A, vectorize them, and then assemble them into a matrix called $\mathcal{A}$. The covariance matrix is then $\mathcal{A}' * \mathcal{A}$.

7.6 Simulations

We compare the performances of controllers for the case when the matrices A and B must be estimated ($\hat{A}, \hat{B}$), to the case when the matrices A and B are known, and finally, to the case when the matrices are estimated and uncertainties in the estimates are incorporated ($\hat{A} + \triangle A$ and $\hat{B} + \triangle B$). The first case (using estimates $\hat{A}, \hat{B}$) in Figure 7.1 is referred to as "sys id", the second case (using estimates A, B) as "no sys id", and the third as "robust sys-id" (incorporates uncertainty and uses the gain matrix K_{Robust}). We see that the performance that uses robust sys-id is conservative and outperforms some of the performances due to estimates from the 100 renditions of "sys-id" while being outperformed by other estimates. This is due to the robust nature of the robust model-based approach for risk sensitive controller design.

7.6.1 Example 1: LQR Simple Example (Model-Based)

Consider the following example for LQR, where the relevant matrices are

$$A = \begin{bmatrix} 1.01 & 0.01 & 0 \\ 0.01 & 1.01 & 0.01 \\ 0 & 0.01 & 1.01 \end{bmatrix} \quad B = I \tag{7.39}$$

$$Q = I \quad R = I \tag{7.40}$$

where $x_{t+1} = Ax_t + Bu_t + w_t$ and the time horizon, T is 15. We plot the cost functions for the four different approaches. We repeat the experiments for 100 iterations.

We compare the LQR cost using model-based methods. Figure 7.1 has four costs. The first corresponds to when the matrices A and B are known. The second deals with the case, when least squares is employed on $M = 100$ episodes. The third case incorporates robustness into the system identification. Not only is least squares employed here, but a confidence interval around each element in the matrices A and B are formed. These confidence intervals are then incorporated in the linear matrix inequality used to solve for the gain matrix for LQR. This is referred to as "robust sys-id". The last case referred to as "conservative robust sys-id" incorporates the confidence interval as well, but in a slightly different way. In "robust sys-id," polytopic uncertainty characterizes the confidence interval. We note in the example, that a steady state controller is computed. While, in "conservative robust sys-id" interval uncertainty characterizes the confidence interval. However, in practice, the time horizon may not be long enough to warrant

a steady state controller. We use here a smaller time horizon in order not to have to deal with large costs. We fully appreciate the potential discrepancy between using a steady state controller for a short time horizon problem. However, as this is a relatively small problem, the cost converges relatively quickly.

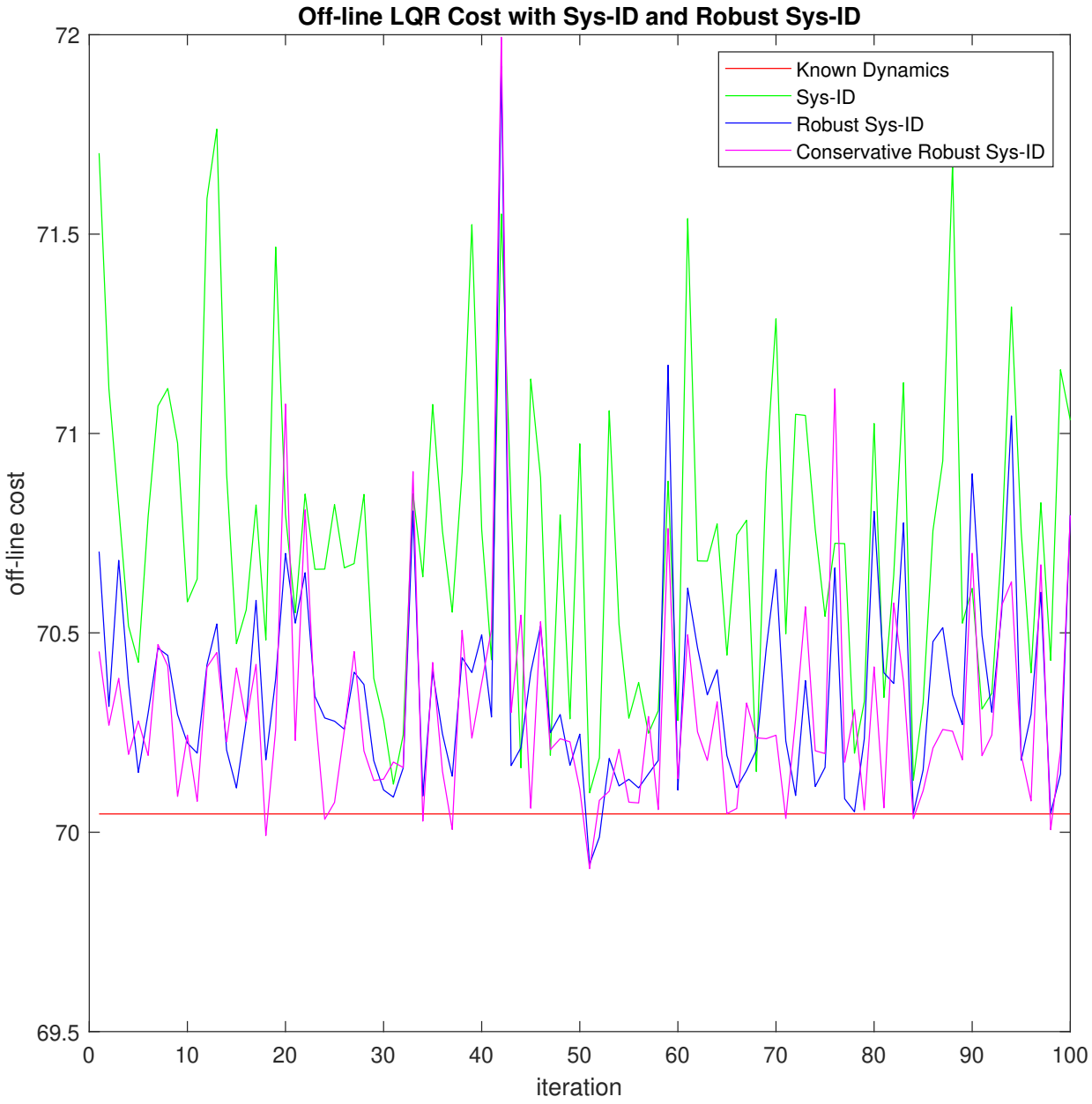

Figure 7.1: Off-line LQR Cost with Sys-ID and Robust Sys-ID SIMPLE Example

7.6.2 Example 1: LEQG Simple Example (Model-Based)

Note, for purposes of LEQG, we use smaller values for matrices Q and R as the larger values can lead to blow up when using an exponential in the cost function as in LEQG.

$$A = \begin{bmatrix} 1.01 & 0.01 & 0 \\ 0.01 & 1.01 & 0.01 \\ 0 & 0.01 & 1.01 \end{bmatrix} \quad B = I \tag{7.41}$$

$$Q = 10^{-3} \times I \quad R = 10^{-3} \times I \tag{7.42}$$

leading to a lot smaller cost value than LQR as in the second figure. We plot the cost function resulting from the LEQG controller in Figure 7.2. Similar to Figure 7.1, we plot 4 different cases: when the dynamics are known, when leastsquares is uses, when polytopic uncertainty is Incorporated, and when interval uncertainty is incorporated.

