Computationally-Aware Switching Criteria for Hybrid Model Predictive Control Of Cyber-Physical Systems

Kun Zhang, *Student Member, IEEE,* Jonathan Sprinkle, *Senior Member, IEEE,* and Ricardo G. Sanfelice, *Senior Member, IEEE*

***Abstract*—This paper describes hybrid model predictive controllers that switch between two predictor functions based on the uncontrollable divergence metric. The uncontrollable divergence metric relates the computational capabilities of the model predictive controller, to the error of the system due to model mismatch of the predictor function during computation of the model predictive control solution. The contribution of this paper is in its treatment of the model predictive controller to permit optimization to take multiple timesteps to occur, but still rely on the uncontrollable divergence metric. The results demonstrate the approach for control of a vertical takeoff and landing aerial vehicle.**

***Note to Practitioners*—The work in this paper demonstrates how to switch between two different predictor models for a model predictive controller, based on the relative speed and accuracy of those predictor models. Unlike other papers, this work permits your optimization routine to take multiple timesteps to complete. However, it requires that the predictor models each be stable, and that you have access to the plant under control in order to measure the error between your predictor models and the plant throughout the state space.**

***Index Terms*—Primary Topics: cyber-physical systems (CPS), model predictive control (MPC). Secondary Topic Keywords: vehicle control, hybrid control.**

I. Introduction

Model Predictive Control (MPC) utilizes a predictive function to simulate the behavior of a set of control inputs over a time horizon, in order to select an input sequence that minimizes the cost according to a cost function under constraints. A sequence of control inputs and state predictions is the result of an optimization solution, obtained either through closed form or through numerical techniques. This paper is motivated by the real-time demands of a cyber-physical system (CPS) under model-predictive control. Cyber-Physical Systems are typified by the need to consider issues of computation, communication, and control when designing, analyzing, and implementing the system. Examples of systems where such an approach might be using MPC in real-time include fixed-wing aerial vehicles, and ground vehicles. Each of these vehicle types has a complex nonlinear plant, for which predictive controllers must synthesize trajectories that are feasible according to the plant, but which do not violate constraints for the safe operation of the vehicle according to its own constraints as well as the avoidance of some areas of the state space. These areas may represent “no fly zones” or obstacles that would result in collision. Thus, the vehicle (system) must synthesize and then follow trajectories to arrive at a final or intermediate destination while meeting state constraints at the final location, and satisfying other state constraints all along the trajectory.

MPC has some attractive features for systems where trajectories could be infeasible, and in which safety constraints can be specified in terms of state and control variables: (a) it handles constraints systematically as part of the cost function; and (b) it guarantees feasibility of the trajectory through the use of a predictive model that approximates the system plant over the horizon.

Limitations of the use of MPC at runtime include the need to carefully consider the timeliness of the optimization process for nonlinear plants using a nonlinear predictor function without a closed form solution [1]. Online optimization is not always guaranteed to complete in real time [2] or may return a local optimum [3]: such a limitation makes MPC unsuitable for use in systems where a safety violation would occur given such a failure. Since the use of MPC with even a single prediction function would therefore also be limited, our approach merely requires the availability of two predictor functions whose optimization in bounded time can be guaranteed.

Fundamentally, designers must trade off the accuracy of the predictor functions with the timing capabilities of the computational nodes. In terms of a suboptimal solution, or unbounded return time, approaches to mitigate these risks fall into three major categories: (i) enforcement of timing constraints if a solution has not yet been found; (ii) continued operation of the system until a solution is found, with a reduced operating envelope to ensure safe operation; or (iii) design of a predictive model that more coarsely approximates the plant model, but which guarantees (or significantly improves the guarantee) that a solution will be obtained in time.

Approach (i), enforcement of timing constraints, can be performed in several ways. Concurrent operation of multiple optimization engines could select the first feasible solution returned. A more draconian approach is to cancel the optimization routine after a certain time and proceed with the best solution found so far; such an approach faces the risk that no

K. Zhang and J. Sprinkle are with the Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ, USA e-mail: {dabiezu,sprinkjm}@email.arizona.edu.

R.G. Sanfelice is with the Department of Computer Engineering, University of California, Santa Cruz, CA, USA.

Manuscript received January 27, 2016.

feasible solution may have been obtained when the routine is halted.

Approach (ii), continued operation of the system in in degraded mode, has a few alternative solutions. One is to continue to execute the control inputs from the previous solution until a new, optimal, solution is returned. The major limitation of this approach is that the solution will be for a timestep that has already passed, and any new constraint issues (e.g., to avoid undesired regions of the system state) will not have been accounted for since the last solution. If the timestep is large relative to the dynamics of the vehicle or its disturbances, this approach could potentially result in large gaps between new control inputs. In situations such as these, the system behavior at runtime might be modified until a viable solution is returned. As an example, a vehicle might reduce its velocity until it gets a feasible solution—which might affect nonfunctional metrics such as passenger comfort.

Approach (iii), selection of a reduced-complexity predictive model, helps to provide a significant margin for robustness to potential disturbances. In some regions of the state space, this approach will execute reliably—e.g, around an operating point where linearization has occurred. Outside this region, an alternative approach is to select reduced order predictive models that are known to be simple to use when computing an optimal solution. An example of this approach is to select the kinematic (instead of dynamic) model for problems such as path planning and following. Unfortunately, as described in [4], a kinematic model diverges more quickly than the dynamic model for systems such as car-like robots, when they are used in prediction.

The two challenges that feature prominently in our expression of this approach are computational burden and predictive divergence. MPC operates based on its predictive model; model accuracy, which is normally proportional to mathematical complexity, effects computational burden such that it may endanger real-time operation.

Given the availability of a set of predictive models (of the controlled plant) with different levels of accuracy, assume that their computational delay and model mismatches can be estimated prior to online control. In [5], we introduced a trade-off metric to reveal the *uncontrollable divergence* between predicted and true states caused by return time and model mismatch. We demonstrate the effective use of this metric as the chief mechanism to select an appropriate predictive model in both [5] and [6]. The metric is obtained at design time through extensive system simulation, and the results of its calculation can be used at runtime to make decisions for the real-time controller.

Contribution of this work

The contribution of this work is in accounting for the discrete nature of the system, especially in the case that optimization may take multiple timesteps to complete. We reframe the approaches in [5] to explicitly account for the discrete sampling of the system, and the discrete times at which system control inputs can be sent. In previous work, we used an event-triggered (rather than time-triggered) approach to send new control inputs to the plant. Further, the application of this approach is applied to a different platform: namely, a vertical takeoff and landing (VTOL) vehicle.

This paper is a significant extension of a previous version published in the *2015 ACM/IEEE International Conference on Cyber-Physical Systems* [5], and a different extension of [5] under revision to *Autonomous Robots*, [6]. This paper extends [5] in significant ways since the main contribution of this work is the discrete nature of sampling and updates for the predictive controller when it is possible that solving for the optimal input sequence will require multiple timesteps, and a new platform on which we demonstrate the approach. Since multiple timesteps may be required, the process for computing the uncontrollable divergence is complicated by the need to consider the predicted state, based on a previous state's prediction of the current state. This distinction is carefully considered in this paper.

The work in [6] extended [5] primarily in the demonstration of how to calculate the uncontrollable divergence metric in a physical platform, especially when it is nontrivial to perform system data gathering in areas of the state space in which operating the system can be unsafe or uncomfortable. That paper addresses stability through tuning the dwell time parameters of the system controller, an approach we also depend on for this paper.

Organization

We begin with a review of the MPC problem, and discussion of related approaches in the literature to addressing model uncertainty, limitations of computation time for optimization, and the application of switched MPC in Section II. In order to provide a full treatment of discrete events in this paper, we provide in Section III a derivation and set of formalisms for the MPC problem that we approach, including our extensions from previous work to account for discrete inputs to the system, for variable delays in time to compute the optimal solution, as well as the definition of the uncontrollable divergence metric in terms of plant/predictive model mismatch for two predictive models. We follow in Section IV with our explicit problem statement, and in Section V demonstrate an example application to a plant model for VTOL aircraft control, including calculation of the uncontrollable divergence, and exposition of the controller switching conditions. We close the paper with a discussion of the limitations of our approach, and our goals for future work in Section VI.

II. BACKGROUND

In this section we formulate the MPC problem and describe the cost functions used, and follow it with discussions of the use of switched MPC in the literature, and the approaches and assumptions for stability of MPC in the literature.

