12-2014

Robust Model Predictive Control with Anytime Estimation

Truong X Nghiem
University of Pennsylvania, nghiem@seas.upenn.edu

Yash Vardhan Pant
University of Pennsylvania, yashpant@seas.upenn.edu

Rahul Mangharam
University of Pennsylvania, rahulm@seas.upenn.edu

Robust Model Predictive Control with Anytime Estimation

Abstract

With an increasing autonomy in modern control systems comes an increasing amount of sensor data to be processed, leading to overloaded computation and communication in the systems. For example, a vision-based robot controller processes large image data from cameras at high frequency to observe the robot's state in the surrounding environment, which is used to compute control commands. In real-time control systems where large volume of data is processed for feedback control, the data-dependent state estimation can become a computation and communication bottleneck, resulting in potentially degraded control performance. Anytime algorithms, which offer a trade-off between execution time and accuracy of computation, can be leveraged in such systems. We present a Robust Model Predictive Control approach with an Anytime State Estimation Algorithm, which computes both the optimal control signal for the plant and the (time-varying) deadline/accuracy constraint for the anytime estimator. Our approach improves the system's performance (concerning both the control performance and the estimation cost) over conventional controllers, which are designed for and operate at a fixed computation time/accuracy setting. We numerically evaluate our approach in an idealized motion model for navigation with both state and control constraints.

Keywords

Anytime Algorithms, Robust Model Predictive Control

Disciplines

Computer Engineering | Electrical and Computer Engineering

Robust Model Predictive Control with Anytime Estimation

Truong X. Nghiem, Yash V. Pant and Rahul Mangharam
Department of Electrical and Systems Engineering
University of Pennsylvania
{nghiem, yashpant, rahulm}@seas.upenn.edu

***Abstract*— With an increasing autonomy in modern control systems comes an increasing amount of sensor data to be processed, leading to overloaded computation and communication in the systems. For example, a vision-based robot controller processes large image data from cameras at high frequency to observe the robot's state in the surrounding environment, which is used to compute control commands. In real-time control systems where large volume of data is processed for feedback control, the data-dependent state estimation can become a computation and communication bottleneck, resulting in potentially degraded control performance. Anytime algorithms, which offer a trade-off between execution time and accuracy of computation, can be leveraged in such systems. We present a Robust Model Predictive Control approach with an Anytime State Estimation Algorithm, which computes both the optimal control signal for the plant and the (time-varying) deadline/accuracy constraint for the anytime estimator. Our approach improves the system's performance (concerning both the control performance and the estimation cost) over conventional controllers, which are designed for and operate at a fixed computation time/accuracy setting. We numerically evaluate our approach in an idealized motion model for navigation with both state and control constraints.**

I. Introduction

Data-driven control systems, such as autonomous navigation and radar-based missile defense, generate a deluge of sensor data which often overload the real-time processing system. In such cases, the computational bottleneck is the state-estimation from sensor data (e.g. position and velocity estimates), while computing the control takes significantly less time, which is often well-bounded. A common underlying assumption in most studies on co-design of computation and control, has been that the state-estimator has a fixed sampling rate, and/or has either no/fixed computation time [1], as well has a fixed estimation error (constant upper bound or distribution) [2]. In reality, for overloaded systems with limited computation power, the computation time for estimation may be too high to neglect or too long to support a stabilizing controller.

Anytime algorithms [3], which have multiple computation time and error operating points, have a computation time/accuracy trade-off which offers much needed flexibility for overloaded systems. This is often achieved by having different implementations to perform the same function with varying levels of effort and accuracy. This paper investigates the design of robust controllers with the use of such anytime algorithms for estimation in overloaded systems. A key challenge is that when an anytime algorithm is used for estimation the static computation time/estimation error representation of an estimator is either too conservative for the setup at hand (operating at a fixed time/error setting) or is no longer applicable (if the entire operating range of the anytime algorithm is to be used).

In this paper, we present a Robust Model Predictive Controller (RMPC) based algorithm, that computes both the control signal and the operating point for the estimator, for a system where the state estimation is done by an anytime algorithm. The focus of this paper is on the RMPC algorithm and the anytime state estimator is abstracted away to its computation time/estimation error characteristics. The main contributions of this paper are:

- We treat the computation delay (and the corresponding estimation error) as a variable to be optimized over and present an algorithm which finds the best operating point for the estimator and also uses the RMPC to compute the optimal control signal.
- We take into account an anytime algorithm based estimator and its computation time (delay) vs. accuracy dependency (resulting in different operating points) and formulate a RMPC for the system. Unlike existing work on anytime control, our approach handles both state and input constraints.
- We have developed a computationally-light method for set operations specific to our problem, providing a significant speedup compared to using existing toolboxes. This allows our method to be applied to systems with many states and also have for longer horizon lengths for the optimization leading to better over all performance.

Organization: We describe the components of the system we consider in our study and the goal for our control algorithm in Section III. We then introduce our RMPC algorithm In Section IV. In Section V, we formulate the optimization problem which is central to our algorithm and also the tightened set constraints to ensure robust feasibility of the RMPC algorithm. In Section VI, we show how to explicitly compute the set constraints for the RMPC algorithm in order to reduce the computation time needed. We finally present a case study, in Section VII, where we apply our RMPC algorithm (with multiple estimator modes) to an idealized motion model for constrained navigation in a 2D plane.

This work was supported by STARnet – a Semiconductor Research Corporation program sponsored by MARCO and DARPA – and by the US Department of Transportation University Transportation Center Program.

II. Related Work

Dean and Boddy [3] introduced the term "Anytime algorithm" in the late 1980s. In [4], Horvitz et al. introduced the flexible computing model for time-critical decision making and planning algorithms in Artificial Intelligence. Anytime algorithms for sensor interpretation and path planning in more complex systems were studied in [5], [6]. Anytime algorithms have also been studied for graph search [7], evaluation of belief networks [8] and GPU architectures [9].

As overloaded real-time systems are becoming increasingly common, anytime algorithms for control have become a topic of research interest. Most notably, Quevedo and Gupta [10], Bhattacharya and Balas [1], and Fontanelli et al. [11] have contributed on the topic. In [10], the authors presented an algorithm that computes control input sequences for time steps into the future when given processor availability and use the previously computed inputs when there is no processor availability. In [11], a switching condition was developed to switch among multiple feedback controllers with different worst case execution times for a single plant. The authors in [1] proposed a methodology to get reduced order controllers with different computation requirements for a given linear time invariant (LTI) plant and a switching scheme to chose which controller to use.

Our approach differs significantly from these works as the anytime computation assumption is on the state estimator and our controller is a robust controller which can switch between different modes of the anytime estimator. Also, while most of these works require either access to the full state of the system or have a fast estimator giving them the state estimate [1], our algorithm accounts for the computation time/error of the estimation algorithm. Furthermore, an advantage of the proposed RMPC is that it can handle both state and input constraints. However, our RMPC formulation differs from related RMPC formulations [12], [2] as it can work with time-varying error bound and execution time (delay) of the state estimator.

III. System Model

The control system consists of three interconnected components as illustrated in Fig. 1:

1) The *plant* is a continuous time LTI system of the form

$$\dot{x}(t) = A_c x(t) + B_c u(t) + w_c(t) \tag{1}$$

where $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^m$ is the control input, and $w_c \in \mathbb{R}^n$ is the process noise. Though w_c is unknown, we assume that it belongs to a known compact and convex constraint set $\mathcal{W}_c \subset \mathbb{R}^n$.