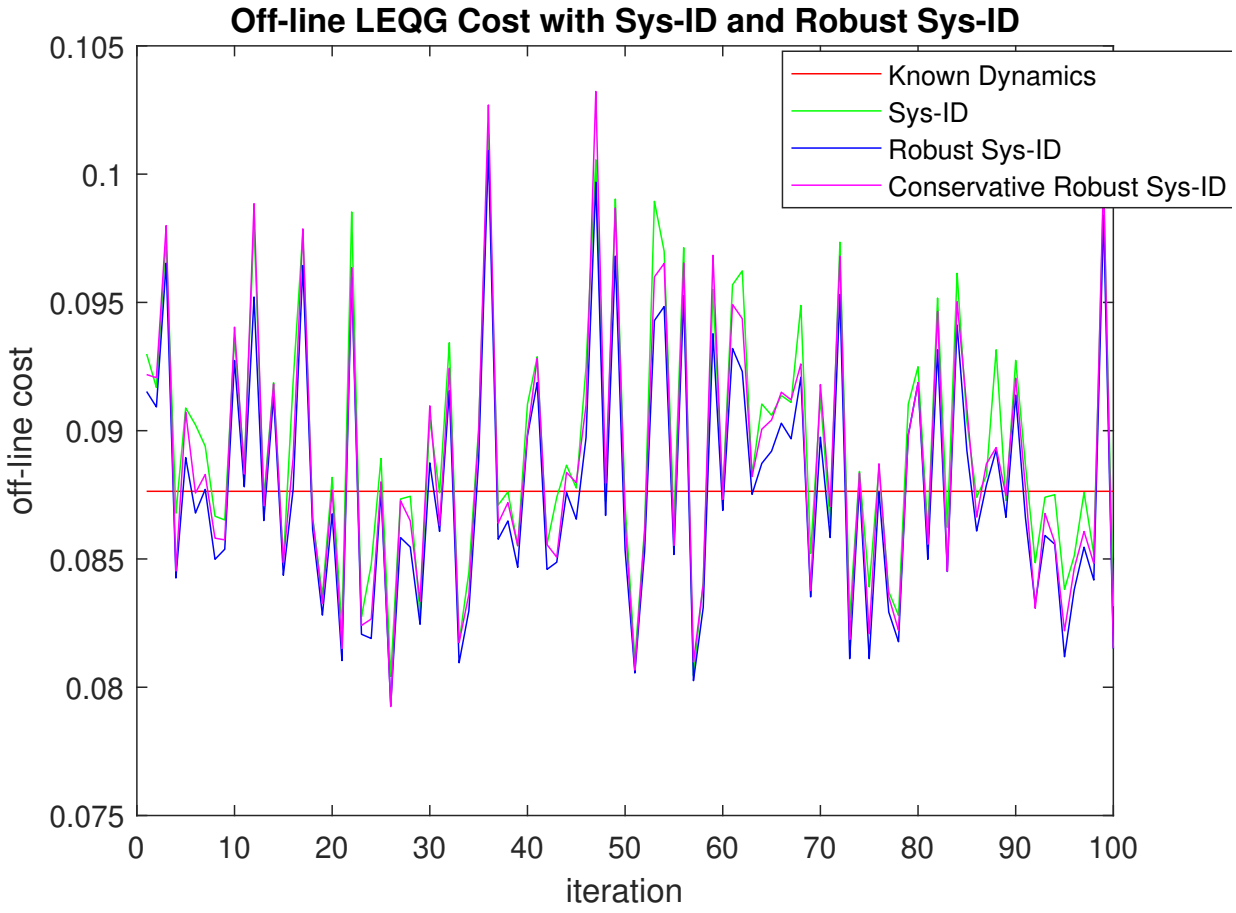

Figure 7.2: Off-line LEQG Cost with Sys-ID and Robust Sys-ID SIMPLE Example

7.6.3 Example 1: Discussion (Model-Based)

We can see from Figures 7.1, 7.2 that the ”robust sys-id” outperforms the ”sys-id” approach with a lower cost. This shows that incorporating uncertainty into system identification and then using this uncertainty in the linear matrix inequalities that solve for the gain matrix improves the output controller. We can do slightly better (although not much more) with the ”conservative robust sys-id” approach which incorporates interval uncertainty instead of polytopic uncertainty. Obviously, when the dynamics are known and do not need to be estimated, we obtain the most cost-efficient controller. We compare these four approaches in 100 cases.

The same approaches are applied to LEQG. In LEQG, we use smaller Q and R and therefore the costs are drastically smaller than in the LQR case. Additionally, the controller used is solved for LEQG but the cost shown in Figure 7.2 is without the exponential. Again, as in the LQR case, we see that "robust sys-id" approach outperforms the "sys-id" approach. This highlights that incorporating robustness in the estimates of the dynamics improves the controller. However, here we see that "conservative robust sys-id" performs worse than "robust sys-id". Perhaps in this case, the larger uncertainty set in "conservative robust sys-id" is leading to a more conservative controller that performs worse.

7.6.4 Example 1: LQR Simple Example (Model-Free vs. Model-Based)

Figure 7.3 compares the model-free to model-based approach when the linear dynamics are unknown for the LQR simple example.