A. Review of MPC

Model predictive control has been particularly effective in controlling plants whose trajectory must be controlled over a long horizon [7], [8]. Fundamentally, by considering the future state value over a fixed horizon of discrete points, feedback control through optimal control techniques is used to optimize cost. In [9], readers can find a thorough review of the setup of MPC. The MPC formulation used in this work is based on that of [5], [6]. The system to control is given by the nonlinear continuous-time plant

$$\dot{z} = f(z, u) \tag{1}$$

with state z and input u, where z and u are constrained to belong to the set $\mathcal{X}$ and $\mathcal{U}$, respectively. A discretization of this model by a sample time ΔT is given by $\hat{z}^+ = \hat{f}(\hat{z}, \hat{u})$, where $\hat{z}, \hat{u}$, and $\hat{f}$ are the discretizations of z, u, and f, respectively, and $\hat{z}^+$ denotes the state value after a discrete step. Let k denote discrete time: given the state $\hat{z}$ at time k, denoted as $\hat{z}_k$, the algorithms in MPC compute the evolution of the state of the discrete-time system in (1) for N steps forward in time according to input sequences of the form $\hat{U}_k = \{\hat{u}_{k,k}, \hat{u}_{k,k+1}, \cdots, \hat{u}_{k,k+N-1}\}$, where $\hat{u}_{k,k+s}$ denotes the value of the input in s discrete steps after of the initial state $\hat{z}_k$. Thus, the goal is to solve the problem[1]

$$\mathbf{P}(\hat{z}_k) : \underset{\hat{U}_k}{\text{argmin}} \; J_N(\hat{z}_k, \hat{U}_k) \tag{2}$$

where

$$J_N(\hat{z}_k, \hat{U}_k) = \sum_{t=k}^{k+N-1} \ell(\hat{z}_{k,t}, \hat{u}_{k,t}) + \varphi(\hat{z}_{k,k+N}) \tag{3}$$

is the cost function, ℓ is the stage cost function, and φ is the terminal cost function. Denoting an optimal sequence by $\hat{U}_k^*$ and the resulting optimal prediction state sequence by $\Xi_k^* = \{\hat{z}_{k,k+1}^*, \cdots, \hat{z}_{k,k+N-1}^*, \hat{z}_{k,k+N}^*\}$, MPC then applies to the discretized system the M first entries in $\hat{U}_k^*$ and then computes a new optimal sequence at discrete time $k+M+1$. The parameter $M \leq N$ determines how often to update the sequence of inputs.

B. Predictive Model Uncertainty

For any realistic physical plant, predictive model uncertainty will become a factor over a long horizon. In linear MPC, the efforts in [10], [11] consider the uncertainty error originating from linearization of a nonlinear plant, or bounded disturbance, when designing robust MPCs. These approaches (referred to as *min-max schemes*) specify as part of the cost function the uncertainty of the *maximum* possible model error and then *minimize* this cost function. Such schemes are also used in [12], [13], and although they carry a high computational burden their result is the minimum error among available linearizations.

Approaches to robustness that do not require min-max schemes include [14], where the approach uses a quadratically constrained quadratic program (QCQP) suitable for following piecewise constant references. For our work, the need to follow a smooth trajectory limits the ability to use these related approaches to mitigate uncertainty.

In cases where the plant structure is known, only the plant parameters are subject to uncertainty, the approach in [15] can be used. If the structure of the plant is uncertain, then work in [16] is relevant to quantifying the behavior of the system. When plant uncertainty can be characterized by linear matrix inequality constraints, the approach in [17] demonstrates how to quantify the performance of the predictive model.

C. Optimization Approaches and Computation Time

Regardless of the predictor function used, increasing the length of the horizon (*i.e.*, larger parameters M and N) results in a longer computation time for $\hat{U}_k^*$ and Ξ_k^*. More subtly, a nonlinear predictor dynamics (or nonlinear cost function), adversely affects computation time, as do input and state constraints.

Work to guarantee polynomial complexity of the optimization algorithm for linear plants is discussed in [13], [18]. That approach uses quadratic maximization through recursion, and is shown to be polynomial in the rank of the constraint/cost matrix: the main limitation is in its applicability only to linear systems. If only probabilistic guarantees of completion for linear systems are needed, work in [19] provides recent results.

Linearized MPC (LMPC) has the advantages over nonlinear approaches due to its relatively low computation time [20], and also that it avoids the non-convex programming common to NMPC [21]. Typically the LMPC approach (discussed in [2], [22], [23]), is performed along the previous horizon of prediction. The propagation error of the linear model from an operating point is the main limitation of LMPC for nonlinear plants. In some work the linearization is performed at runtime, which carries with it a high (but bounded) computational cost.

In this work, we assume that optimization times for a cold-start (i.e., no optimization history) of any predictive model that we could select with our hybrid controller can be bounded. If the optimization time cannot be bounded for one of the models that we use, then it would not have been used in a non-switching MPC controller.

D. Switched MPC

For a limited set of known operating points, work in [24] discusses multiple predetermined operating models for MPC predictor functions; this overcomes the limitation of runtime linearization in terms of computation time. Thus, as the state moves to a new operating region with a more accurate linearization, the predictor function can switch. The logic for switching between dynamical models naturally applies the use of finite automata (for piecewise affine systems) [25]. In that work, the ability to utilize mixed-integer quadratic programming (MIQP) as the solver is demonstrated, if the system model can be written in closed form. Mayne [9] lists the various approaches to hybrid MPC for implementation, including work by the authors in [26]. Explicit switching conditions based on transitions and guard conditions (in the case of hybrid automata) or implicit switching conditions based on state values and jump sets, are each appropriate methods through which switching can be specified.

Switching conditions play a significant role in any hybrid system's stability and performance: it is therefore necessary to craft these switching rules using tools that will enable stable

[1]

behavior. We return to this thesis later in the paper when discussing the switching criteria, and their relationship to plant dynamics and error.

III. Formulation

We extend the formalisms from Section II-A in order to distinguish solutions based on the predictor function used, the elapsed time in discrete steps for each predictor function, and the state values upon which solutions were based.

A. Predictive models

The formalisms used to distinguish our predictive models are taken after those in [6]. We define a family of discrete-time equivalent systems, which will be used in MPC as predictor functions for a nonlinear plant. Given a discrete set $\mathbb{Q} := \{0, 1, \cdots, q_{\max}\}$ and, for each $q \in \mathbb{Q}$, a function $\hat{f}^q : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$, let $\{\hat{f}^q\}_{q\in\mathbb{Q}}$ define a family of right-hand sides defining discrete-time models of (1) that are used by the predictive controller.[2] The constant $q_{\max}$ defines the number of available models for prediction. For each $q \in \mathbb{Q}$, we define $\hat{\Gamma}^q : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ as the model mismatch function

$$\hat{\Gamma}^q(\hat{z}, \hat{u}) := \hat{f}^e(\hat{z}, \hat{u}) - \hat{f}^q(\hat{z}, \hat{u}) \tag{4}$$

which captures the error between the q-th model that is available for MPC ($\hat{f}^q$) and a reference discretization of the right-hand of the system ($\hat{f}^e$). Note that $\hat{f}^e$ is not necessarily equivalent to $\hat{f}$, but is sufficiently accurate to serve as a reference.

To distinguish between the plant's states and MPC's predicted states, let subindex t represent the discrete time over the prediction horizon $N \in \{1, 2, \cdots\}$ and write $\hat{z}_{k,t}$ to denote the predicted state at time t resulting from the initial state $\hat{z}_k$ and under the effect of the planned (given) inputs $\hat{u}_{k,k}, \hat{u}_{k,k+1}, \cdots, \hat{u}_{k,t-1}$.

By convention, $\hat{z}_{k,k}$ is equivalent to $\hat{z}_k$ (*i.e.*, the predicted state at time k based on the sampled state at time k). We note that, $\hat{u}_{k,t}$ is the control input that should be selected at time t, and it is generated from the state sample at time k, viz. $\hat{z}_k$. When we write $\hat{u}_{k,k}$, it is the first element of the solution to the MPC problem in (6), with initial state $\hat{z}_k$, and it is intended to be applied at time k as the control input. In this way, for a chosen $q \in \mathbb{Q}$, the discrete-time nonlinear model used by MPC for prediction is

$$\hat{z}_{k,t+1} = \hat{f}^q(\hat{z}_{k,t}, \hat{u}_{k,t}) \tag{5}$$

for each t over the prediction horizon, i.e.,

$$t \in \{k, k+1, \cdots, k+N-1\}$$

where $\hat{z}_k$ is fed into the prediction as the initial state.

Given $\hat{z}_k$, a selected MPC solves the optimization problems $\mathbf{P}^q(\hat{z}_k)$ at time k by using the predictive model $\hat{f}^q$. By use of the superscript (i.e., $\mathbf{P}^q$) we denote the use of predictive function $q \in \mathbb{Q}$. This work denotes the input sequence

$$\{\hat{u}^q_{k,k}, \hat{u}^q_{k,k+1}, \cdots, \hat{u}^q_{k,k+N-1}\}$$

by $\hat{U}^q_k$, and formulate $\mathbf{P}^q(\hat{z}_k)$ as the following:

$$\mathbf{P}^q(\hat{z}_k) : \underset{\hat{U}^q_k \subset \mathcal{U}}{\operatorname{argmin}}\{J_N(\hat{z}_k, \hat{U}^q_k)\} \tag{6}$$

where the cost function J_N is an analog to that in (3) that is specialized through the use of the q-th prediction function as

$$J_N(\hat{z}_k, \hat{U}^q_k) = \sum_{t=1}^{N} \ell(\hat{z}^q_{k,k+t}, \hat{u}^q_{k,k+t}) + F(\hat{z}^q_{k,k+N}) \tag{7}$$

where $\ell : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^+$ is the stage cost function and $F : \mathbb{R}^n \to \mathbb{R}^+$ is the terminal cost function.