2) The *estimator* observes the plant output $y(t) = Cx(t) + v(t)$ (where v is the output noise) and estimates the current state of the plant, which cannot be measured directly. Let $\hat{x}(t)$ denote the estimated state at time t and $e(t) = x(t) - \hat{x}(t)$ be the error between the actual and estimated states. Note that $e(t)$ is unknown, however, depending on the characteristics of the estimation algorithm, it would be bounded. The upper bound $\epsilon(t)$ of the norm $\|e(t)\|$, where $\|\cdot\|$ is any vector norm, determines the *accuracy* of the state estimation. The set of error vectors corresponding to an accuracy $\epsilon \geq 0$ is denoted by $\mathcal{E}(\epsilon) := \{e \in \mathbb{R}^n : \|e\| \leq \epsilon\}$. We assume that the estimator is an *anytime algorithm* where there is a trade-off between the accuracy and the computation time. In particular, the more accurate the estimation is the longer it takes to compute, and conversely. The accuracy of the estimator, hence its computation time, can be adjusted online by changing its parameters or its algorithm.

3) The *controller* computes the control input for the plant as well as adapts the estimator's accuracy to achieve a predefined control goal.

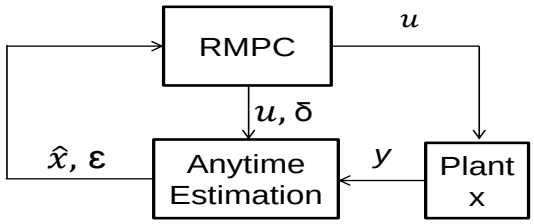

Fig. 1. Structure of the control and anytime estimation system.

We consider a discrete time implementation of the controller and the estimator, in which the output $y(t)$ is sampled periodically at instants $t_{s,k} = kT$, where $k \in \mathbb{Z}^+$ and $T > 0$ is a predefined sampling period. The sampled output is fed to the estimator which computes the state estimate $\hat{x}_k := \hat{x}(t_{s,k})$ with the desired accuracy $\epsilon_k := \epsilon(t_{s,k})$ determined by the controller in the previous time step. The controller then uses this state estimate to compute the control input u_k as well as decide on the desired state estimate's accuracy ϵ_{k+1} for the next step. Let δ_k be the worst-case total execution time of both the estimator and the controller corresponding to the accuracy ϵ_k of the state estimation at time step $t_{s,k}$. We make the following theoretical assumption of the state estimator.

Assumption 1: The estimation algorithm is given with a finite set of $p > 0$ modes (or options) $\Delta = \{(\delta_i, \epsilon_i)\}_{i=1}^{p}$; each mode corresponds to a pair of time delay and estimation accuracy. In each time step k, one of the estimation modes is selected, that is $(\delta_k, \epsilon_k) \in \Delta$.

This assumption means that in this paper we will not design nor analyze the estimation algorithm; in other words, the estimator is a black-box given to us with known characteristics.

Furthermore, the control implementation is subject to the following assumption.

Assumption 2 (Time-triggered actuation): The control actuation is delayed by δ_k, i.e., the control input computed by the controller is applied exactly at the actuation instant $t_{a,k} = t_{s,k} + \delta_k$.

The order of sensing–computing–actuating and their timing are illustrated in the diagram in Fig. 2. We remark that in each step $k \geq 0$, the estimation accuracy ϵ_k and hence the delay δ_k are already decided in the previous step and known to the controller. The previous control input u_{k-1} is still used until $t_{a,k}$ when the new control input u_k is computed and

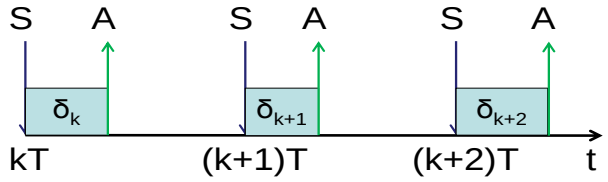

Fig. 2. Timing diagram of Control with Anytime Estimation. The symbols S and A signify the instants of (periodic) sensing and actuation respectively.

applied by the controller. The controller also chooses the next desired accuracy ϵ_{k+1} and delay δ_{k+1} to be used in the next step $k+1$. In the first step $k=0$, the initial accuracy ϵ_0, the initial delay δ_0, and the initial control input u_{-1} are chosen by the designer.

A. Discrete-time System Dynamics

The plant's state at each sampling time $t_{s,k}$ can be described by the discrete-time system:

$$x_{k+1} = Ax_k + B_1(\delta_k)u_{k-1} + B_2(\delta_k)u_k + w_k, k \geq 0 \quad (2)$$

in which

$$A = \mathrm{e}^{A_c T}, \quad w_k = \int_0^T \mathrm{e}^{A_c(T-t)} w_c(t_{s,k}+t)\,\mathrm{d}t$$

$$B_1(\delta_k) = \int_0^{\delta_k} \mathrm{e}^{A_c(T-t)} B_c\,\mathrm{d}t,\ B_2(\delta_k) = \int_{\delta_k}^{T} \mathrm{e}^{A_c(T-t)} B_c\,\mathrm{d}t.$$

Here w_k is the accumulated process noise during the interval. Because $w_c(t)$ is constrained in the compact and convex set $\mathcal{W}_c$ and T is finite, we can find a compact and convex set $\mathcal{W}$ that bounds w_k, namely

$$w_k \in \mathcal{W} \qquad \forall k \geq 0. \quad (3)$$

Note that both the current control u_k and the previous control u_{k-1} appear in Eq. (2). Furthermore, the input matrices $B_1(\delta_k)$ and $B_2(\delta_k)$ depend on the delay δ_k; hence δ_k is also an input to the dynamics. The estimation accuracy ϵ_k does not appear in the equation because it only affects the state estimate $\hat{x}_k$ used by the controller to compute u_k; therefore ϵ_k indirectly affects the dynamics via the control input.

B. State and Control Constraints

For every step $k \geq 0$, the actual state of the plant $x_k := x(t_{s,k})$ must satisfy a safety condition that

$$x_k \in \mathcal{S} \quad (4)$$

where $\mathcal{S} \subset \mathbb{R}^n$ is the set of safe states. We assume that $\mathcal{S}$ is a polytope of the form $\mathcal{S} := \{x \in \mathbb{R}^n : Hx \leq b\}$, where matrix H and vector b constitute an H-representation of $\mathcal{S}$. Note that $\mathcal{S}$ is not necessarily bounded. In addition, a control input u_k is only valid if it belongs to the predefined set of admissible control inputs $\mathcal{U} \subseteq \mathbb{R}^m$

$$u_k \in \mathcal{U} \qquad \forall k \geq 0. \quad (5)$$

The sets $\mathcal{S}$ and $\mathcal{U}$ are part of the problem statement and are either chosen by the designer or determined by physical constraints of the plant and the actuators.

C. Control Performance

The goal of the controller is two fold: it needs to maintain the state and control constraints while minimizing a cost function given by $J = \sum_{k=0}^{\infty} (\ell(x_k, u_k) + \pi(\delta_k))$, where $\ell(\cdot)$ is the stage cost function for the state and control, and $\pi(\cdot)$ is the stage cost function for the estimation and computation. These stage cost functions are chosen by the designer to achieve a desired control performance.