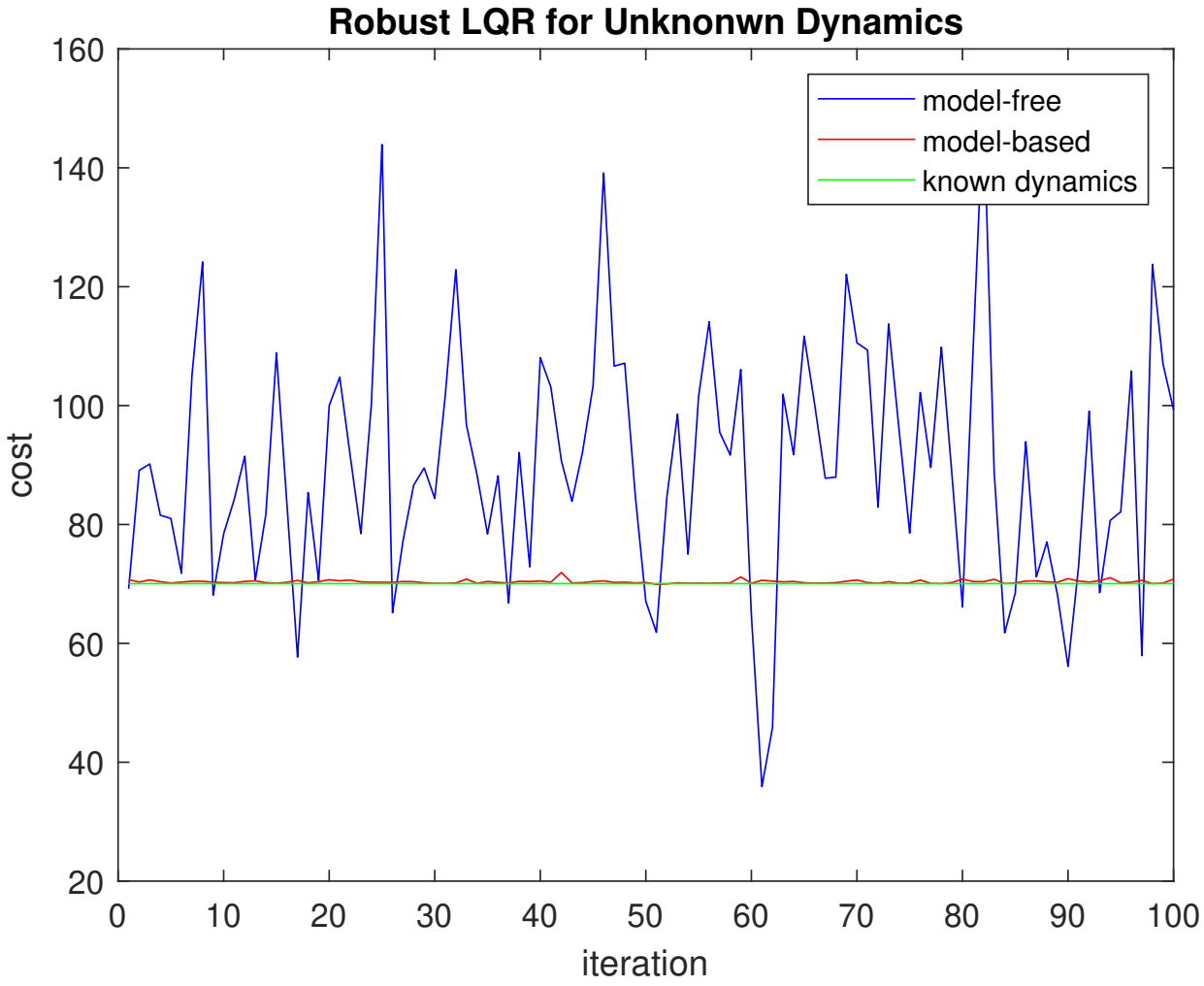

Figure 7.3: LQR Cost Model-Free Model-Based SIMPLE

7.6.5 Example 1: LEQG Simple Example (Model-Free vs. Model-Based

Again, the model based-approach and model-free approach are compared in the LEQG setting in Figure 7.4.

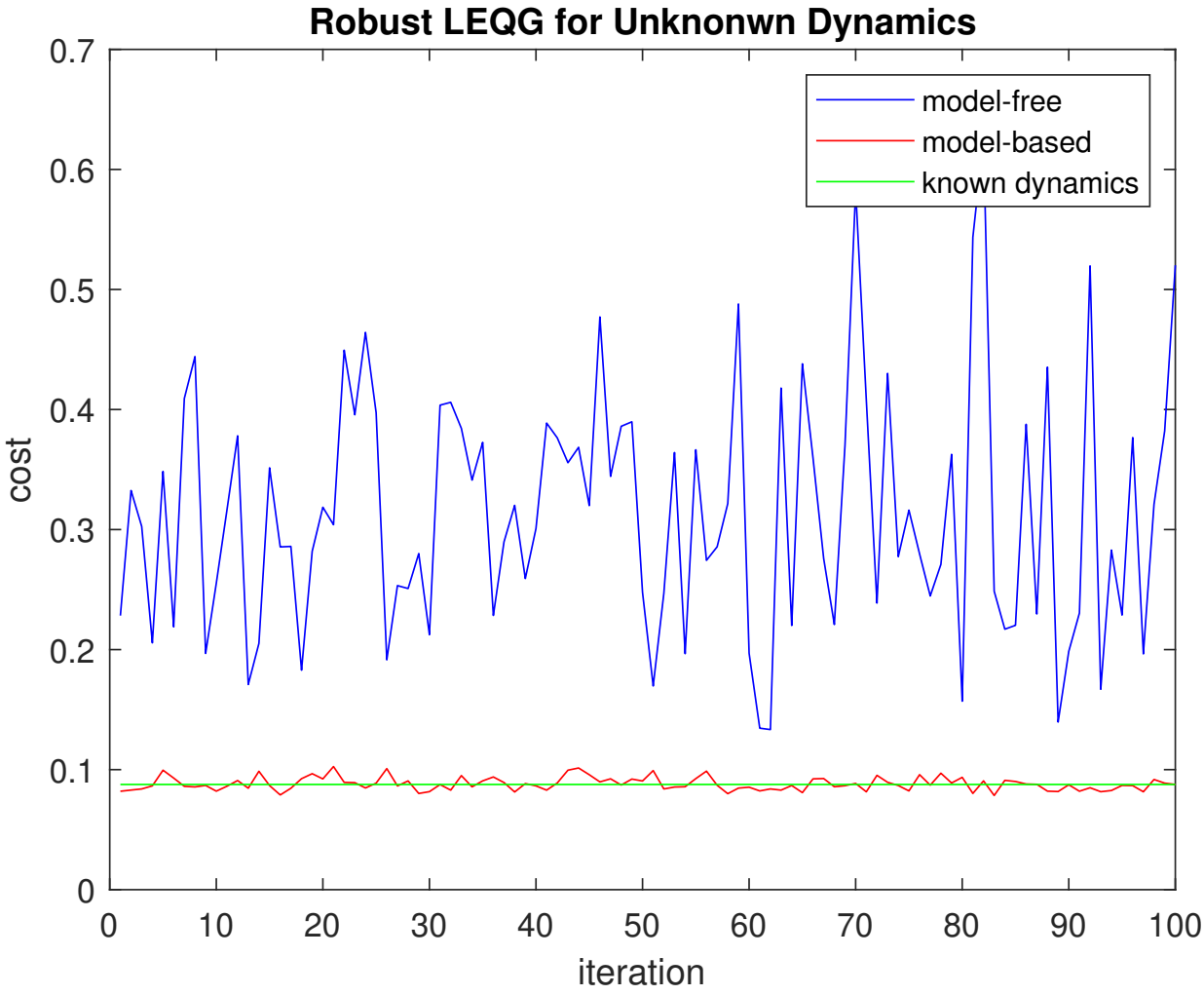

Figure 7.4: LEQG Cost Model-Free Model-Based SIMPLE

7.6.6 Example 1: Discussion (Model-Free vs. Model-Based)

The model-based approach performs better than the model-free approach in both LQR and LEQG. In the model-based approach, we estimate the matrices A and B and then solve for the LQR controller. In the model-free approach, gradients are estimated and then the control matrix is updated using these gradients. Unsurprisingly, the model-based approach outperforms the model-free approach because the model-based approach utilizes the structure of the problem

more.

7.6.7 Example 2: LQR Vehicle Example (Model-Based)

Consider the second LQR example, a vehicle example where the matrices are defined as follows

$$A = \begin{bmatrix} 1.0 & 0 & 0.1 & 0 \\ 0 & 1.0 & 0 & 0.1 \\ 0 & 0 & 0.9 & 0 \\ 0 & 0 & 0 & 0.9 \end{bmatrix} \quad B_u = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0.1 & 0 \\ 0 & 0.1 \end{bmatrix} B_w = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0.1 & 0 \\ 0 & 0.1 \end{bmatrix} \quad Q = I_4 \quad R = I_2$$

where $x_{t+1} = Ax_t + B_u u_t + B_w w_t$ and the time horizon, T is 15. See Figure 7.5 for the LQR cost when the dynamics are known, the dynamics are estimated using least squares, and the dynamics are estimated using uncertainty sets.