Denote the optimal solution[3] to $\mathbf{P}^q(\hat{z}_k)$ by

$$\hat{U}^{*q}_k = \{\hat{u}^{*q}_{k,k}, \hat{u}^{*q}_{k,k+1}, \cdots, \hat{u}^{*q}_{k,k+N-1}\}$$

By feeding $\hat{U}^{*q}_k$ into the predictive model $\hat{f}^q$, we obtain the optimal prediction sequence

$$\Xi^{*q}_k = \{\hat{z}^{*q}_{k,k+1}, \cdots, \hat{z}^{*q}_{k,k+N-1}, \hat{z}^{*q}_{k,k+N}\}$$

Taking the first input $\hat{u}^{*q}_{k,k}$ and applying it to the plant, we arrive at the implicit control law

$$\kappa^q(\hat{z}_k) := \hat{u}^{*q}_{k,k}$$

where $\kappa^q(\hat{z}_k)$ solves the problem in (2) using predictive model q (i.e., $\mathbf{P}^q(\hat{z}_k)$) Then, at k, the state of the plant under this control law is updated via

$$\hat{z}_{k+1} = \hat{f}^e\left(\hat{z}_k, \kappa^q(\hat{z}_k)\right)$$

At the next step, $k+1$, the hybrid controller may select a new value for q denoted $q' \in \mathbb{Q}$, or keep it constant ($q' = q$), and the problem $\mathbf{P}^{q'}(\hat{z}_{k+1})$ is solved. The algorithm continues this process indefinitely.

We let $\kappa^{\mathbb{Q}}(\hat{z})$ represent an implicit control law that selects between predictive models $q, q' \in \mathbb{Q}$ to apply either κ^q or $\kappa^{q'}$. We later define our process for selecting q at each timestep.

N.B. This approach is not intended to stabilize systems with significant model mismatch, and not intended to address issues of robustness to significant structural or parameter-based error. We require that each predictive model would result in stable behaviors if deployed on its own, since that model would otherwise not be considered a suitable prediction model for an MPC controller.

B. MPC with Computation Time

To improve the applicability of the work to practical systems, this work studies the closed-loop system considering the time delay of the solution to (6) (*i.e.*, $\mathbf{P}^q(\hat{z})$)), to permit the results to be applied to time-triggered systems.

Figure 1 provides an abstract example wherein the time required to compute the solution to (6) is larger than the

[2]The dimension of the state of each resulting discrete-time model is allowed to be different. In such a case, since the number of models is finite, one can embed all of the models into the largest space of size n by adding dummy variables. The same argument applies for the dimension of the inputs.

[3]Since we will later be selecting q to optimize another function, we use $\hat{u}^{*q}$ instead of $\hat{u}^{q*}$ to represent the optimal solution.

sampling and control input timestep; Figure 2 depicts the notation used to relate the sample times employed in the calculation of a solution to (6) and the times at which an input is provided to the physical plant. To make the use of this notation clear, we devote some effort to explaining how to represent discrete advancements in the solution, versus discrete time instances at which the state is sampled and control variables inputted.

1) Solution timesteps: Let $\Delta t^q(\hat{z})$ represent the return time of $\mathbf{P}^q(\hat{z})$ for the given state $\hat{z}$. The return time $\Delta t^q(\hat{z})$ ranges within the interval $(0, \Delta t^q_{\max}] \subset \mathbb{R}^+$, where the constant $\Delta t^q_{\max}$ is the assumed maximum return time of $\mathbf{P}^q(\hat{z})$ for all $\hat{z} \in \mathcal{X}$. Let

$$S^q(\hat{z}) := \left\lceil \frac{\Delta t^q(\hat{z})}{\Delta T} \right\rceil \in \mathbb{N}$$

represent the number of discrete sampling intervals that $\mathbf{P}^q(\hat{z})$ takes to generate a solution at $\hat{z}$ (as shown in Figure 1) and note the following inequality

$$S^q(\hat{z}) \leqslant S^q_{\max} := \left\lceil \frac{\Delta t^q_{\max}}{\Delta T} \right\rceil$$

For brevity of notation, we distinguish specific intervals and return times at different timesteps as $\Delta t^q(\hat{z}_k) = \Delta t^q_k$ and $S^q(\hat{z}_k) = S^q_k$. Note that, ΔT is not necessarily larger than $\Delta t^q_{\max}$, especially since high-fidelity models are more likely to use multiple intervals of ΔT within $\Delta t^q(\hat{z})$ seconds. If it is necessary to bound $\Delta t^q_{\max}$, especially for some highly complex model, premature termination of optimization can be deployed if sub-optimality is sufficient to ensure stability.

2) Solution timestep indices: Solutions to (6) are only drawn from solution timesteps; that is, intermediate timesteps at which a new solution is not available are not used as the initial condition for solving the MPC problem.

To identify these solution timesteps, we introduce a function to which to refer to previous and next timesteps for states and inputs.

Definition III.1 (Solution timestep)**.** *Let a* solution timestep *be a discrete timestep value $k \in \mathbb{N}^+ := \{1, 2, \ldots\}$, at which a solution to* (6) *becomes available.*

It is worth noting that due to the variances of the time $S^q(\hat{z})$, that the solution timesteps will not be regular or periodic (even for the same predictor model, $q \in \mathbb{Q}$), although they will be bounded per our assumptions.

Definition III.2 (Solution timestep series)**.** *Let the series $K = \{K_0, K_1, \ldots\}$ represent the series of solution timesteps for a given solution to the system.*

Definition III.3 ($\mathrm{pre}(\cdot)$ solution timestep)**.** *Let* $\mathrm{pre} : \mathbb{N}^+ \to K$ *be the most recent previous solution timestep $k' \in K$ for timestep k, where $k' < k$.*[4]

Definition III.4 ($\mathrm{next}(\cdot)$ solution timestep)**.** *Let* $\mathrm{next} : \mathbb{N}^+ \to K$ *be the next timestep from k at which a solution is expected.*[5]

Definition III.5 (identification whether k is a solution timestep)**.** *Let* $\mathrm{soln} : \mathbb{N}^+ \to \{0, 1\}$ *indicate whether or not k is a solution timestep, i.e.,* $\mathrm{soln}(k) = 1$ *for each $k \in K$ and* $\mathrm{soln}(k) = 0$ *if $k \in \mathbb{N}^+ \setminus K$.*

From the definitions above, $\mathrm{next(k)}$ identifies k as its $\mathrm{pre(k)}$ if and only if k is a solution timestep. Namely, $k = \mathrm{pre(next(k))} \iff \mathrm{soln}(k) = 1$.

N.B. we permit the controller $\kappa^{\mathbb{Q}}(\hat{z})$ to change modes only at solution timesteps, e.g., when $k \in K$.

Example III.1. *Take $K = \{0, 2, 3, 6, 9, 13\}$ and note that, in particular, $K_0 = 0$ and $K_3 = 6$. This set K represents an MPC controller execution where a solution was not available at time $k = 1$, but is available at time $k = 2$. We say that the solution started at $k = 0$ became available at $k = 2$. Note that* $\mathrm{pre}(2) = 0$ *and* $\mathrm{pre}(5) = 3$*, while* $\mathrm{next}(9) = 13$*. Operations compose naturally, namely,* $\mathrm{pre}(\mathrm{pre}(4)) = 2$, $\mathrm{next}(\mathrm{next}(4)) = 9$, $\mathrm{pre}(\mathrm{next}(5)) = 3$*, and, as we mentioned below Definition III.5,* $\mathrm{pre}(\mathrm{next}(6)) = 6$*. The main goal of the functions* $\mathrm{next}(\cdot)$ *and* $\mathrm{pre}(\cdot)$ *is to identify the state values at which we expect an answer (or from which we should select our control input) during execution.*

3) Considering arrival of a new solution: For any timestep $k \in \mathbb{N}^+$, it is possible that in the continuous time open interval between timesteps $k-1$ and k (i.e., $[\Delta T(k-1), \Delta T k)$) , either (a) no new solution was finalized ($\mathrm{soln}(k) = 0$), or (b) a new solution was finalized (thus, $\mathrm{soln}(k) = 1$).

Possibility (a): If in the continuous time interval $[\Delta T(k - 1), \Delta T k)$ a solution is not finalized, then the computation to produce $\hat{U}^{*q}_{\mathrm{pre(k)}}$ has not yet completed. Thus the solution we are executing became available by time $\mathrm{pre(k)}$, meaning that it was based on the state at time $\mathrm{pre(pre(k))}$: it was this timestep which produced $\hat{U}^{*q}_{\mathrm{pre(pre(k))}}$.

Possibility (b): If a new solution was finalized in the interval $[\Delta T(k - 1), \Delta T k)$, then

$$\mathrm{pre(k)} + S^q_{\mathrm{pre(k)}} = k \tag{8}$$

and a new MPC procedure based on $\hat{z}_k$ is triggered. If $\Delta t^q_{\max} > \Delta T$, it may be that more than one time step elapses during the time of calculation of the solution to (6). To make it clear that the time index is the previous solution time to k, we indicate the solution to (6) based on the state at time $\hat{z}_{\mathrm{pre(k)}}$ as

$$\hat{U}^{*q}_{\mathrm{pre(k)}} = \mathbf{P}^q(\hat{z}_{\mathrm{pre(k)}})$$

where

$$\hat{U}^{*q}_{\mathrm{pre(k)}} = \{\hat{u}^{*q}_{\mathrm{pre(k)},\mathrm{pre(k)}}, \hat{u}^{*q}_{\mathrm{pre(k)},\mathrm{pre(k)}+1}, \ldots, \hat{u}^{*q}_{\mathrm{pre(k)},\mathrm{pre(k)}+N-1}\}$$

as illustrated in Figure 2.