D. Control Problem

The control problem is stated as follows.

Problem 1: Design a feedback controller, which computes the admissible control input $u_k \in \mathcal{U}$ and the required estimation accuracy ϵ_{k+1} (equivalently the delay δ_{k+1}) based on the current state estimate $\hat{x}_k$, to minimize the cost J while maintaining the state constraint $x_k \in \mathcal{S}$ for all $k \geq 0$.

E. Notations

In the rest of this paper, we use the following notational convention. We write $x_{j\,|\,k}$ for a variable x at time step j in the RMPC optimization for time step $k \leq j$ (i.e., the prediction made at time step k of variable x at time step j). To emphasize that this variable depends on an independent variable v we write $x_{j\,|\,k}(v)$; however in cases when the dependency is implicitly understood, we only write $x_{j\,|\,k}$ for brevity. We use $\mathbb{I}_n$ to denote the identity matrix of size $n \times n$. The notation $\mathbf{0}_n$ ($\mathbf{1}_n$) represents the column vector of length n whose elements are all 0's (respectively 1's). Similarly, $\mathbf{0}_{n\times m}$ ($\mathbf{1}_{n\times m}$) is the matrix of size $n \times m$ whose elements are all 0's (1's). When the dimensions of the vectors or matrices are obvious from the context, we drop the subscripts for brevity.

IV. Robust MPC with Anytime Estimation

In this paper, we design the controller using a Robust Model Predictive Control (RMPC) approach via constraint restriction [2], [12]. In order to ensure robust safety and feasibility, the key idea of this approach is to tighten the state constraint iteratively to account for possible effect of the disturbances. As time progresses, this "robustness margin" is used in the MPC optimization with the nominal dynamics, i.e., the original dynamics where the disturbances are either removed or replaced by nominal disturbances. Because only the nominal dynamics are used, the complexity of the optimization is the same as for the nominal problem.

Since the controller only has access to the estimated state $\hat{x}$, we need to rewrite the plant's dynamics with respect to $\hat{x}$. The error between x_k and $\hat{x}_k$ is $e_k = x_k - \hat{x}_k$. At time step $k+1$ we have

$$\begin{aligned}\hat{x}_{k+1} &= x_{k+1} - e_{k+1} \\ &= Ax_k + B_1(\delta_k)u_{k-1} + B_2(\delta_k)u_k + w_k - e_{k+1},\end{aligned}$$

then, by writing $x_k = \hat{x}_k + e_k$, we obtain the dynamics

$$\hat{x}_{k+1} = A\hat{x}_k + B_1(\delta_k)u_{k-1} + B_2(\delta_k)u_k + \hat{w}_k, \quad k \geq 0 \quad (6)$$

where $\hat{w}_k = w_k + Ae_k - e_{k+1}$. The set of possible values of $\hat{w}_k$ depends on the estimation accuracy at steps k and $k+1$ and is denoted by $\widehat{\mathcal{W}}(\epsilon_k, \epsilon_{k+1})$, i.e.,

$\widehat{\mathcal{W}}(\epsilon, \epsilon') := \{w + Ae - e' : w \in \mathcal{W}, e \in \mathcal{E}(\epsilon), e' \in \mathcal{E}(\epsilon')\}$. Note that $\widehat{\mathcal{W}}(\epsilon_k, \epsilon_{k+1})$ is independent of the time step k. It can be computed as $\widehat{\mathcal{W}}(\epsilon, \epsilon') = \mathcal{W} \oplus A\mathcal{E}(\epsilon) \oplus (-\mathcal{E}(\epsilon'))$ where the symbol $\oplus$ denotes the Minkowski sum of two sets.

The dynamics in Eq. (6) has a nonstandard form where it depends on both the current and the previous control inputs. However we can expand the state variable to store the previous control input as

$$\hat{z}_k = \begin{bmatrix} \hat{x}_k \\ u_{k-1} \end{bmatrix} \in \mathbb{R}^{n+m}$$

and rewrite the dynamics as, for all $k \geq 0$,

$$\hat{z}_{k+1} = \hat{A}(\delta_k)\hat{z}_k + \hat{B}(\delta_k)u_k + \hat{F}\hat{w}_k. \tag{7}$$

Here, the system matrices are

$$\begin{aligned} \hat{A}(\delta_k) &= \begin{bmatrix} A & B_1(\delta_k) \\ \mathbf{0}_{m\times n} & \mathbf{0}_{m\times m} \end{bmatrix}, \\ \hat{B}(\delta_k) &= \begin{bmatrix} B_2(\delta_k) \\ \mathbb{I}_m \end{bmatrix}, \quad \hat{F} = \begin{bmatrix} \mathbb{I}_n \\ \mathbf{0}_{m\times n} \end{bmatrix}. \end{aligned} \tag{8}$$

Let the actual expanded state be $z_k = \left[x_k^T, u_{k-1}^T\right]^T$. Because the expanded state consists of both the plant's state and the previous control input, the state constraint $x_k \in \mathcal{S}$ and the control constraint $u_k \in \mathcal{U}$ are equivalent to the joint constraint $z_k \in \mathcal{S} \times \mathcal{U}$. We can now describe the RMPC algorithm for the dynamics in Eq. (7).

A. Tractable RMPC Algorithm

Let $N \geq 1$ be the horizon length of the RMPC optimization. Because the system matrices in the state equation (7) depend nonlinearly on the variables δ_k, the RMPC optimization is generally a mixed-integer nonlinear program, which is very hard to solve. To simplify the RMPC optimization to make it tractable, we fix the estimation mode for the entire RMPC horizon.

Let $\mathbb{P}_{\delta,\epsilon}(\hat{x}_k, \delta_k, \epsilon_k, u_{k-1})$ denote the RMPC optimization problem at step $k \geq 0$ where the current state estimate is $\hat{x}_k$, the current estimation mode is $(\delta_k, \epsilon_k) \in \Delta$, the previous control input is u_{k-1}, and the estimation mode for the entire horizon (after step k) is fixed at $(\delta, \epsilon) \in \Delta$. The specific RMPC optimization formulation will be presented in Section V. Because the estimation mode is fixed for the RMPC optimization, the RMPC cost function only needs to include the first component of J: $J_{\delta,\epsilon} = \sum_{j=k}^{k+N} \ell(x_j, u_j)$. The estimation and computation cost will be added later $J_{\delta,\epsilon}^{\text{total}} = J_{\delta,\epsilon}^{\star} + \alpha\pi(\delta)$ where $\alpha \geq 0$ is a weight specified by the designer. Since the system matrices become constant now, if the stage cost $\ell(\cdot)$ is linear or positive semidefinite quadratic, each optimization problem $\mathbb{P}_{\delta,\epsilon}(\cdot)$ is tractable and can be solved efficiently as we will show later. The RMPC algorithm with Anytime Estimation is stated in Alg. 1.

Algorithm 1 RMPC algorithm with Anytime Estimation.