7.6.8 Example 2: LEQG Vehicle Example (Model-Based)

We can also consider an LEQG setting. Note, for purposes of LEQG, we use smaller values for matrices Q and R as the larger values can lead to blow up when using an exponential in the cost function as in LEQG. Therefore, Q and R for LEQG in the vehicle example are

$$Q = 10^{-3} \times I \quad R = 10^{-4} \times I \tag{7.43}$$

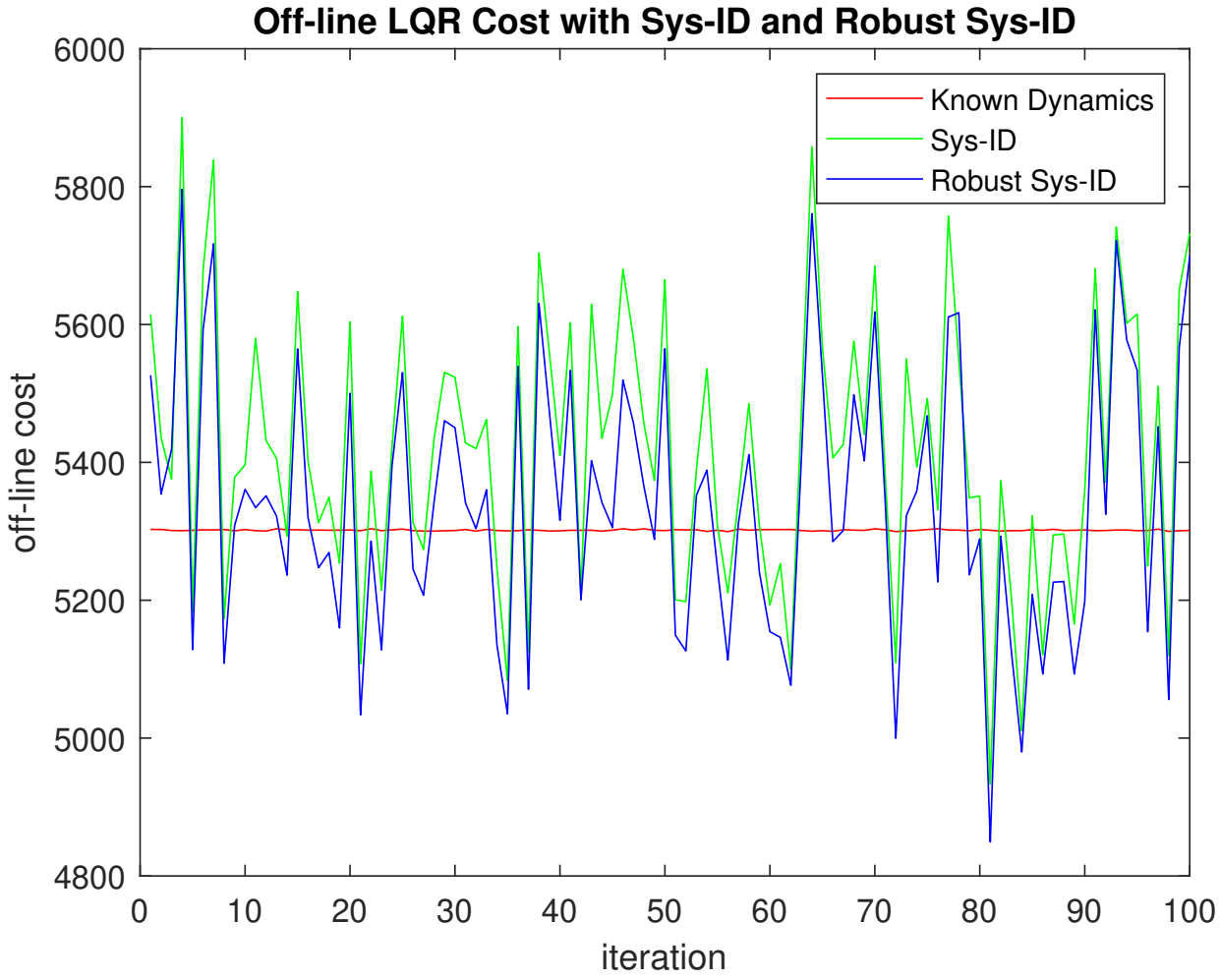

Figure 7.5: Off-line LQR Cost with Sys-ID and Robust Sys-ID VEHICLE

leading to a lot smaller cost value than LQR. See Figure 7.6 for the LEQG cost when the dynamics are known, the dynamics are estimated using least squares, and the dynamics are estimated using uncertainty sets.