We know that when in mode q during the interval that the solution index $\mathrm{pre(k)}$ will take $S^q_{\mathrm{pre(k)}}$ timesteps to return.

[4]Given $k \in \mathbb{N}^+$, $\mathrm{pre(k)}$ denotes the previous solution timestep that is less than k; *i.e.*, $\mathrm{pre(k)} < k \ \forall k$. Note that $\mathrm{pre(k)}$ is not defined for $k < K_0$, $k \in \mathbb{N}^+$.

[5]Given $k \in \mathbb{N}^+$, $\mathrm{next(k)}$ denotes the least next solution timestep that is greater than k; *i.e.*, $\mathrm{next(k)} > k \ \forall k$. Note that $\mathrm{next(k)}$ is not defined for $k > \sup K$ when $\sup K < \infty$.

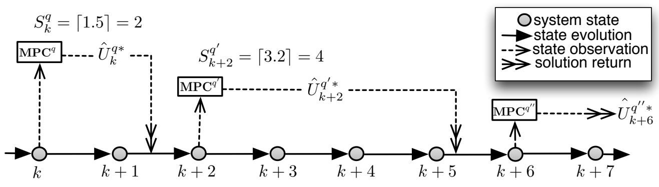


Fig. 1. Specific values are assumed to illustrate the system evolution. For example, $\hat{U}_{k+2}^{q'*}$ is generated upon the sampling instance at $k+2$ by a MPC procedure, and this procedure returns at $[\Delta T(k+5), \Delta T(k+6))$. Before the termination of this procedure, the previous solution $\hat{U}_k^{q*}$ is applied within $[\Delta T(k+2), \Delta T(k+6))$.

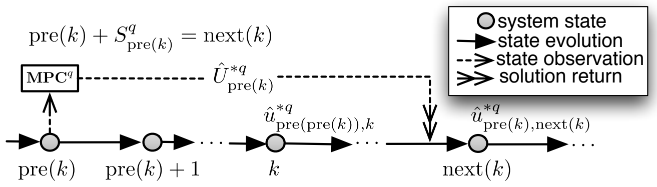

$\mathrm{pre}(k) + S_{\mathrm{pre}(k)}^q = \mathrm{next}(k)$
MPC^q
$\hat{U}_{\mathrm{pre}(k)}^{*q}$
$\hat{u}_{\mathrm{pre}(\mathrm{pre}(k)),k}^{*q}$
$\hat{u}_{\mathrm{pre}(k),\mathrm{next}(k)}^{*q}$
system state
state evolution
state observation
solution return
$\mathrm{pre}(k)$
$\mathrm{pre}(k)+1$
k
$\mathrm{next}(k)$

Fig. 2. System update process illustration.

Also, index pre(pre(k)) took $S_{\mathrm{pre(pre(k))}}^{q'}$ steps to return. Thus,

$$\mathrm{pre(k)} = \mathrm{pre(pre(k))} + S_{\mathrm{pre(pre(k))}}^{q'}$$

and as long as $\mathrm{soln}(k) = 0$,

$$\mathrm{next(k)} = \mathrm{pre(k)} + S_{\mathrm{pre(k)}}^{q}$$

so by simple substitution and rearrangement:

$$\mathrm{next(k)} - \mathrm{pre(pre(k))} = S_{\mathrm{pre(k)}}^{q} + S_{\mathrm{pre(pre(k))}}^{q'}$$

The system update function is therefore formulated as:

$$\hat{z}_{k+1} = \hat{f}^e(\hat{z}_k, \hat{u}_k) \tag{9}$$

where

$$\hat{u}_k = \begin{cases} \hat{u}_{\mathrm{pre(pre(k))},k}^{*q} & \text{if } \mathrm{soln}(k) = 0 \\ \hat{u}_{\mathrm{pre(k)},k}^{*q} & \text{if } \mathrm{soln}(k) = 1 \end{cases}$$

C. Definition of Uncontrollable Divergence (UD)

The metric UD is calculated using the error of the predictive models with respect to the plant model, and the state change that occurs while awaiting the return of the optimization function. Suppose a MPC procedure is triggered at k (*i.e.*, $k \in K$ is a solution timestep, so $\mathrm{soln}(k) = 1$), and the MPC procedure returns within $[\Delta T(\mathrm{next(k)} - 1), \Delta T\mathrm{next(k)})$, $\mathrm{next(k)} = k + S_k^q$, as shown in Figure 3. The metric UD is defined as the difference between the predicted state and the actual state after feeding the control input into the plant:

$$\mathrm{UD}^q(\hat{z}_k) := \left\| \hat{z}_{\mathrm{next(k)}} - \hat{z}_{k,\mathrm{next(k)}}^{*q} \right\| \tag{10}$$

Considering that several timesteps may elapse between k and $\mathrm{next(k)}$, we consider the uncontrollable divergence by firstly investigating an error between the reference model and the q-th predictor model within the elapsed interval $[\Delta T k, \Delta T(k+1))$. Recalling that $\hat{\Gamma}^q(\hat{z}_k, \hat{u}_k) := \hat{f}^e(\hat{z}_k, \hat{u}_k) - \hat{f}^q(\hat{z}_k, \hat{u}_k)$, we have the below:

$$\begin{aligned} & \hat{f}^e(\hat{z}_k, \hat{u}_k) - \hat{f}^q(\hat{z}_k^{*q}, \hat{u}_k^{*q}) \\ = \; & (\hat{f}^q(\hat{z}_k, \hat{u}_k) - \hat{f}^q(\hat{z}_k^{*q}, \hat{u}_k^{*q})) + \hat{\Gamma}^q(\hat{z}_k, \hat{u}_k) \\ = \; & \left(\frac{\partial \hat{f}^q}{\partial \hat{z}} |_{\hat{z}_k^{*q}, \hat{u}_k^{*q}} \right) (\hat{z}_k - \hat{z}_k^{*q}) + \\ & \left(\frac{\partial \hat{f}^q}{\partial \hat{u}} |_{\hat{z}_k^{*q}, \hat{u}_k^{*q}} \right) \left(\nabla \hat{u} |_{\hat{z}_k^{*q}, \hat{u}_k^{*q}} \right) (\hat{z}_k - \hat{z}_k^{*q}) + \hat{\Gamma}^q(\hat{z}_k, \hat{u}_k) \end{aligned}$$

we conclude that $(\hat{z}_{\mathrm{next(k)}} - \hat{z}_{k,\mathrm{next(k)}}^{*q})$ is bounded, if $\frac{\partial \hat{f}^q}{\partial \hat{z}}$, $\frac{\partial \hat{f}^q}{\partial \hat{u}}$, $\nabla \hat{u}$, $\hat{\Gamma}^q$, and S_k^q are also bounded.

D. Calculation of UD

To estimate the value of (10) for variable return times between plant predictor models at various portions of the state, off-line computation and approximation are required. The state at k can be approximated as

$$\begin{aligned} \hat{z}_k & \approx \hat{z}_{\mathrm{pre(k)},k}^{*q} + \sum_{i=0}^{S_{\mathrm{pre(k)}}^q - 1} \hat{\Gamma}^q(\hat{z}_{k,k+i}^{*q}, \hat{u}_{k,k+i}^{*q}) \\ & \approx \hat{z}_{\mathrm{pre(k)},k}^{*q} + S_{\mathrm{pre(k)}}^q \hat{\Gamma}^q(\hat{z}_k, \hat{u}_k) \end{aligned}$$

where each $\hat{u}_{k,k+i} \in \kappa^q(\hat{z}_{k,k+i}^{*q})$. By denoting $\varpi^q(\hat{z}, \tau)$ the control signal $\hat{u}$ which is synthesized from $\mathbf{P}^q(\hat{z})$ and applied at time $k+i$ (*e.g.*, $\hat{u}_{k,k+i}^{*q} = \varpi^q(\hat{z}_k, k+i)$, $i \in \{0, 1, \cdots, N-1\}$), we have

$$\hat{u}_{\mathrm{pre(k)},k+i}^{*q} = \varpi^q(\hat{z}_{\mathrm{pre(k)}}, k+i) = \varpi^q(\hat{z}_{\mathrm{pre(k)},k+i}^{*q}, k+i)$$

and then approximate the difference

$$\begin{aligned} & \hat{u}_{\mathrm{pre(k)},k+i}^{*q} - \hat{u}_{k,k+i}^{*q} \\ = \; & \varpi^q(\hat{z}_{\mathrm{pre(k)}}, k+i) - \varpi^q(\hat{z}_k, k+i) \\ = \; & \varpi^q(\hat{z}_{\mathrm{pre(k)},k}^{*q}, k+i) - \varpi^q(\hat{z}_k, k+i) \\ \approx \; & (-\nabla \varpi^q |_{\hat{z}_k, k+i})(\hat{z}_k - \hat{z}_{\mathrm{pre(k)},k}^{*q}) \\ \approx \; & (-\nabla \varpi^q |_{\hat{z}_k, k+i})(S_{\mathrm{pre(k)}}^q \hat{\Gamma}^q(\hat{z}_k, \hat{u}_k)) \end{aligned} \tag{11}$$