1: (δ_0, ϵ_0) and u_{-1} specified by designer
2: Apply u_{-1}
3: **for** $k = 0, 1, \ldots$ **do**
4: Estimate $\hat{x}_k$ with mode (δ_k, ϵ_k)
5: **for** each $(\delta, \epsilon) \in \Delta$ **do**
6: Solve $\mathbb{P}_{\delta,\epsilon}(\hat{x}_k, \delta_k, \epsilon_k, u_{k-1})$
7: Calculate cost $J_{\delta,\epsilon}^{\text{total}} = J_{\delta,\epsilon}^{\star} + \alpha\pi(\delta)$
8: **end for**
9: $(\delta^\star, \epsilon^\star, u^\star_{k\,|\,k}) \leftarrow \arg\min_{\delta,\epsilon} J_{\delta,\epsilon}^{\text{total}}$
10: Wait until $t_{a,k}$
11: Apply control input $u_k = u^\star_{k\,|\,k}$ and estimation mode $(\delta_{k+1}, \epsilon_{k+1}) = (\delta^\star, \epsilon^\star)$
12: **end for**

V. Robust MPC Formulation

We formulate the RMPC optimization $\mathbb{P}_{\delta,\epsilon}(\hat{x}_k, \delta_k, \epsilon_k, u_{k-1})$ with respect to the nominal dynamics, which is the original dynamics in Eq. (7) but the disturbances are either removed or replaced by nominal disturbances. To ensure robust feasibility and safety, the state constraint set is tightened after each step using a candidate stabilizing state feedback control, and a terminal constraint is derived. In this RMPC formulation, we extend the approach in [2], [12]. At time step k, given $(\hat{x}_k, \delta_k, \epsilon_k, u_{k-1})$ and for a fixed (δ, ϵ), we solve the following optimization $\mathbb{P}_{\delta,\epsilon}(\hat{x}_k, \delta_k, \epsilon_k, u_{k-1})$:

$$J^{\star}_{\delta,\epsilon}\left(\hat{x}_k, \delta_k, \epsilon_k, u_{k-1}\right) = \min_{\boldsymbol{u},\boldsymbol{x}} \sum_{j=0}^{N} \ell(\bar{x}_{k+j\,|\,k}, u_{k+j\,|\,k}) \tag{9a}$$

subject to, $\forall j \in \{0, \ldots, N\}$

$$\bar{z}_{k+j+1\,|\,k} = \hat{A}(\delta_{k+j\,|\,k})\bar{z}_{k+j\,|\,k} + \hat{B}(\delta_{k+j\,|\,k})u_{k+j\,|\,k} \tag{9b}$$

$$(\delta_{k+j+1\,|\,k}, \epsilon_{k+j+1\,|\,k}) = (\delta, \epsilon), (\delta_{k\,|\,k}, \epsilon_{k\,|\,k}) = (\delta_k, \epsilon_k) \tag{9c}$$

$$\bar{x}_{k+j\,|\,k} = \begin{bmatrix} \mathbb{I}_n & \mathbf{0}_{n\times m} \end{bmatrix} \bar{z}_{k+j\,|\,k} \tag{9d}$$

$$\bar{z}_{k\,|\,k} = \left[\hat{x}_k^T, u_{k-1}^T\right]^T \tag{9e}$$

$$\bar{z}_{k+j\,|\,k} \in \mathcal{Z}_j\left(\epsilon_k, \epsilon\right) \tag{9f}$$

$$\bar{z}_{k+N+1\,|\,k} \in \mathcal{Z}_f\left(\epsilon_k, \epsilon\right) \tag{9g}$$

in which $\bar{z}$ and $\bar{x}$ are the variables of the nominal dynamics. The constraints of the optimization are explained below.

- Eq. (9b) is the nominal dynamics.
- Eq. (9c) states that the estimation mode is fixed at (δ, ϵ) except for the first time step when it is (δ_k, ϵ_k).
- Eq. (9d) extracts the nominal state $\bar{x}$ of the plant from the nominal expanded state $\bar{z}$.
- Eq. (9e) initializes the nominal expanded state at time step k by stacking the current state estimate and the previous control input.
- Eq. (9f) tightens the admissible set of the nominal expanded states by a sequence of shrinking sets.
- Eq. (9g) constrains the terminal expanded state to the terminal constraint set $\mathcal{Z}_f$.

The state constraint $\mathcal{Z}_j$: The tightened state constraint sets $\mathcal{Z}_j\left(\epsilon_k, \epsilon\right)$ are parameterized with two parameters ϵ_k and ϵ. They are defined as follows, for all $j \in \{0, \ldots, N\}$

$$\mathcal{Z}_0(\epsilon_k, \epsilon) = \mathcal{Z} \ominus \hat{F}\mathcal{E}(\epsilon_k) \tag{10a}$$

$$\mathcal{Z}_{j+1}(\epsilon_k, \epsilon) = \mathcal{Z}_j(\epsilon, \epsilon) \ominus L_j\hat{F}\widehat{\mathcal{W}}(\epsilon_k, \epsilon) \tag{10b}$$

in which the symbol $\ominus$ denotes the Pontryagin difference between two sets. The set $\mathcal{Z}$ combines the constraints for

both the plant's state and the control input: $\mathcal{Z} = \mathcal{S} \times \mathcal{U}$. The matrix L_j is the state transition matrix for the nominal dynamics in Eq. (9b) under a candidate state feedback gain $K_j(\delta)$, for $j \in \{0, \ldots, N\}$

$$L_0 = \mathbb{I} \tag{11a}$$

$$L_{j+1} = (\hat{A}(\delta) + \hat{B}(\delta)K_j(\delta))L_j \tag{11b}$$

Note that the possibly time-varying sequence $K_j(\delta)$ is designed for each choice of δ (i.e., the system matrices $\hat{A}(\delta)$ and $\hat{B}(\delta)$), hence L_j depends on δ; however we write L_j for brevity. The candidate control $K_j(\delta)$ is designed to stabilize the nominal system (9b), desirably as fast as possible so that the sets $\mathcal{Z}_j$ are shrunk as little as possible. In particular, if $K_j(\delta)$ renders the nominal system nilpotent after $M < N$ steps then $L_j = \mathbf{0}$ for all $j \geq M$, therefore $\mathcal{Z}_j(\epsilon_k, \epsilon) = \mathcal{Z}_M(\epsilon_k, \epsilon)$ for all $j > M$.

The terminal constraint $\mathcal{Z}_f$: $\mathcal{Z}_f$ is given by

$$\mathcal{Z}_f(\epsilon_k, \epsilon) = \mathcal{C}(\delta, \epsilon) \ominus L_N \hat{F} \widehat{\mathcal{W}}(\epsilon_k, \epsilon) \tag{12}$$

where $\mathcal{C}(\delta, \epsilon)$ is a robust control invariant admissible set for δ [13], i.e., there exists a feedback control law $u = \kappa(z)$ such that $\forall z \in \mathcal{C}(\delta, \epsilon)$

$$\hat{A}(\delta)z + \hat{B}(\delta)\kappa(z) + L_N \hat{F} w \in \mathcal{C}(\delta, \epsilon), \forall w \in \widehat{\mathcal{W}}(\epsilon, \epsilon) \tag{13a}$$

$$z \in \mathcal{Z}_N(\epsilon, \epsilon) \tag{13b}$$

We remark that $\mathcal{C}(\delta, \epsilon)$ does not depend on (δ_k, ϵ_k), therefore it can be computed offline for each mode (δ, ϵ).

A. Robust Feasibility

The RMPC formulation in Eq. (9), with a fixed estimation mode $(\delta, \epsilon) \in \Delta$, is designed to ensure that it is robustly feasible, as stated in Theorem 1.