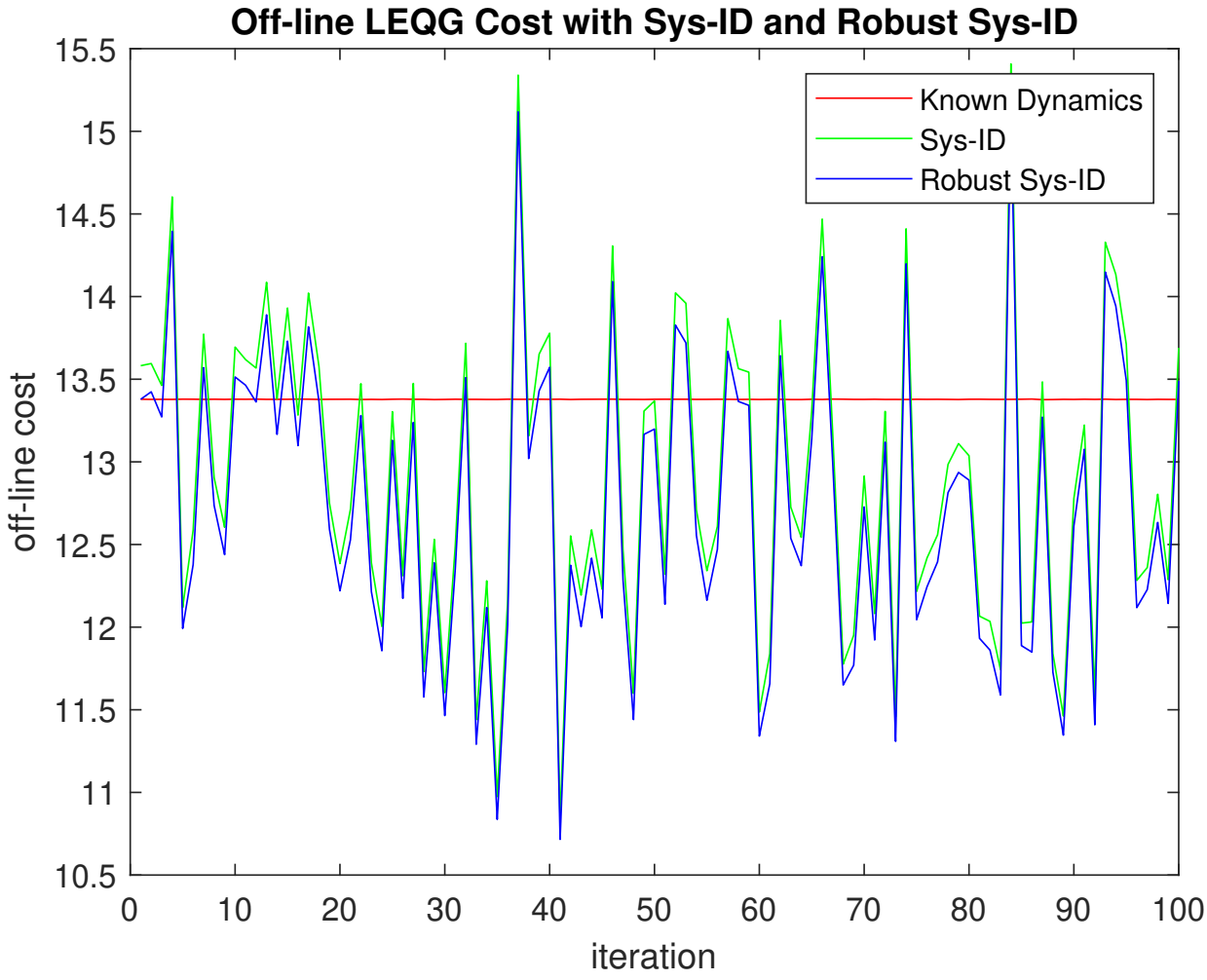

Figure 7.6: Off-line LEQG Cost with Sys-ID and Robust Sys-ID VEHICLE

7.6.9 Example 2: Discussion (Model-Based)

Again, the ”robust sys-id” approach outperforms the ”sys-id” approach both in the LQR and LEQG setting as in the SIMPLE example. Also, like in the SIMPLE example, here the ”robust sys-id” approach outperforms the ”sys-id” approach more in the LQR setting than in the LEQG setting.

7.6.10 Example 2: LQR Vehicle Example (Model-Free vs. Model-Based)

We compare the model-free and model-based LQR costs for the vehicle example in Figure 7.7. As is expected, the cost from the model-based approach outperforms the cost from model-free approach.

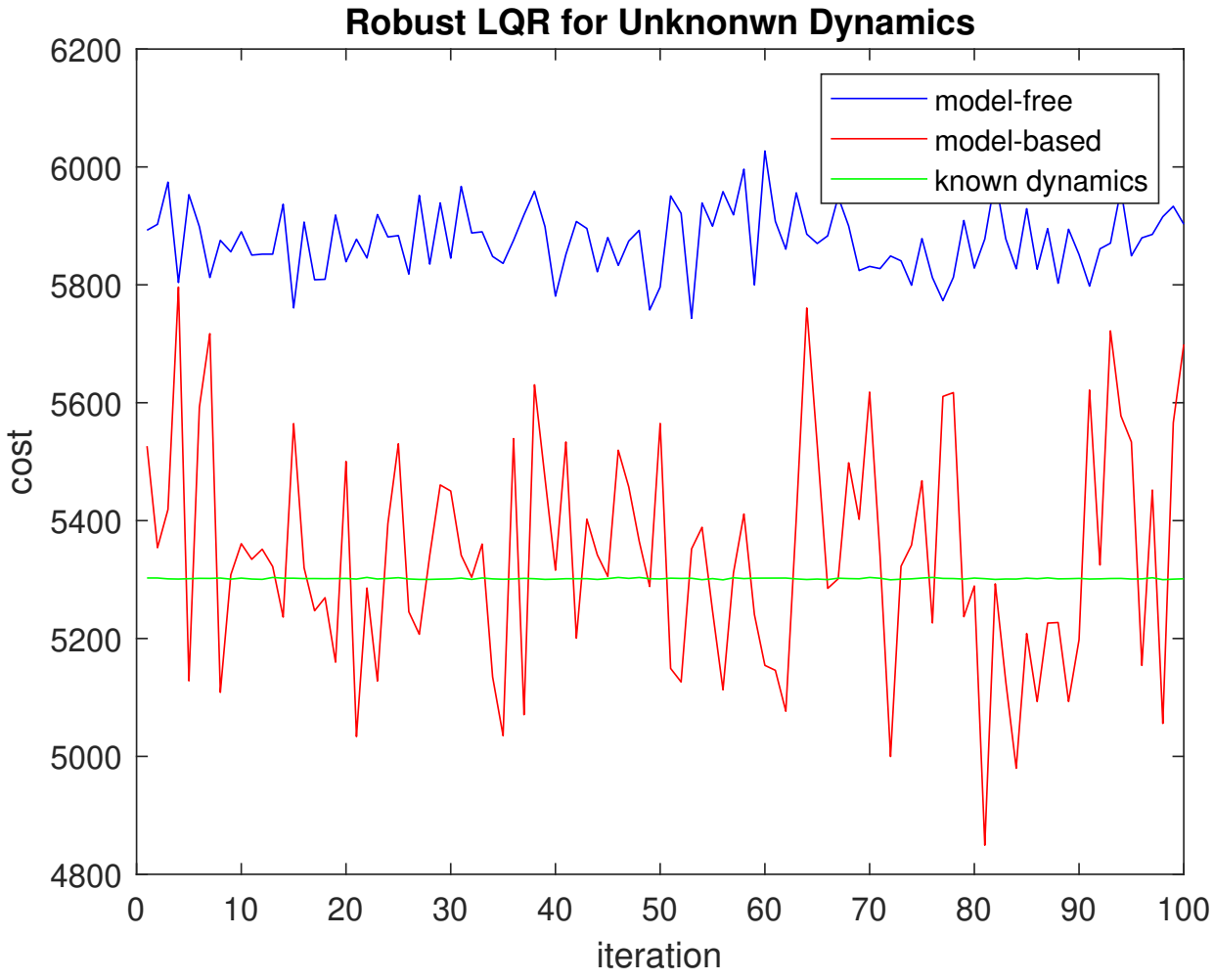

Figure 7.7: LQR Cost Model-Free Model-Based VEHICLE

7.6.11 Example 2: LEQG Vehicle Example (Model-Free vs. Model-Based)

We compare the model-free and model-based LEQG costs for the vehicle example in Figure 7.8. As in the LQR case in Figure 7.7, the cost from the model-based approach outperforms the cost from model-free approach.