Note that, during each discrete timestep in the range $[\Delta T k, \Delta T \mathrm{next(k)})$, $\hat{U}_k^{*q}$ is applied to prediction, while $\hat{U}_{\mathrm{pre(k)}}^{*q}$ is utilized in the actual control. By applying (11), we obtain the following approximation

$$\begin{aligned} & \hat{z}_{k+i} - \hat{z}_{k+i-1,k+i}^{*q} \\ \approx \; & \hat{f}^e(\hat{z}_{k+i-1}, \hat{u}_{\mathrm{pre(k)},k+i-1}^{*q}) - \hat{f}^q(\hat{z}_{k+i-1}, \hat{u}_{k,k+i-1}^{*q}) \\ \approx \; & (\hat{f}^q(\hat{z}_{k+i-1}, \hat{u}_{\mathrm{pre(k)},k+i-1}^{*q}) - \hat{f}^q(\hat{z}_{k+i-1}, \hat{u}_{k,k+i-1}^{*q})) \\ & + \hat{\Gamma}^q(\hat{z}_{k+i-1}, \hat{u}_{k+i-1}) \\ \approx \; & (\frac{\partial \hat{f}^q}{\partial u} |_{\hat{z}_k, \hat{u}_{k,k}^{*q}})(\hat{u}_{\mathrm{pre(k)},k+i-1}^{*q} - \hat{u}_{k,k+i-1}^{*q}) + \hat{\Gamma}^q(\hat{z}_k, \hat{u}_k) \\ \approx \; & (\frac{\partial \hat{f}^q}{\partial \hat{u}} |_{\hat{z}_k, \hat{u}_{k,k}^{*q}})(-\nabla \varpi^q |_{\hat{z}_k, k+i-1})(S_{\mathrm{pre(k)}}^q \hat{\Gamma}^q(\hat{z}_k, \hat{u}_k)) \end{aligned}$$

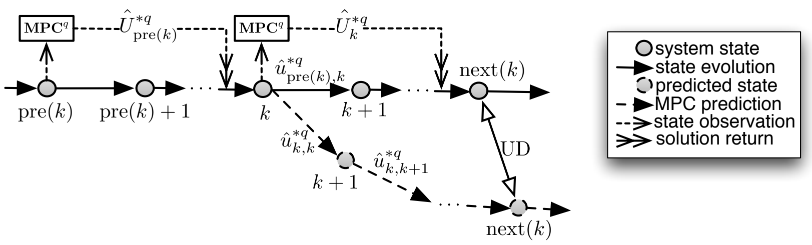


Fig. 3. Uncontrollable divergence demonstration.

and then we have

$$\begin{aligned}
& \hat{z}_{\mathrm{next(k)}} - \hat{z}^{*q}_{k,\mathrm{next(k)}} \\
\approx\ & (\hat{z}_{k+1} - \hat{z}^{*q}_{k,k+1}) + (\hat{z}_{k+2} - \hat{z}^{*q}_{k+1,k+2}) + \cdots \\
& + (\hat{z}_{\mathrm{next(k)}} - \hat{z}^{*q}_{\mathrm{next(k)}-1,\mathrm{next(k)}}) \\
\approx\ & (\frac{\partial \hat{f}^q}{\partial \hat{u}}|_{\hat{z}_k,\hat{u}^{*q}_{k,k}})(-\sum_{i=0}^{S^q_k-1} \nabla \varpi^q|_{\hat{z}_k,k+i})(S^q_{\mathrm{pre(k)}}\hat{\Gamma}^q(\hat{z}_k,\hat{u}_k))
\end{aligned} \tag{12}$$

in which the derivatives $\frac{\partial \hat{f}^q}{\partial \hat{u}}$ and $\nabla \varpi^q$ can be obtained by sensitivity analysis. Note that, the expected value or maximum value (*i.e.*, $S^q_{\max}$) can be applied to replace $S^q_{\mathrm{pre(k)}}$ and S^q_k in (12), in order to consider either the average performance, or the worst case performance when estimating UD.

IV. Problem

A. Applicable Systems

If the following assumptions are satisfied by a candidate CPS, then this work can be applied:

- The origin of the state and input space is an equilibrium point; namely, $0 \in \mathcal{X}$, $0 \in \mathcal{U}$, and $f(0,0) = 0$;
- Implicit MPC control through $\kappa^{\mathbb{Q}}(\cdot)$ will always drive the plant within the desired state space. *i.e.*, the function $\hat{f}^e(\hat{z}, \kappa^{\mathbb{Q}}(\hat{z})) \in \mathcal{X}$, is defined $\forall \hat{z} \in \mathcal{X}$ and $\hat{u} = \kappa^{\mathbb{Q}}(\hat{z}) \in \mathcal{U}$, and the solution to $\mathbf{P}^q(\hat{z})$ for each $q, q' \in \mathbb{Q}$ is defined.
- The horizon is long enough; $\forall q, q' \in \mathbb{Q}$, $S^q_{\max} + S^{q'}_{\max} \leqslant N$. This prevents the depletion of control inputs while waiting for MPC return.
- In order to ensure that UD is bounded, we require that $\frac{\partial \hat{f}^q}{\partial \hat{z}}$, $\frac{\partial \hat{f}^q}{\partial \hat{u}}$, $\nabla \hat{u}$, and $\hat{\Gamma}^q$ are bounded $\forall \hat{z} \in \mathcal{X}$, $\forall \hat{u} \in \mathcal{U}$ and $\forall q \in \mathbb{Q}$.

The introduction of the assumption that we will not run out of control inputs, requires an upper bound of the number of timesteps for each model $q, q' \in \mathbb{Q}$, in order that when switching between the two models, it cannot happen that a sufficient delay will occur such that the number of available inputs from the solution to (6) will be exhausted. Note that we can safely assume that if not switching, the inputs will not be depleted, since we assume that the MPC design under a single controller is suitable for our application. As we saw in (9) and the logic to select $\hat{u}_k$, the selected value could be from $\mathrm{pre(pre(k))}$.

In our previous work [5], [6] we demonstrated the ability to use the hybrid MPC controller on a ground vehicle, and in the remainder of this paper we turn to an alternative plant which is a vertical takeoff and landing (VTOL) fixed-wing aircraft.

B. Problem Statement

Consider a hybrid predictive controller which is synthesized according to the following principle: for a given state $\hat{z}$, select a model from the family of vehicle models $\{\hat{f}^q\}_{q\in\mathbb{Q}}$ such that the error (or divergence) between the discretized state of the plant ($\hat{z}$) and of the model ($\hat{z}_k$) obtained with the same inputs is minimized. In this controller, it is possible that solving for the control input sequence will require more than one discrete timestep.

Problem: use the uncontrollable divergence as the switching criterion for the above hybrid controller.

Analysis: The logic to select q when solving $\mathbf{P}^q(\hat{z}_k)$ should minimize the error between prediction and the actual state at $\mathrm{next(k)}$. Then, the selection of q at $\hat{z}_k$ is given by the law

$$\begin{aligned}
q &= \operatorname*{argmin}_{q\in\mathbb{Q}} \|\mathrm{UD}^q(\hat{z}_k)\| \\
&= \operatorname*{argmin}_{q\in\mathbb{Q}} \left\| \hat{z}_{\mathrm{next}(k)} - \hat{z}^{*q}_{k,\mathrm{next}(k)} \right\|
\end{aligned} \tag{13}$$

In order to permit this approach at runtime, we calculate the UD for various plant models offline, and generate lookup tables in order to solve the problem in (13) at runtime. According to work such as in [27], we utilize an average dwell-time metric, selected based on experience and by simulation, in order to allow switches to happen at any time when necessary, but constraining the average durations between switches. The dwell times we use are incorporated into the S^q for each model.

V. Example

We next use an example plant with several predictor models, to demonstrate the approach to calculate the uncontrollable divergence, and to generate the controllers. In one case, we demonstrate that an unstable model does not satisfy the necessary criteria in order to apply our approach.

A. Plant and Predictive Models

A vertical takeoff and landing (VTOL) aircraft example, as shown in Figure 4, is used to demonstrate the above approach to applying the UD to MPC at runtime when multiple timesteps are required to calculate the optimal control input sequence.

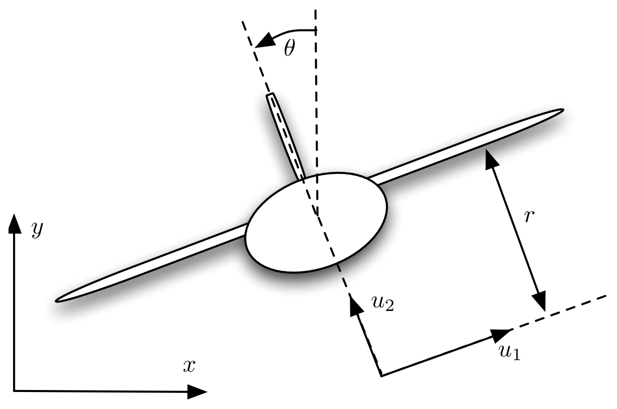


Fig. 4. Vertical takeoff and landing (VTOL) aircraft model. The forces generated by the downward thrust and transversal thrust are resolved as force u_1 and u_2 acting at a distance r below the aircraft.