Theorem 1 (Robust Feasibility of RMPC): For any estimation mode (δ, ϵ), if $\mathbb{P}_{\delta,\epsilon}(\hat{x}_{k_0}, \delta_{k_0}, \epsilon_{k_0}, u_{k_0-1})$ is feasible then the system (2) controlled by the RMPC and subjected to disturbances constrained by Eq. (3) robustly satisfies the state constraint (4) and the control input constraint (5), and all subsequent optimizations $\mathbb{P}_{\delta,\epsilon}(\hat{x}_k, \delta_k, \epsilon_k, u_{k-1})$, $\forall k > k_0$, are feasible.

Proof: See the Appendix. ∎

The control algorithm in Alg. 1, in each time step k, solves $\mathbb{P}_{\delta,\epsilon}(\hat{x}_k, \delta_k, \epsilon_k, u_{k-1})$ for each estimation mode $(\delta, \epsilon) \in \Delta$ and selects the control input u_k and the next estimation mode $(\delta_{k+1}, \epsilon_{k+1})$ corresponding to the best total cost $J^{\text{total}}_{\delta,\epsilon}$. Therefore, during the course of control, the algorithm may switch between the estimation modes in Δ depending on the system's state. Theorem 2 states that if the control algorithm Alg. 1 is feasible in its first time step then it will be robustly feasible and the state and control input constraints are also robustly satisfied.

Theorem 2: If at the initial time step there exists $(\delta, \epsilon) \in \Delta$ such that $\mathbb{P}_{\delta,\epsilon}(\hat{x}_0, \delta_0, \epsilon_0, u_{-1})$ is feasible then the system (2) controlled by Alg. 1 and subjected to disturbances constrained by Eq. (3) robustly satisfies the state constraint (4) and the control input constraint (5), and all subsequent iterations of the algorithm are feasible.

Proof: The Theorem can be proved by recursively applying Theorem 1. Indeed, suppose at time step k the algorithm is feasible and results in control input u_k and next estimation mode $(\delta_{k+1}, \epsilon_{k+1})$, then $\mathbb{P}_{\delta_{k+1},\epsilon_{k+1}}(\hat{x}_k, \delta_k, \epsilon_k, u_{k-1})$ is feasible. By Theorem 1, $u_k \in \mathcal{U}$ and at the next time step $k+1$, $x_{k+1} \in \mathcal{S}$ and $\mathbb{P}_{\delta_{k+1},\epsilon_{k+1}}(\hat{x}_{k+1}, \delta_{k+1}, \epsilon_{k+1}, u_k)$ is also feasible, hence the algorithm is feasible. Therefore, the Theorem holds by induction. ∎

VI. Implementation Details

Efficient computation of the constraint sets necessary for the RMPC formulation can be achieved for some special cases of disturbances and estimation errors as explained below.

A. Compute State Constraint Sets

If the error set $\mathcal{E}$ and the disturbance set $\mathcal{W}$ are defined by vector norms, while the other sets are polytopes in H-representation form, then it is possible to compute the Pontryagin difference in the above equations efficiently [14].

- If $\mathcal{E}(\epsilon) := \left\{ e \in \mathbb{R}^n : \|e\|_p \leq \epsilon \right\}$ for some norm $p \in \{1, 2, \infty\}$ then its support function is $h_\epsilon(\eta) = \epsilon \|\eta\|_q$ with $p^{-1} + q^{-1} = 1$. Similarly we also have $h_w(\eta)$.
- Suppose $\mathcal{Z} = \{z : H_\mathcal{Z} z \leq b_\mathcal{Z}\}$ is a polytope in H-representation form (i.e., intersection of a finite number of half-planes). The rows of $H_\mathcal{Z}$ are $H^T_{\mathcal{Z},i}$.
- The most demanding computation in computing the constraint sets of the RMPC optimization is the Pontryagin difference. If a set V has support function h_V then $\mathcal{Z} \ominus V$ can be computed as:

$$\mathcal{Z} \ominus V = \left\{ z : H_\mathcal{Z} z \leq b_\mathcal{Z} - \begin{bmatrix} h_V(H_{\mathcal{Z},1}) \\ \vdots \\ h_V(H_{\mathcal{Z},m}) \end{bmatrix} \right\}$$

- The support function $h_{\epsilon_k,\epsilon}$ for $\widehat{\mathcal{W}}(\epsilon_k, \epsilon)$ is

$$h_{\epsilon_k,\epsilon}(\eta) = h_w(\eta) + h_{\epsilon_k}\left(A^T \eta\right) + h_\epsilon(-\eta)$$

The computation of Z_j therefore involves only simple linear algebra: matrix and vector multiplications, additions and substractions, as well as vector norms. It is not difficult to see that we can write

$$\mathcal{Z}_j(\epsilon_k, \epsilon) = \{z : H_\mathcal{Z} z \leq b_\mathcal{Z} - \epsilon d_j - \epsilon_k g_j\} \tag{14}$$

where d_j and g_j are constant vectors, which depend only on δ. Therefore we can improve these computations further by pre-computing offline these vectors; then once ϵ_k is given, we can compute Z_j in real time very fast. The terminal set Z_f can be computed in the same way.

B. Compute Robust Control Invariant Set

To compute the set $\mathcal{C}(\delta, \epsilon)$ in Eq. (13):

1) Compute $\Omega = \mathcal{Z}_N(\epsilon, \epsilon)$ and $\widehat{\mathcal{W}}(\epsilon, \epsilon)$ (see above).
2) Call the Matlab Invariant Set Toolbox [15] to compute the maximal robust control invariant set $\mathcal{C}(\delta, \epsilon)$ inside Ω subject to the disturbance set $\widehat{\mathcal{W}}(\epsilon, \epsilon)$.

VII. Case study

To evaluate the performance of the proposed control scheme, we apply it to an idealized tracking and navigation problem in a 2-dimensional plane. The estimator, which is an anytime algorithm, is responsible for localization of a vehicle. The controller, which is the RMPC algorithm, is tasked with moving the vehicle to the origin while keeping it inside a safe set. The continuous-time state-space representation of the vehicle is

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{v_x} \\ \dot{v_y} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} a_x \\ a_y \end{bmatrix}.$$

Here, the states are position along x-axis (x), position along y-axis (y), velocity along x-axis (v_x), and velocity along y-axis (v_y). The control inputs are acceleration along x-axis (a_x) and acceleration along y-axis (a_y). The sampling period for discrete-time implementation of the controller and the estimator is $T = 0.02s$. For simplicity, we assume that there is no process noise.

A. The state estimator

In this study, we assume that the estimator is a vision-based system which can give estimates of the 4 states. For example, the estimator can utilize a camera looking down on the vehicle in the x–y plane, and the captured images are processed to detect and localize the vehicle as well as to estimate its velocity. There are three estimation modes ($\delta_1 = 10\text{ms}, \epsilon_1 = 0.1$), ($\delta_2 = 5\text{ms}, \epsilon_2 = 0.5$), ($\delta_3 = 2.5\text{ms}, \epsilon_3 = 1.0$), assuming a linear relation between computation time and estimation accuracy. The realization of these modes can come from the use of different camera resolutions, frame rates as well as feature tracking/estimation algorithms, which is beyond the scope of this paper. The estimation error is such that $\|e_k\|_\infty \leq \epsilon_k$ where $e_k = x_k - \hat{x}_k$. In our simulation, the state estimate $\hat{x}_k$ is generated by adding a uniformly random error e_k, bounded by ϵ_k, to the true state x_k.