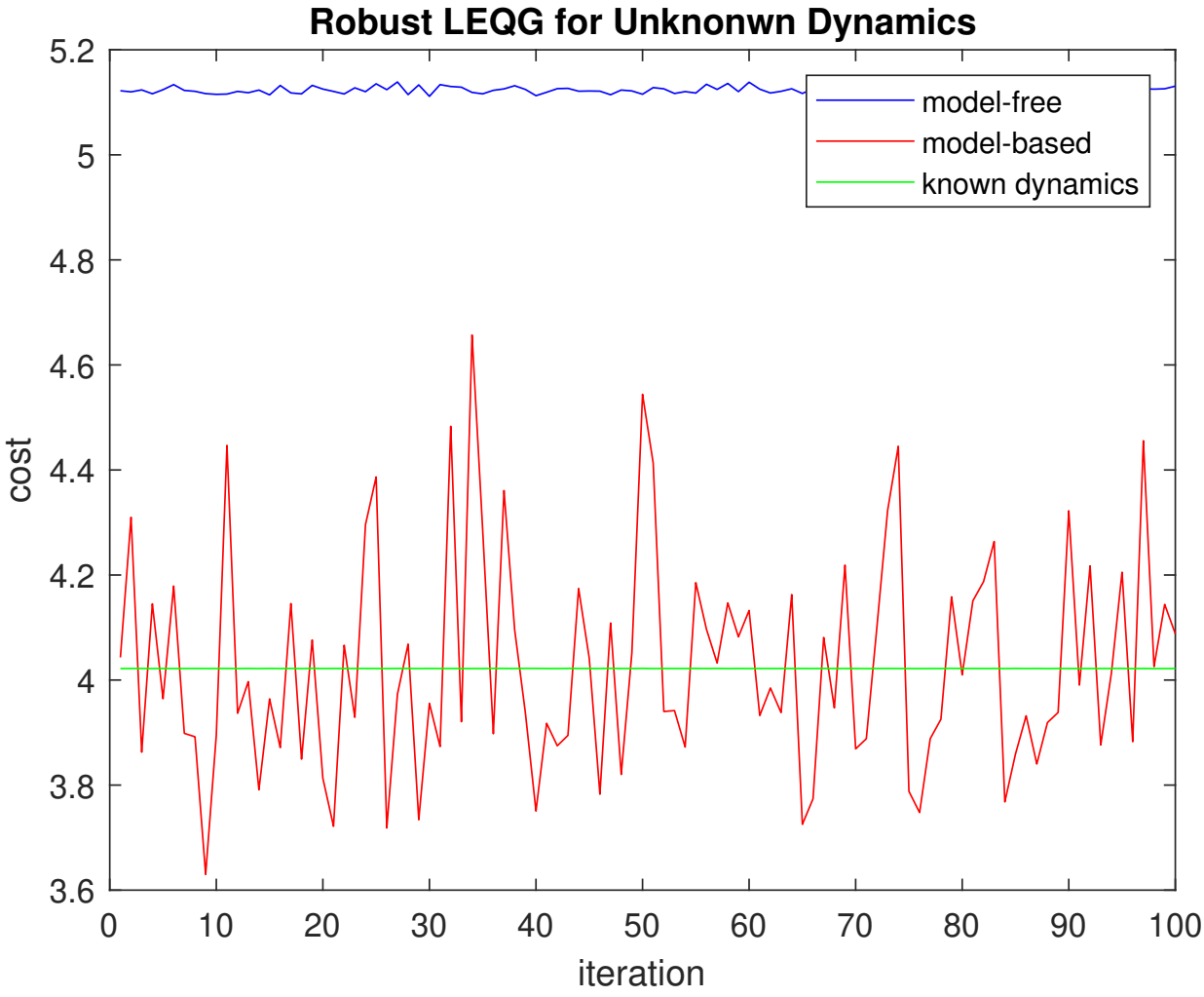

Figure 7.8: LEQG Cost Model-Free Model-Based VEHICLE

7.6.12 Example 2: Discussion (Model-Free vs. Model-Based)

Like in the Simple example. the model-based approach outperforms the model-free approach in both the LQR and LEQG settings in the vehicle example.

7.7 Conclusion

In the final chapter, we studied the risk-sensitive control problem when the dynamics are unknown. We showcased both model-based and model-free solutions in both the LQR and LEQG setting. We also applied the approaches to two examples. The model-based and model-free approaches for LQR builds on previous work ([6], [5]). We extend those works in multiple ways. First, in the model-free approach we implement an algorithm using a two-sided simulated gradient instead of a one-sided simulated gradient ([5]). Secondly, in the model-based approach we propose a linear matrix inequality solution that incorporates uncertainty in the least squares estimator. We also suggest different ways of building the uncertainty set using interval and polytopic sets. Thirdly, we showcase the model-free and model-based approach in both the LQR and LEQG settings (something not done in ([6]) and [5]).

There are multiple directions for further exploration. We highlight some of them here. Firstly, we can work to build better, less-conservative uncertainty sets. Secondly, we can apply our algorithm to model-predictive control (MPC) ([81]). Thirdly, we can implement risk-sensitive Q-learning another model-free approach in addition to policy optimization. Risk-sensitive Q-learning has been discussed in ([82]). Fourthly, there are other approaches for incorporating risk (ie. cumulative prospect theory) that we could explore other than exponential of the utility function ([83], [84], [85], [86]). Fifthly, there is further work to be done on extending the

algorithm to unknown nonlinear dynamics. In the nonlinear setting, we don't have the benefit of assuming the controller is linear with respect to the state. One possible way around this is to parameterize the controller, we would like to find, as a neural network. Then employ policy optimization, as in the linear case, where the parameters of the neural network representing the controller are updated. And lastly another direction that can be taken is to use ideas from risk sensitive and robust control for adversarial neural networks. The problem of optimization in neural networks can be viewed as an optimal control problem. In light of this, adversarial neural networks could be viewed as a risk-sensitive or robust optimal control problem.

Chapter 8: Future Directions

We explore here future directions for all three parts of the thesis: sensor scheduling, distributed risk sensitive control, and risk sensitive control with unknown dynamics.

In Chapter 3 on the topic of sensor scheduling, one possible future direction is to extend the semidefintie program solution to distributed estimation problems. Another future direction is to extend the tracking algorithm 2 to a multi-sensor problem.

Regarding Chapter 4 on the topic of sensor scheduling, in the future we can explore algorithms that would optimally minimize the upper bound of the MSE. The upper bound is a fraction, and so the challenge is maximizing the denominator and minimizing the numerator simultaneously. Additionally, the discussion until now for sensor scheduling has been exclusively in the linear setting. A future direction would be to tackle sensor scheduling for nonlinear systems. One possible approach would be to formulate the sensor scheduling problem as a POMDP and then use Monte-Carlo sampling as in [38]. Monte-Carlo sampling has been studied in MDPs in [62] and POMDPs in [63].

Regarding Chapter 6 on the topic distributed risk sensitive control, in the future we could consider improved upper bounds for the distributed LEQG and better understanding of what the current bounds mean. Additionally, there are other approaches to distributed control that we can consider that include distributed optimization ([73], [74], [75]).

In Chapter 7 on the topic of risk sensitive control with unknown dynamics, there are multiple directions for further exploration. Firstly, we can work to build better, less-conservative uncertainty sets. Secondly, we can apply our algorithm to model-predictive control (MPC) ([81]). Thirdly, we can implement risk-sensitive Q-learning another model-free approach in addition to policy optimization. Risk-sensitive Q-learning has been discussed in ([82]). Fourthly, there are other approaches for incorporating risk (ie. cumulative prospect theory) that we could explore other than exponential of the utility function ([83], [84], [85], [86]). Fifthly, there is further work to be done on extending the algorithm to unknown nonlinear dynamics. In the nonlinear setting, we don't have the benefit of assuming the controller is linear with respect to the state. One possible way around this is to parameterize the controller, we would like to find, as a neural network. Then employ policy optimization, as in the linear case, where the parameters of the neural network representing the controller are updated. And lastly another direction that can be taken is to use ideas from risk sensitive and robust control for adversarial neural networks. The problem of optimization in neural networks can be viewed as an optimal control problem. In light of this, adversarial neural networks could be viewed as a risk-sensitive or robust optimal control problem.