1) Reference Plant: The reference plant $\hat{f}^e$ is formulated by

$$\begin{aligned}\ddot{x} &= -g\sin(\theta) + u_1\cos(\theta)/m - u_2\sin(\theta)/m - c\dot{x}/m\\ \ddot{y} &= g(\cos(\theta)-1) + u_1\sin(\theta)/m + u_2\cos(\theta)/m - c\dot{y}/m \quad (14)\\ \ddot{\theta} &= ru_1/J\end{aligned}$$

where x and y denote the position of the center of mass, θ is the orientation, m denote the mass of the aircraft, J the moment of inertia, g the gravity constant, and c the damping coefficient. As shown in Figure 4, u_1 and u_2 are resolved forces acting at a distance r below the aircraft.

2) Predictive Model 1 ($q = 1$): The first predictive model is derived from (14) by removing the last terms:

$$\begin{aligned}\ddot{x} &= -g\sin(\theta) + u_1\cos(\theta)/m - u_2\sin(\theta)/m\\ \ddot{y} &= g(\cos(\theta)-1) + u_1\sin(\theta)/m + u_2\cos(\theta)/m \qquad (15)\\ \ddot{\theta} &= ru_1/J\end{aligned}$$

3) Predictive Model 2 ($q = 2$): The second predictive model is a linearized version of (15) about the origin:

$$\begin{aligned}\ddot{x} &= u_1/m + (-g - u_2/m)\theta\\ \ddot{y} &= u_2/m + u_1\theta/m \qquad (16)\\ \ddot{\theta} &= ru_1/J\end{aligned}$$

4) Predictive Model 3 ($q = 3$): The third predictive model is reduced from (16) by removing the last terms:

$$\begin{aligned}\ddot{x} &= u_1/m\\ \ddot{y} &= u_2/m\\ \ddot{\theta} &= ru_1/J\end{aligned}$$

5) Predictive Model 4 ($q = 4$): The fourth predictive model is reduced from (16) by removing the first terms:

$$\begin{aligned}\ddot{x} &= (-g - u_2/m)\theta\\ \ddot{y} &= u_1\theta/m \qquad (17)\\ \ddot{\theta} &= ru_1/J\end{aligned}$$

B. Simulation Setup

The VTOL aircraft model is discretized using a sampling interval $\Delta T = 0.02$ sec, and it has the state and control input

$$\hat{z} = \begin{bmatrix} x & y & \theta & \dot{x} & \dot{y} & \dot{\theta}\end{bmatrix}^T$$
$$\hat{u} = \begin{bmatrix} u_1 & u_2\end{bmatrix}^T$$

where ^{T} denotes matrix transpose. The cost function in (7) is selected as

$$\begin{aligned}J_N(\hat{z}_k, \hat{U}_k^q) &= \sum_{t=0}^{N} (\hat{z}^q_{k,k+t})^T P (\hat{z}^q_{k,k+t})\\ &\quad + \sum_{t=0}^{N-1} (\hat{u}^q_{k,k+t})^T R (\hat{u}^q_{k,k+t})\end{aligned}$$

in which $N = 35$, and matrix P and R are selected as

$$P = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0\\ 0 & 10 & 0 & 0 & 0 & 0\\ 0 & 0 & 10 & 0 & 0 & 0\\ 0 & 0 & 0 & 10 & 0 & 0\\ 0 & 0 & 0 & 0 & 10 & 0\\ 0 & 0 & 0 & 0 & 0 & 10\end{bmatrix}$$
$$R = \begin{bmatrix} 0 & 0\\ 0 & 0\end{bmatrix}$$

i.e., there are no constraints associated with inputs. Aircraft parameters as set to $m = 4000$, $J = 47.6$, $r = 2.5$, $g = 9.8$ and $c = 15$. The optimization problem is described in AMPL [28] and then fed into the solver Minos [29]. Aircraft state update is simulated in MATLAB, which is used as the primary system integration environment.

C. Simulation

The simulation results are provided in order to demonstrate the feasibility of this approach for the plant models selected, if the assumptions can be met.

1) Hybrid MPC with predictive models $q = 1$ and $q = 2$: Model 1 and Model 2 are used to synthesize the hybrid MPC. According to (12), the UDs of Model 1 and Model 2 can be obtained, as shown in Figure 5. For the VTOL example, UD (or model mismatch $\hat{\Gamma}_q$) is determined by $\dot{x}$, $\dot{y}$ and θ. To represent the comparison in an intuitive way, we select specific values of θ and then plot out UDs over the plane of $\dot{x}$ and $\dot{y}$. Note that the upper limits for S_k^q, u_1 and u_2 are used when estimating UDs. By using the switching logic (13), one can easily plot out the switching boundary, as shown and explained in Figure 6. Through experimentation in simulation we set $S_{\max}^{q=1} = S_{\max}^{q=2} = 5$, based on a minimum return time of 3.49 timesteps and a maximum return time of just over 4 timesteps.

Simulations are conducted using MPC with the predictor function f^{q_1}, f^{q_2}, and a hybrid MPC using $\mathbb{Q} = \{q_1, q_2\}$. Simulation results are summarized in Figure 8, Figure 9 and Figure 10.

2) Hybrid MPC with Predictive Models $q = 3$, $q = 4$: Similar as the above procedure, a hybrid MPC can be synthe-

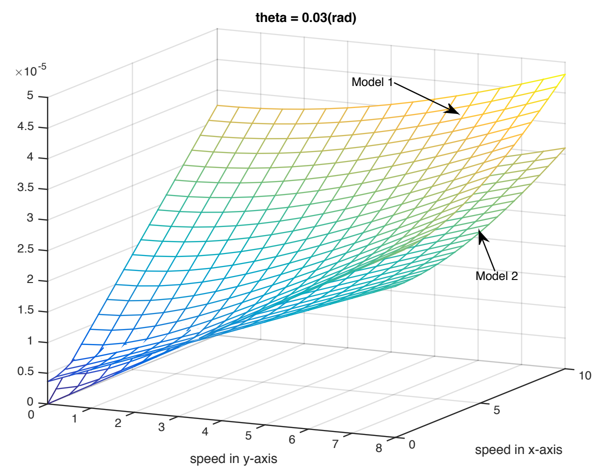


Fig. 5. Comparison of UDs for models $q = 1, q = 2$. Since the model mismatch $\hat{\Gamma}_q$ of VTOL plane is determined by $\dot{x}$, $\dot{y}$ and θ, this figure shows the condition when $\theta = 0.03$ rad.

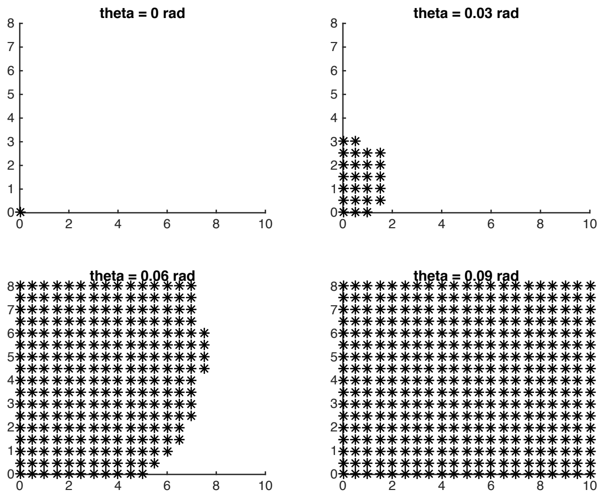


Fig. 6. Black dots represent where $q = 1$ should be applied to the MPC, the empty part is where $q = 2$ should be used. The abscissa and ordinate denote $\dot{x}$ and $\dot{y}$, respectively. These four figures show the evolution of switching boundary in slices from $\theta = 0$ rad to $\theta = 0.09$ rad.

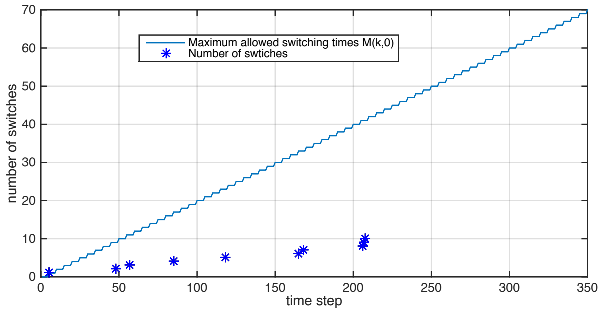


Fig. 7. Number of switches during the simulation never exceed the maximum limit $M(k, 0)$.

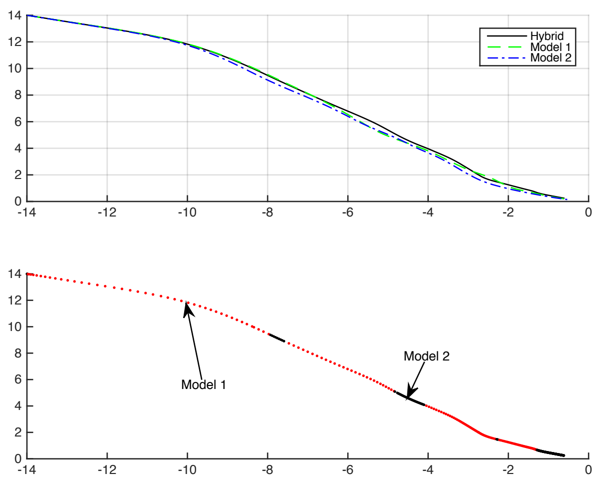


Fig. 8. The above figure plots out the airplane trajectories under the control of hybrid MPC (black solid line), Model 1-based MPC (green dashed line) and Model 2-based MPC (blue dash-dot line). The airplane moves from the initial position $(-14, 14)$ to the origin. It slowly lands and slides from the upper left to the origin. The bottom figure shows the switches that happened in hybrid MPC controlled trajectory.