B. Simulation setup and results

The controller is tasked with minimizing a ℓ_2-norm cost on the states and inputs while respecting the state and input constraints. When the origin is within the safe set of positions, this minimization results in a trajectory bringing the vehicle from an initial displacement to the origin. We chose a horizon length $N = 50$ for the RMPC and a simulation length of 400 time steps. For the scope of this study, we neglected the cost of using a particular estimation mode, i.e., $\alpha = 0$ in Alg. 1. This means that the mode selection depends solely on the control performance for the RMPC formulation for that mode. Note, the computations for the feasible set were done using the results of Section VI-A, which gave a significant speed up. Discrete time LQR design was used to select a time-invariant stabilizing feedback gain $K(\delta)$ for each estimation mode.The robust control invariant set $\mathcal{C}(\delta, \epsilon)$ was computed using the Matlab invariant set toolbox [15]. The RMPC optimization was solved in MATLAB using CVX [16], [17].

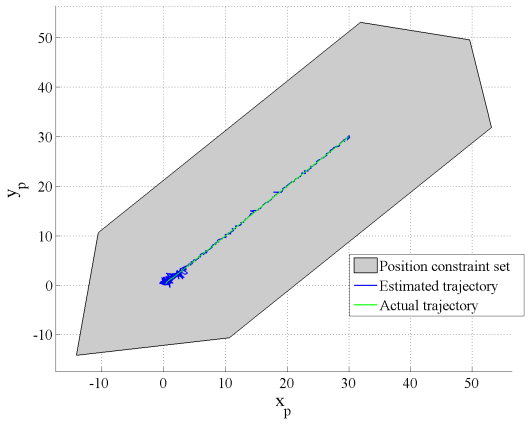

Fig. 3. Trajectory and safe set of positions.

For the simulation, the initial state was set to be $[30, 30, 0, 0]$ corresponding to initial position of $30, 30$ units in the x–y plane and zero initial velocities (in both directions). The initial inputs were also set to zero. Fig. 3 shows the estimated trajectory and the actual trajectory of the system, which stays inside the safe set of positions (the polyhedron in the figure). The velocities were constrained such that $\|v_x(t)\|_\infty \leq 100, \forall t$ and $\|v_y(t)\|_\infty \leq 100, \forall t$. The inputs were also constrained to be such that $\|a_x(t)\|_\infty \leq 100, \forall t$ and $\|a_y(t)\|_\infty \leq 10, \forall t$. The initial feasible set $\mathcal{Z}$ can be found by the Cartesian product of these three (the position, velocity and acceleration) sets. Initially, the estimator was started in the 1^{st} mode. Using Alg. 1, the optimal mode of the estimator and the corresponding Robust MPC input to the system are picked at each time step. Note that the actual trajectory is nearly a straight line from the initial point to the origin, as it would be in a noiseless and delay free system, even though the measured trajectory shows the effects of the estimation error and computation delay. This shows the robustness of the RMPC algorithm in the presence of time varying delays and estimation errors due the estimator mode switching. Fig. 4 shows the velocities (both estimated and actual). Note that as expected, the vehicle slows down as it approaches the origin and the trend continues even as estimation error increases. Fig. 5 shows the acceleration inputs to the system. Note, the main assumption behind this work is that there is a benefit from switching between different (δ, ϵ) estimation modes. Fig. 6 shows the mode selected at each time step and the switching of the modes indicates that there is indeed a cost improvement in switching modes while maintaining feasibility. This shows Alg. 1 gives better performance than a Robust MPC formulation with a fixed delay/estimation error mode.

VIII. Conclusions and future work

In this paper we presented an RMPC algorithm for a control system, with state and input constraints, where the state estimator is an anytime algorithm with multiple operation

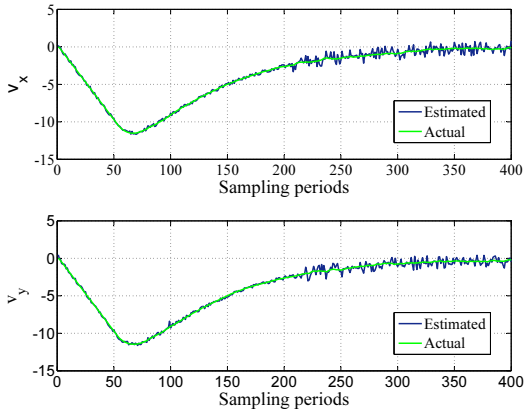

Fig. 4. Estimated and actual velocities of the vehicle.

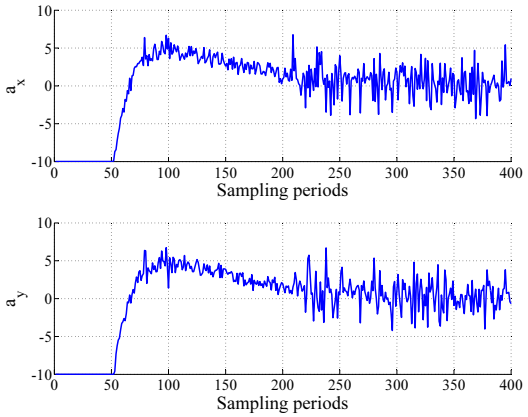

Fig. 5. Control input (acceleration) to the vehicle.

modes. The modes represent different trade-offs between computation time and estimation accuracy. The proposed RMPC algorithm computes both the control input to the plant as well as the mode for the anytime estimator. Robust feasibility and robust satisfiability of the constraints are ensured by tightening the constraints corresponding to the estimation mode, and by a terminal set based on robust control invariance. As a case study, we applied our scheme to an idealized motion model with three estimation modes and showed the benefits of switching between modes.

Ongoing research focuses on reducing the computational burden of solving the RMPC optimization for each estimation mode at each time step. This can be achieved by developing conditions that eliminate certain modes which are either infeasible at the current time step or are suboptimal. Another branch of ongoing work, which is beyond the scope of this paper, is on developing anytime estimators for vision-based control. This will enable us to apply our approach to real systems.

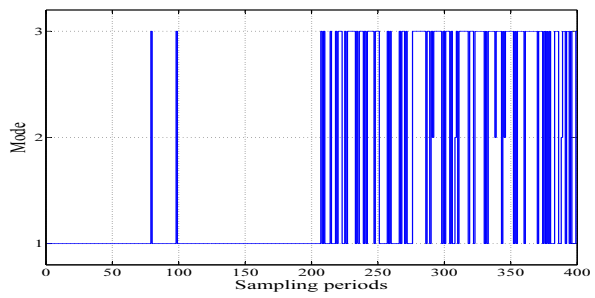

Fig. 6. Estimation mode switching of the proposed control algorithm.

APPENDIX

Proof of Theorem 1: We will prove the theorem by recursion. We will show that if at any time step k the RMPC problem $\mathbb{P}_{\delta,\epsilon}(\hat{x}_k, \delta_k, \epsilon_k, u_{k-1})$ is feasible and feasible control input $u_k = u^\star_{k\,|\,k}$ is applied with estimation mode $(\delta_{k+1}, \epsilon_{k+1}) = (\delta, \epsilon)$ then u_k is admissible and at the next time step $k+1$, the actual plant's state x_{k+1} is inside $\mathcal{S}$ and the optimization $\mathbb{P}_{\delta,\epsilon}(\hat{x}_{k+1}, \delta_{k+1}, \epsilon_{k+1}, u_k)$ is feasible for all disturbances. Then we can conclude the theorem because, by recursion, feasibility at time step k_0 implies robust constraint satisfaction and feasibility at time step k_0+1, and so on at all subsequent time steps.