Bibliography

[1] D. Hartman and J. Baras, *"Near-Optimal Solution to the Non-Uniform Sampling Problem in Kalman Filtering"*, IEEE Conference on Decision and Control (2019).

[2] D. Maity, D. Hartman, and J. Baras, *"Sensor scheduling for Linear Dynamical Systems: A Covariance Tracking based Approach"*, Automatica (2021).

[3] D. Hartman, D. Maity, and J. Baras, *"Upper Bound Relaxation for Sensor Scheduling of Linear Dynamical Systems"*, (In Preparation) (2020).

[4] F. Borrelli and T. Keviczky, *"Distributed LQR design for identical dynamically decoupled systems"*, IEEE Transactions on Automatic Control (2008).

[5] K. Zhang, B. Hu, and T. Basar, *"Policy Optimization for H_2 linear control with H_∞ robustness guarantee: Implicit regularization and global convergence"*, arXiv preprint arXiv:1910.09496 (2019).

[6] S. Dean, H. Mania, N. Matni, B. Recht, and S. Tu, *"On the sample complexity of the linear quadratic regulator"*, Foundations of Computational Mathematics (2019).

[7] I. Matei and J. Baras, *Consensus-based distributed linear filtering*, IEEE Conference on Decision and Control (2010).

[8] S. Boyd, *Stanford University EE 363 Linear Dynamical Systems Lecture Notes*, https://web.stanford.edu/class/ee363/lectures.html (2008).

[9] M. Khan, *"Matrix Inversion Lemma and Information Filter"*, Honeywell Technology Solutions Lab (2005).

[10] C. Van Loan and G. Golub, *"Matrix Computation"*, Johns Hopkins University Press (1983).

[11] L. Dritsas, *"Notes on Robust Control, H_∞, LMIS, BRL"*, (2011).

[12] D. Bertsekas, *"Dynamic programming and optimal control"*, Athena Scientific (2011).

[13] A. Krause and D. Golovin, *"Submodular Function Maximization"*, (2001).

[14] J. Hassan and J. Shamma, *"Distributed submodular minimization and motion coordination over discrete space"*, arXiv preprint arXiv:1709.07379 (2017).

[15] G. Nemhauser, L. Wolsey, and M. Fisher, *"An analysis of approximations for maximizing submodular set functions"*, Mathematical Programming (1978).

[16] T. Basar and P. Bernhard, *"H_∞- Optimal Control and Related Minmax Design Problems"*, (1995).

[17] V. Yakubovich, *"The solution of certain matrix inequalities in automatic control theory"*, Soviet Math (1962).

[18] P. Bernhard, *"A lecture on the game theoretic approach to H_∞ optimal control"*, (1991).

[19] C. Scherer, *"Riccati Inequality and State Space H_∞ Optimal Control"*, Doctoral dissertation, Julius Maximilians University Würzburg, Germany (1990).

[20] D. Arzelier, *"LMIS in systems control state-space methods performance analysis and synthesis"*, (2008).

[21] R. Caverly and J. Forbes, *"LMI properties and applications in systems, stability, and control theory"*, arXiv preprint arXiv:1903.08599 (2019).

[22] M. Fazel, R. Ge, S. Kakade, and M. Mesbahi, *"Global convergence of policy gradient methods for the linear quadratic regulator"*, International Conference on Machine Learning (2018).

[23] M. James, J. Baras, and R. Elliott, *"Risk-sensitive control and dynamic games for partially observed discrete-time nonlinear systems"*, IEEE Transactions on Automatic Control (1994).

[24] M. James, J. Baras, and R. Elliott, *Output feedback risk-sensitive control and differential games for continuous-time nonlinear systems*, Proceedings of 32nd IEEE Conference on Decision and Control (1992).

[25] M. James and J. Baras, *Partially Observed Differential Games, Infinite-Dimensional Hamilton–Jacobi–Isaacs Equations, and Nonlinear H_∞ Control*, SIAM Journal on Control and Optimization 34.4: 1342-1364 (1996).

[26] J. Baras, *Maximum Entropy Models, Dynamic Games, and Robust Output Feedback Control for Nonlinear Systems*, Proceedings of the 45th IEEE Conference on Decision and Control (2006).

[27] J. Baras and M. Rabi, *aximum entropy models, dynamic games, and robust output feedback control for automata*, Proceedings of the 44th IEEE Conference on Decision and Control (2005).

[28] J. Baras and M. James, *Robust output feedback control for discrete-time nonlinear systems: the finite-time case*, Proceedings of 32nd IEEE Conference on Decision and Control (1993).

[29] M. James and E. R. Baras, John, *Risk-sensitive control and dynamic games for partially observed discrete-time nonlinear systems*, IEEE transactions on automatic control 39.4: 780-792 (1994).

[30] D. Jacobson, *"Optimal stochastic linear systems with exponential performance criteria and their relation to deterministic differential games"*, IEEE Transactions on Automatic Control (1973).

[31] M. Fu, *"Stochastic gradient estimation" Handbook of simulation optimization*, Springer (2015).

[32] A. Mourikis and S. Roumeliotis, *"Optimal sensor scheduling for resource-constrained localization of mobile robot formations"*, IEEE Transactions on Robotics (2006).

[33] J. Löfberg, *YALMIP : A Toolbox for Modeling and Optimization in MATLAB*, In Proceedings of the CACSD Conference (2004).

[34] I. CVX Research, *CVX: Matlab Software for Disciplined Convex Programming, version 2.0*, (2012).

[35] L. Meier, J. Peschon, and R. Dressler, *"Optimal control of measurement subsystems"*, IEEE Transactons on Automatci Control (1967).

[36] M. Vitus, W. Zhang, A. Abate, J. Hu, and C. Tomlin, *"On efficient sensor scheduling for linear dynamical systems"*, Automatica (2012).

[37] S. Joshi and S. Boyd, *"Sensor selection via convex optimization"*, IEEE Transactions on Signal Processing (2008).

[38] Y. He and C. K.P., *"Sensor scheduling for target tracking in sensor networks"*, IEEE Conference on Decision and Control (2004).

[39] A. Chhetri, D. Morrell, and Papandreou-Suppappola, *"On the use of binary programming for sensor scheduling"*, IEEE Transactions on Signal Processing (2007).

[40] S. Liu, M. Fardad, E. Masazade, and P. Varshney, *"Optimal periodic sensor scheduling in networks of dynamical systems"*, IEEE Transactions on Signal Processing (2014).

[41] T. Soleymani, S. Hirche, and J. Baras, *"Optimal information control in cyber-physical systems"*, IFAC Workshop on Distributed Estimation and Control in Networked Systems (2016).

[42] S. A. H. B. Kailath, T., *Linear Estimation*, Prentice Hall (2000).

[43] L. Vandenberghe and S. Boyd, *"Semidefinite Programming"*, SIAM Review (1996).