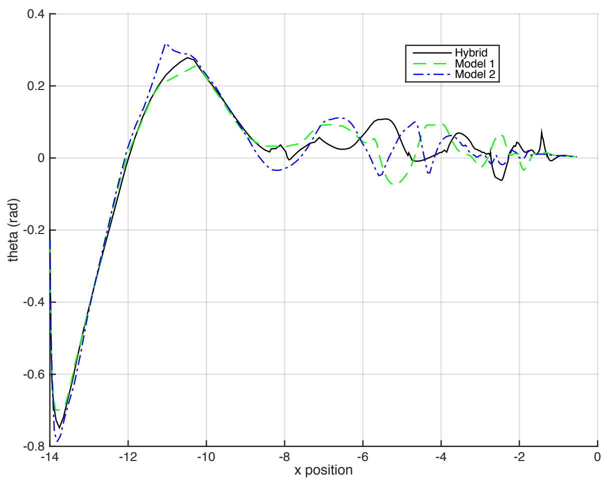


Fig. 9. Vehicle θ relative to position along the x-axis.

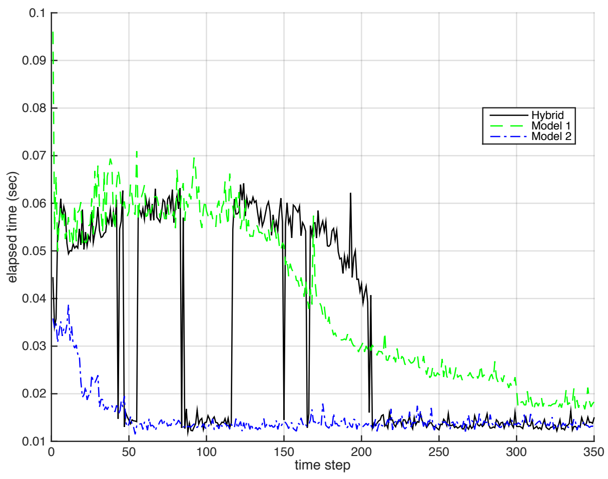


Fig. 10. Comparison of return time. The green line represents $\Delta T^{q=1}$ (return time of Model 1), and the blue line represents $\Delta T^{q=2}$. The black line is the return time of the Hybrid MPC controller.

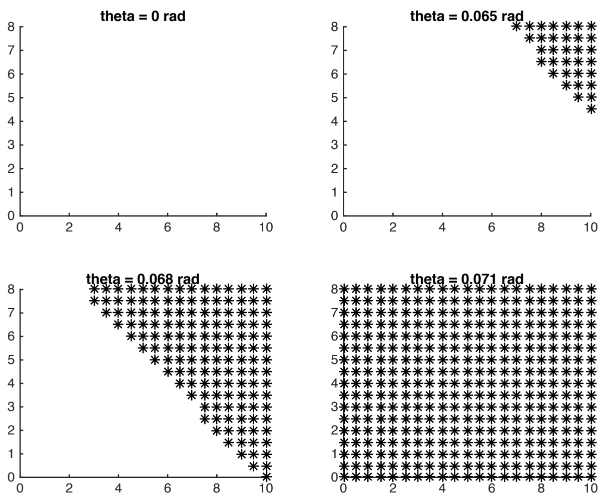


Fig. 11. Black dots represent where Model 4 should be applied to the MPC, the empty part is where Model 3 should be used. Ordinates denote $\dot{x}$ and $\dot{y}$, respectively. These four figures show the evolution of switching boundary from $\theta = 0$ rad to $\theta = 0.071$ rad.

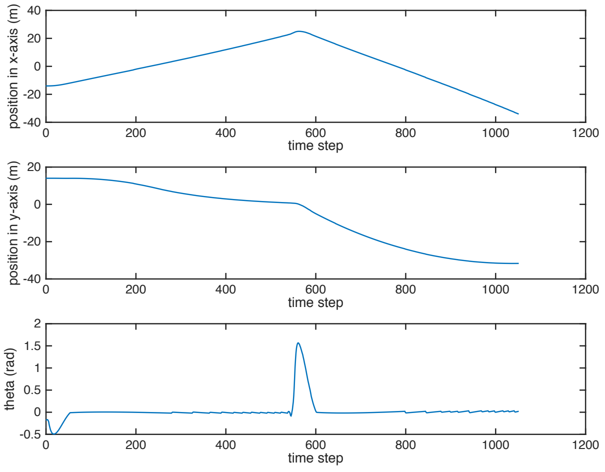


Fig. 12. The initial state of this simulation is the same as that in Figure 8. However, the position x and y move far from the origin. The simulation step is long enough to show such instability.

sized using Models 3 and 4. The switching boundary is shown in Figure 11.

In this case, we selected the structure of $q = 3$ and $q = 4$ deliberately, in order to violate our assumptions regarding applicability of the controllers to our approach. Namely, model $q = 3$ is not stable at the origin. Although it is possible to calculate the UD for this model, doing so results in selecting the model $q = 3$ at low angles near the origin, which results in instability, shown in Figure 12. Although the roll angle θ is bounded around the origin, the positional coordinates are far from the origin, and the airplane is sliding along the x-axis without decaying to the origin.

VI. Conclusion

In this paper, we described a more realistic approach to the application of uncontrollable divergence as a switching condition for CPS systems using model predictive control. The work relies on the nature of the varying dynamical models and kinematic models, in order to return results from the optimization routine faster for simpler models (which are less accurate) than dynamical models (which are more accurate). The merit of the work is in understanding how to quantify this tradeoff as a switching condition. The result is a switching condition formalism that supports shorter sampling periods, thus enabling an optimization routine to run longer than one sampling period. The examples provide demonstration of the approach to a VTOL vehicle that is controlled to the origin.

A. Limitations

The approach described in this paper is suitable *only* for systems with two predictor models, each of which is stable independently. Our approach does not include any verification requirements to ensure that unstable predictor models are used, as shown in Section V-C2. We leave for future work to develop tools that will detect these issues during calculation of the UD, and leave open the possibility that such tools may necessarily be analytic rather than computational.

Additional complexities in application of the results stem from the requirement to gather sufficient data to calculate the uncontrollable divergence with a physical plant. Since the plant may have states which are undesirable to explore for reasons of safety or comfort, and since a ground truth measurement may be difficult to obtain in all states, developers may wish to apply techniques to approximate the error functions from a sample set. One such example is provided in [6], which explores how to calculate the UD for a ground vehicle platform. That paper also describes how comparisons to a high-fidelity simulation and the physical plant can be carried out.

We obtain the UD using extensive simulation, and that we take advantage of the continuity (and stability) of the physical plant to sample its behavior to estimate the time required to optimize the control inputs. We leave as future work the approach to estimate (or bound) the optimization time at runtime for the candidate control inputs, which would not require extensive simulation.

A final note is that the UD calculations must be re-done if a new computational node is utilized to perform the optimization routine during MPC execution. This is due to the fact that the calculation of the UD depends heavily on the time required to return the solution to the MPC problem for each plant. Thus, a UD calculated for a computing platform of one type cannot be reused on another platform directly. This is analogous to changes in worst case execution time (WCET) for a real-time system: changes in the computing platform impact the WCET calculation, and can similarly impact schedulability.

B. Future work

In future work, we are interested in $\mathbb{Q}$ with $q_{\max} > 2$; since in this paper it is only possible to switch to at least one more state, implementation is much simpler. When extending to more than one possible predictor function, the work may require the specification of guard conditions, or application of other hybrid automata in the switching conditions.

When the optimization process may have a variable number of parameters for the cost function (e.g. obstacles to avoid), the optimization time may also vary. In future work we have explored how to estimate the upper bound of the execution time of the optimizer as a function of the number of obstacles over the horizon.

We are also interested in determining necessary conditions for processes and prediction functions that are suitable for the approach outlined in this paper. As presented in this work and our previous work, the switching surfaces are easy to determine and approximate as a function, but all processes and prediction functions may not satisfy these criteria.

The most compelling extension to the work regards theoretical demonstration of stability. Techniques for the demonstration of stability for constrained MPC can be found in [30]. The complexity of demonstrating stability in our approach comes from our inability to assume temporally decaying uncertainty or cost (as shown in [31], [32]). Further, because typical approaches to MPC stability use the Lyapunov function that guarantees stability as the cost function, it is not clear that such an approach will work for our system, since each predictor model will have its own cost function. For future work, we have ideas for how to extend the state of the art in order to demonstrate necessary conditions for stability for our approach. Our proposed approach is based on deriving the dwell-time parameters and the decay of the cost function from the UD metric throughout the state space. We suspect that for many MPC systems that the approach will work, but the applicability to general systems will require further study.