Suppose $\mathbb{P}_{\delta,\epsilon}(\hat{x}_k, \delta_k, \epsilon_k, u_{k-1})$ is feasible. Then it has a feasible solution $\left(\{\bar{z}^\star_{k+j\,|\,k}\}_{j=0}^{N+1}, \{u^\star_{k+j\,|\,k}\}_{j=0}^{N}\right)$ that satisfies all the constraints in Eq. (9). Now we will construct a feasible candidate solution for $\mathbb{P}_{\delta,\epsilon}(\hat{x}_{k+1}, \delta_{k+1}, \epsilon_{k+1}, u_k)$ at the next time step by shifting the above solution by one step. Consider the following candidate solution for $\mathbb{P}_{\delta,\epsilon}(\hat{x}_{k+1}, \delta_{k+1}, \epsilon_{k+1}, u_k)$:

$$\bar{z}_{k+j+1\,|\,k+1} = \bar{z}^\star_{k+j+1\,|\,k} + L_j \hat{F} \hat{w}_k \tag{15a}$$

$$\bar{z}_{k+N+2\,|\,k+1} = \hat{A}(\delta)\, \bar{z}_{k+N+1\,|\,k+1} + \hat{B}(\delta)\, u_{k+N+1\,|\,k+1} \tag{15b}$$

$$u_{k+i+1\,|\,k+1} = u^\star_{k+i+1\,|\,k} + K_i(\delta)\, L_i \hat{F} \hat{w}_k \tag{15c}$$

$$u_{k+N+1\,|\,k+1} = \kappa\left(\bar{z}_{k+N+1\,|\,k+1}\right) \tag{15d}$$

in which $j \in \{0, \ldots, N\}$, $i \in \{0, \ldots, N-1\}$, and $\kappa(\cdot)$ is the feedback control law for the invariant set $\mathcal{C}(\delta, \epsilon)$ that is used in the terminal set in Section V. We first show that the input and state constraints are satisfied for u_k and x_{k+1}, then we will prove the feasibility of the above candidate solution for $\mathbb{P}_{\delta,\epsilon}(\hat{x}_{k+1}, \delta_{k+1}, \epsilon_{k+1}, u_k)$.

Validity of the applied input and the next state: The next plant's state is

$$\begin{aligned} x_{k+1} &= Ax_k + B_1(\delta_k)\, u_{k-1} + B_2(\delta_k)\, u_k + w_k \\ &= A(\hat{x}_k + e_k) + B_1(\delta_k)\, u_{k-1} + B_2(\delta_k)\, u^\star_{k\,|\,k} + w_k \\ &= \begin{bmatrix} A & B_1(\delta_k) \end{bmatrix} \begin{bmatrix} \hat{x}_k \\ u_{k-1} \end{bmatrix} + B_2(\delta_k)\, u^\star_{k\,|\,k} \\ &\qquad + e_{k+1} + (w_k + Ae_k - e_{k+1}) \end{aligned}$$

in which $e_{k+1} \in \mathcal{E}(\epsilon)$ and $(w_k + Ae_k - e_{k+1}) \in \widehat{\mathcal{W}}(\epsilon_k, \epsilon)$. Note that $\bar{z}^\star_{k\,|\,k} = \left[\hat{x}_k^T, u_{k-1}^T\right]^T$. Hence we have

$$\begin{aligned} \begin{bmatrix} x_{k+1} \\ u_k \end{bmatrix} &= \hat{A}(\delta_k)\bar{z}^\star_{k\,|\,k} + \hat{B}(\delta_k) u^\star_{k\,|\,k} \\ &\qquad + \hat{F} e_{k+1} + \hat{F}(w_k + Ae_k - e_{k+1}) \\ &= \bar{z}^\star_{k+1\,|\,k} + \hat{F} e_{k+1} + \hat{F}(w_k + Ae_k - e_{k+1}) \end{aligned}$$

where we use the dynamics in Eq. (9b). From Eq. (9f) and Eq. (10), $\bar{z}^{\star}_{k+1\,|\,k}$ satisfies $\bar{z}^{\star}_{k+1\,|\,k} \in \mathcal{Z}_1(\epsilon_k, \epsilon) = \mathcal{Z} \ominus \hat{F}\mathcal{E}(\epsilon) \ominus \hat{F}\widehat{\mathcal{W}}(\epsilon_k, \epsilon)$. It follows that $\left[x^T_{k+1}, u^T_k\right]^T \in \mathcal{Z} = \mathcal{S} \times \mathcal{U}$, therefore $x_{k+1} \in \mathcal{S}$ and $u_k \in \mathcal{U}$.

Initial condition: We have from Eq. (7) that $\hat{z}_{k+1} = \hat{A}(\delta_k)\hat{z}_k + \hat{B}(\delta_k)u_k + \hat{F}\hat{w}_k$. On the other hand, by Eq. (15a),

$$\begin{aligned}\bar{z}_{k+1\,|\,k+1} &= \bar{z}^{\star}_{k+1\,|\,k} + L_0\hat{F}\hat{w}_k \\ &= \hat{A}(\delta_k)\bar{z}^{\star}_{k\,|\,k} + \hat{B}(\delta_k)u^{\star}_{k\,|\,k} + L_0\hat{F}\hat{w}_k.\end{aligned}$$

Note that $\bar{z}^{\star}_{k\,|\,k} = \hat{z}_k$, $u_k = u^{\star}_{k\,|\,k}$, and $L_0 = \mathbb{I}$. Therefore $\bar{z}_{k+1\,|\,k+1} = \hat{z}_{k+1}$, hence the initial condition is satisfied.

Dynamics: We show that the candidate solution satisfies the dynamics constraint in Eq. (9b). For $0 \leq j < N$ we have

$$\begin{aligned}&\bar{z}_{k+j+2\,|\,k+1} \\ &= \bar{z}^{\star}_{k+j+2\,|\,k} + L_{j+1}\hat{F}\hat{w}_k \\ &= \hat{A}(\delta)\,\bar{z}^{\star}_{k+j+1\,|\,k} + \hat{B}(\delta)u^{\star}_{k+j+1\,|\,k} + L_{j+1}\hat{F}\hat{w}_k \\ &= \hat{A}(\delta)\left(\bar{z}_{k+j+1\,|\,k+1} - L_j\hat{F}\hat{w}_k\right) \\ &\quad + \hat{B}(\delta)\left(u_{k+j+1\,|\,k+1} - K_j(\delta)L_j\hat{F}\hat{w}_k\right) + L_{j+1}\hat{F}\hat{w}_k \\ &= \hat{A}(\delta)\,\bar{z}_{k+j+1\,|\,k+1} + \hat{B}(\delta)u_{k+j+1\,|\,k+1} \\ &\quad - \left(\hat{A}(\delta) + \hat{B}(\delta)K_j(\delta)\right)L_j\hat{F}\hat{w}_k + L_{j+1}\hat{F}\hat{w}_k \\ &= \hat{A}(\delta)\,\bar{z}_{k+j+1\,|\,k+1} + \hat{B}(\delta)u_{k+j+1\,|\,k+1}\end{aligned}$$

where the equality in Eq. (11b) is used to derive the last equality. Therefore the dynamics constraint is satisfied for all $0 \leq j < N$. For $j = N$, the constraint is satisfied by construction by Eq. (15b).