[44] S. Boyd and L. Vandenberghe, *Convex optimization*, Cambridge university press (2004).

[45] D. Axehill, L. Vandenberghe, and A. Hansson, *"Convex relaxations for mixed integer predictive control"*, Automatica (2010).

[46] V. S. Mai, D. Maity, B. Ramassubramanian, and M. Rotkowitz, *"Convex methods for rank-constrained optimization problems"*, Proceedings of the Conference on Control and its Applications, Society for Industrial and Applied Mathematics (2015).

[47] Y. Wang, H. Jie, and L. Cheng, *"A Fusion Localization Method based on a Robust Extended Kalman Filter and Track-Quality for Wireless Sensor Networks"*, Sensors (2019).

[48] V. Tzoumas, A. Jadbabaie, and G. Pappas, *"Near-optimal sensor scheduling for batch state estimation: Complexity,algorithms, and limits"*, IEEE Conference on decision and Control (2016).

[49] B. Sinopoli, L. Schenato, M. Francheschetti, K. Poolla, M. Jordan, and S. Sastry, *"Kalman filtering with intermittent observations"*, IEEE Transactions on Automatic Control (2004).

[50] R. Zhang and J. Lavaei, *"Sparse semidefinite programs with near-linear time complexity"*, IEEE Conference on Decision and Control (2018).

[51] L. Chamon, G. Pappas, and A. Ribeiro, *"The mean square error in Kalman filtering sensor selection is approximately supermodular"*, IEEE Conference on Decision and Control (2017).

[52] S. T. Jawaid and S. L. Smith, *"Submodularity and greedy algorithms in sensor scheduling for linear dynamical systems"*, Automatica (2015).

[53] A. Clark, *"Submodularity in dynamics and control of networked systems"*, Springer (2016).

[54] T. Summers, F. Cortesi, and J. Lygeros, *"On submodularity and controllability in complex dynamical systems"*, IEEE Transactions on Control of Network Systems 3.1 (2016).

[55] A. Olshevsky, *"Minimal Controllability Problems"*, IEEE Transcations Control of Network Systems (2014).

[56] V. Tzoumas, N. Atanasov, A. Jadbabaie, and G. Pappas, *"Scheduling nonlinear sensors for stochastic process estimation"*, IEEE Conference on decision and Control (2016).

[57] P. Singh, M. Chen, L. Carlone, S. Karaman, E. Frazzoli, and D. Hsu, *"Supermodular mean squared error minimization for sensor scheduling in optimal Kalman filtering"*, American Control Conference (2017).

[58] J. Baras and A. Bensoussan, *"Optimal sensor scheduling in nonlinear filtering of diffusion processes"*, SIAM Journal on Control and Optimization (1989).

[59] K. Astrom and B. Bernhardsson, *"Comparison of Riemann and Lebesgue sampling for first order stochastic systems"*, IEEE Conference on Decision and Control (2002).

[60] J. Tropp, *Column subset selection, matrix factorization, and eigenvalue optimization*, Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics (2009).

[61] K. Somasundaram and J. Baras, *"Performance improvements in distributed estimation and fusion induced by a trusted core"*, IEEE 12th International Conference on Information Fusion (2009).

[62] H. Chang, M. Fu, and S. Marcus, *An adaptive sampling algorithm for solving Markov decision processes*, Operations Research 53, no. 1: 126-139. (2005).

[63] D. Silver and J. Veness, *Monte-Carlo planning in large POMDPs*, Neural Information Processing Systems (2010).

[64] A. Savkin, R. Evans, and E. Skafidas, *"The problem of optimal robust sensor scheduling"*, Systems Control Letters (2001).

[65] U. Shaked and T. Yahalo, *H_∞/-optimal estimation: a tutorial*, Proceedings of the 31st IEEE Conference on Decision and Control (1992).

[66] S. Lall, *EE 365: Lecture on Infinite Horizon Linear Exponential Quadratic Regulator*, https://web.stanford.edu/class/ee365/lectures/leqr.pdf (2014).

[67] M. C. Campi and M. R. James, *"Nonlinear discrete-time risk-sensitive optimal control"*, International Journal of Robust and Nonlinear Control (1996).

[68] P. Chrsitofides, R. Scattolini, D. Munoz de la Pena, and J. Liu, *Distributed model predicitve control: A tutorial review and future research directions*, Computers Chemical Engineering pp. 21-41 (2013).

[69] E. Vlahakis, L. Dritsas, and G. Halikias, *Distributed LQR design for a class of large-scale multi-area power systems*, Energies 12 no. 14 (2019).

[70] L. Lessard, *"Optimal control of a fully decentralized quadratic regulator"*, Allerton Conference on Communication, Control, and Computing (2012).

[71] S. Boyd, *Lecture Notes on Infinite Horizon Linear Quadratic Regulator*, https://stanford.edu/class/ee363/lectures.html (2008).

[72] D. Hartman and J. Baras, *A Distributed LQG and LEQG Controller*, In Preparation (2021).

[73] C. Conte, T. Summers, M. Zellinger, M. Morari, and C. Jones, *Distributed algorithms for optimization problems with equality constraints*, IEEE Conference on Decision and Control (2012).

[74] A. Nedić and J. Liu, *Distributed optimization for control*, Annual Review of Control, Robotics, and Autonomous Systems (2018).

[75] I. Matei and J. Baras, *Distributed algorithms for optimization problems with equality constraints*, IEEE Conference on Decision and Control (2013).

[76] G. Duan and H. Yu, *Control Systems: Analysis, Design and Applications*, Boca Raton, FL: CRC Press (2013).

[77] M. Fu, *Stochastic gradient estimation*, Handbook of simulation optimization: 105-147 (2015).

[78] Y.-S. Wang, N. Matni, and J. Doyle, *A system-level approach to ocntroller synthesis*, IEEE Transactions on Automatic Control: 4079-4093 (2019).

[79] B. Krishnakumar and S. Ghadimi, *"Zeroth-order (non)-convex stochastic optimization via conditional gradient and gradient updates"*, Proceedings of the 32nd International Conference on Neural Information Processing Systems (2018).

[80] M. Peet, *"http://control.asu.edu/Classes/MAE598/598Lecture13.pdf"*, LMIs in Control (2020).

[81] A. Bemporad and M. Morari, *Robust model predictive control: A survey*, Robustness in identification and control (pp. 207-226) (1999).

[82] V. Borkar, *Q-learning for risk-sensitive control*, Mathematics of operations research 27, no. 2: 294-311 (2002).

[83] L. Prashanth, J. Cheng, M. Fu, S. Marcus, and C. Szepesvari, *Cumulative prospect theory meets reinforcement learning: Prediction and control*, International Conference on Machine Learning, pp. 1406-1415 (2016).

[84] Y. Shen, M. Tobia, and K. Obermayer, *Risk-sensitive reinforcement learning*, Neural computation 26, no. 7: 1298-1328 (2014).

[85] O. Mihatsch and R. Neuneier, *Risk-sensitive reinforcement learning*, Machine learning 49, no. 2: 267-290 (2002).

[86] P. Geibel and F. Wysotzki, *Risk-sensitive reinforcement learning applied to control under constraints*, Journal of Artificial Intelligence Research 24: 81-108 (2005).