Acknowledgment

Work by K. Zhang and J. Sprinkle is supported in part by the National Science Foundation under award CNS-1253334. Research by R. G. Sanfelice has been partially supported by the National Science Foundation under Grant no. ECS-1150306 and by AFOSR under Grant no. FA9550-12-1-0366. The authors thank the comments of the anonymous reviewers of this work in its initial revision in the *2015 ACM/IEEE International Conference on Cyber-Physical Systems*, and the feedback and discussions as part of that conference presentation.

References

[1] M. Cannon and B. Kouvaritakis, "Continuous-time predictive control of constrained nonlinear systems," in *Nonlinear Model Predictive Control*. Springer, 2000, pp. 205–215.

[2] P. Falcone, M. Tufo, F. Borrelli, J. Asgari, and H. Tsengz, "A linear time varying model predictive control approach to the integrated vehicle dynamics control problem in autonomous systems," in *Decision and Control, 2007 46th IEEE Conference on*. IEEE, 2007, pp. 2980–2985.

[3] R. S. Parker, E. P. Gatzke, R. Mahadevan, E. S. Meadows, and F. Doyle, "Nonlinear model predictive control: issues and applications," *IEE CONTROL ENGINEERING SERIES*, pp. 33–58, 2001.

[4] M. Egerstedt, X. Hu, and A. Stotsky, "Control of a car-like robot using a dynamic model," in *ICRA*. Citeseer, 1998, pp. 3273–3278.

[5] K. Zhang, J. Sprinkle, and R. G. Sanfelice, "A hybrid model predictive controller for path planning and path following," in *International Conference on Cyber-Physical Systems (ICCPS)*. Seattle, WA: ACM, Apr. 2015, pp. 139–148. [Online]. Available: http//dx.doi.org/10.1145/2735960.2735966

[6] ——, "Computationally-aware control of autonomous vehicles: A hybrid model predictive control approach," *Autonomous Robots*, pp. 503–517, 2015. [Online]. Available: http://dx.doi.org/10.1007/s10514-015-9469-5

[7] E. F. Camacho and C. A. Bordons, *Model Predictive Control in the Process Industry*. Springer-Verlag New York, Inc., Aug. 1997.

[8] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: theory and practice—a survey," *Automatica*, vol. 25, no. 3, pp. 335–348, May 1989.

[9] D. Q. Mayne, "Model predictive control: recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967–2986, 2014.

[10] A. Richards, "Robust model predictive control for time-varying systems," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*. IEEE, 2005, pp. 3747–3752.

[11] M. Bahadorian, B. Savkovic, R. Eaton, and T. Hesketh, "Robust model predictive control for automated trajectory tracking of an unmanned ground vehicle," in *American Control Conference (ACC), 2012*. IEEE, 2012, pp. 4251–4256.

[12] T. Alamo, D. M. de la Peña, and E. F. Camacho, "Min–Max MPC based on a network problem," *Systems & Control Letters*, vol. 57, no. 2, pp. 184–192, Jan. 2008.

[13] T. Alamo, D. M. de la Pena, D. Limon, and E. F. Camacho, "Constrained min-max predictive control: modifications of the objective function leading to polynomial complexity," *Automatic Control, IEEE Transactions on*, vol. 50, no. 5, pp. 710–714, May 2005.

[14] M. N. Zeilinger, D. M. Raimondo, A. Domahidi, M. Morari, and C. N. Jones, "On real-time robust model predictive control," *Automatica*, vol. 50, no. 3, pp. 683–694, 2014.

[15] C. Løvaas, M. M. Seron, and G. C. Goodwin, "Robust output-feedback model predictive control for systems with unstructured uncertainty," *Automatica*, vol. 44, no. 8, pp. 1933–1943, 2008.

[16] P. Falugi and D. Q. Mayne, "Getting robustness against unstructured uncertainty: a tube-based MPC approach," *Automatic Control, IEEE Transactions on*, vol. 59, no. 5, pp. 1290–1295, 2014.

[17] M. V. Kothare, V. Balakrishnan, and M. Morari, "Robust constrained model predictive control using linear matrix inequalities," *Automatica*, vol. 32, no. 10, pp. 1361–1379, 1996.

[18] T. Alamo, D. M. de la Pea, and E. F. Camacho, "An Efficient Maximization Algorithm With Implications in Min-Max Predictive Control," *Automatic Control, IEEE Transactions on*, vol. 53, no. 9, pp. 2192–2197, Oct. 2008.

[19] T. Alamo, R. Tempo, and E. F. Camacho, "Randomized Strategies for Probabilistic Solutions of Uncertain Feasibility and Optimization Problems," *Automatic Control, IEEE Transactions on*, vol. 54, no. 11, pp. 2545–2559, Nov. 2009.

[20] F. Künhe, J. Gomes, and W. Fetter, "Mobile robot trajectory tracking using model predictive control," in *II IEEE latin-american robotics symposium*, 2005.

[21] M. A. Henson, "Nonlinear model predictive control: current status and future directions," *Computers and Chemical Engineering*, vol. 23, no. 2, pp. 187–202, 1998.

[22] J. B. Rawlings, "Tutorial: Model predictive control technology," in *American Control Conference, 1999. Proceedings of the 1999*, vol. 1. IEEE, 1999, pp. 662–676.

[23] F. Kuhne, W. F. Lages, and J. M. G. da Silva Jr, "Model predictive control of a mobile robot using linearization," in *Proceedings of Mechatronics and Robotics*, 2004, pp. 525–530.

[24] Z. Zhao, X. Xia, J. Wang, J. Gu, and Y. Jin, "Nonlinear dynamic matrix control based on multiple operating models," *Journal of Process Control*, vol. 13, no. 1, pp. 41–56, Jan. 2003.

[25] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, 1999.

[26] R. Goebel, R. G. Sanfelice, and A. R. Teel, *Hybrid Dynamical Systems: Modeling, Stability, and Robustness*. New Jersey: Princeton University Press, 2012.

[27] J. Hespanha and A. Morse, "Stability of switched systems with average dwell-time," in *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, vol. 3, 1999, pp. 2655–2660 vol.3.

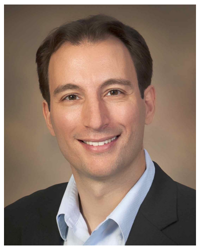
[28] R. Fourer, D. M. Gay, and B. W. Kernighan, "Ampl: A modeling language for mathematical programming," p. 540, 2002, ISBN 0-534-38809-4.

[29] B. Murtagh and M. A. Saunders, "MINOS 5.5 User's Guide," Jul. 1998. [Online]. Available: http://web.stanford.edu/group/SOL/guides/minos55.pdf

[30] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.
[31] D. L. Marruedo, T. Alamo, and E. Camacho, "Stability analysis of systems with bounded additive uncertainties based on invariant sets: Stability and feasibility of MPC," in *American Control Conference, 2002. Proceedings of the 2002*, vol. 1. IEEE, 2002, pp. 364–369.
[32] P. O. Scokaert, J. B. Rawlings, and E. S. Meadows, "Discrete-time stability with perturbations: Application to model predictive control," *Automatica*, vol. 33, no. 3, pp. 463–470, 1997.

Kun Zhang received the Ph.D. degree in Electrical and Computer Engineering at the University of Arizona in 2015. His research interests are domain-specific modeling, cyber-physical systems and software engineering.

Jonathan Sprinkle is the Litton Industries John M. Leonis Distinguished Associate Professor of Electrical and Computer Engineering at the University of Arizona. He received the B.S. degree from Tennessee Technological University in 1999, the M.S. and Ph.D. degrees from Vanderbilt University in 2000 and 2003, respectively. From 2003-2007 he was at the University of California, Berkeley as a postdoctoral scholar. He joined the University of Arizona in 2007. In 2013 he received the NSF CAREER award, and in 2009, he received the UA's Ed and Joan Biggers Faculty Support Grant for work in autonomous systems. His work has an emphasis for industry impact, and he was recognized with the UA "Catapult Award" by Tech Launch Arizona in 2014, and in 2012 his team won the NSF I-Corps Best Team award. His research interests and experience are in systems control and engineering, and he teaches courses ranging from systems modeling and control to mobile application development and software engineering.



Ricardo G. Sanfelice received the B.S. degree in Electronics Engineering from the Universidad de Mar del Plata, Buenos Aires, Argentina, in 2001, and the M.S. and Ph.D. degrees in Electrical and Computer Engineering from the University of California, Santa Barbara, CA, USA, in 2004 and 2007, respectively. In 2007 and 2008, he held postdoctoral positions at the Laboratory for Information and Decision Systems at the Massachusetts Institute of Technology and at the Centre Automatique et Systmes at the cole de Mines de Paris. In 2009, he joined the faculty of the Department of Aerospace and Mechanical Engineering at the University of Arizona, Tucson, AZ, USA, where he was an Assistant Professor. In 2014, he joined the faculty of the Computer Engineering Department, University of California, Santa Cruz, CA, USA, where he is currently an Associate Professor. Prof. Sanfelice is the recipient of the 2013 SIAM Control and Systems Theory Prize, the National Science Foundation CAREER award, the Air Force Young Investigator Research Award, and the 2010 IEEE Control Systems Magazine Outstanding Paper Award. His research interests are in modeling, stability, robust control, observer design, and simulation of nonlinear and hybrid systems with applications to power systems, aerospace, and biology.