State constraints: We need to show that $\bar{z}_{(k+1)+j\,|\,k+1} \in \mathcal{Z}_j(\epsilon, \epsilon)$ for all $j \in \{0, \ldots, N\}$. Consider any $0 \leq j < N$. Eq. (10b) states that $\mathcal{Z}_{j+1}(\epsilon_k, \epsilon) = \mathcal{Z}_j(\epsilon, \epsilon) \ominus L_j\hat{F}\widehat{\mathcal{W}}(\epsilon_k, \epsilon)$. From the construction of the candidate solution we have $\bar{z}_{k+j+1\,|\,k+1} = \bar{z}^{\star}_{k+j+1\,|\,k} + L_j\hat{F}\hat{w}_k$, where $\hat{w}_k \in \widehat{\mathcal{W}}(\epsilon_k, \epsilon)$ and $\bar{z}^{\star}_{k+j+1\,|\,k} \in \mathcal{Z}_{j+1}(\epsilon_k, \epsilon)$. By definition of the Pontryagin difference, we conclude that $\bar{z}_{k+j+1\,|\,k+1} \in \mathcal{Z}_j(\epsilon, \epsilon)$ for all $j \in \{0, \ldots, N-1\}$.

At $j = N$ the candidate solution in Eq. (15a) gives us $\bar{z}_{(k+1)+N\,|\,k+1} = \bar{z}^{\star}_{k+N+1\,|\,k} + L_N\hat{F}\hat{w}_k$. Because $\bar{z}^{\star}_{k+N+1\,|\,k} \in \mathcal{Z}_f(\epsilon_k, \epsilon) = \mathcal{C}(\delta, \epsilon) \ominus L_N\hat{F}\widehat{\mathcal{W}}(\epsilon_k, \epsilon)$ and $\hat{w}_k \in \widehat{\mathcal{W}}(\epsilon_k, \epsilon)$, we have $\bar{z}_{(k+1)+N\,|\,k+1} \in \mathcal{C}(\delta, \epsilon)$. The definition of $\mathcal{C}(\delta, \epsilon)$ in Eq. (13) implies $\mathcal{C}(\delta, \epsilon) \subseteq \mathcal{Z}_N(\epsilon, \epsilon)$. Therefore $\bar{z}_{(k+1)+N\,|\,k+1} \in \mathcal{Z}_N(\epsilon, \epsilon)$.

Terminal constraint: We need to show that $\bar{z}_{k+N+2\,|\,k+1} \in \mathcal{Z}_f(\epsilon, \epsilon) = \mathcal{C}(\delta, \epsilon) \ominus L_N\hat{F}\widehat{\mathcal{W}}(\epsilon, \epsilon)$. Add $L_N\hat{F}\hat{w}$, for any $w \in \widehat{\mathcal{W}}(\epsilon, \epsilon)$, to both sides of Eq. (15b) and note that $u_{k+N+1\,|\,k+1} = \kappa\left(\bar{z}_{k+N+1\,|\,k+1}\right)$, we have

$$\begin{aligned}\bar{z}_{k+N+2\,|\,k+1} + L_N\hat{F}\hat{w} &= \hat{A}(\delta)\,\bar{z}_{k+N+1\,|\,k+1} \\ &\quad + \hat{B}(\delta)\,\kappa\left(\bar{z}_{k+N+1\,|\,k+1}\right) + L_N\hat{F}\hat{w}.\end{aligned}$$

It follows from $\bar{z}_{k+N+1\,|\,k+1} \in \mathcal{C}(\delta, \epsilon)$ and from the definition of the invariant control invariant admissible set $\mathcal{C}(\delta, \epsilon)$ (Eq. (13)) that $\bar{z}_{k+N+2\,|\,k+1} + L_N\hat{F}\hat{w} \in \mathcal{C}(\delta, \epsilon)$ for all $w \in \widehat{\mathcal{W}}(\epsilon, \epsilon)$. Then by definition of the Pontryagin difference, we conclude that $\bar{z}_{k+N+2\,|\,k+1} \in \mathcal{C}(\delta, \epsilon) \ominus L_N\hat{F}\widehat{\mathcal{W}}(\epsilon, \epsilon) = \mathcal{Z}_f(\epsilon, \epsilon)$.

REFERENCES

[1] R. Bhattacharya and G. J. Balas, "Anytime control algorithm: Model reduction approach," *Journal of Guidance and Control*, vol. 27, no. 5, pp. 767–776, 2004.

[2] A. Richards and J. How, "Robust model predictive control with imperfect information," in *American Control Conference*, 2005, pp. 268–273.

[3] M. Boddy and T. Dean, "Solving Time-dependent Planning Problems," *Joint Conf. on AI*, pp. 979–984, 1989.

[4] E. J. Horvitz, H. J. Suermondt, and G. F. Cooper, "Bounded Conditioning: Flexible inference for decision under scarce resources," *Workshop on Uncertainty in AI*, 1989.

[5] S. Zilberstein and S. Russel, "Anytime sensing, planning and action: A Practical model for robot control," *Joint Conf. on AI*, pp. 1402–1407, 1993.

[6] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[7] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun, "Anytime Search in Dynamic Graphs," *Artif. Intell.*, vol. 172, no. 14, pp. 1613–1643, 2008.

[8] M. Wellman and C. L. Liu, "State-Space Abstraction for Anytime Evaluation of Probabilistic Networks," *Conf. on Uncertainty in AI*, 1994.

[9] R. Mangharam and A. Saba, "Anytime Algorithms for GPU Architectures," in *Proc. of the IEEE Real-Time Systems Symposium*, 2011.

[10] D. Quevedo and V. Gupta, "Sequence-based anytime control," *IEEE Trans. Automat. Contr.*, vol. 58, no. 2, pp. 377–390, Feb 2013.

[11] D. Fontanelli, L. Greco, and A. Bicchi, "Anytime control algorithms for embedded real-time systems," in *Hybrid Systems: Computation and Control*. Springer, 2008, pp. 158–171.

[12] L. Chisci, J. A. Rossiter, and G. Zappa, "Systems with persistent disturbances: predictive control with restricted constraints," *Automatica*, vol. 37, no. 7, pp. 1019–1028, 2001.

[13] E. C. Kerrigan, "Robust constraint satisfaction: Invariant sets and predictive control," Ph.D. dissertation, University of Cambridge, 2000.

[14] I. Kolmanovsky and E. G. Gilbert, "Theory and computation of disturbance invariant sets for discrete-time linear systems," *Mathematical Problems in Engineering*, vol. 4, pp. 317–367, 1998.

[15] E. C. Kerrigan, "Matlab invariant set toolbox, version 0.10.5," http://www-control.eng.cam.ac.uk/eck21/matlab/invsetbox/, Apr 2005.

[16] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.0 beta," http://cvxr.com/cvx, Sept. 2013.

[17] ——, "Graph implementations for nonsmooth convex programs," in *Recent Advances in Learning and Control*. Springer-Verlag, 2008, pp. 95–110.