Nonlinear Hierarchical MPC With Application to Aircraft Fuel Thermal Management Systems

Daniel D. Leister, *Graduate Student Member, IEEE*, and Justin P. Koeln, *Member, IEEE*

***Abstract*— A nonlinear hierarchical model predictive control (MPC) framework is proposed and applied to maximize the thermal endurance of aircraft. Effectively controlling the fuel temperatures in a nonlinear multitimescale aircraft fuel thermal management system (FTMS) requires controllers capable of long-term planning and fast update rates. In this article, a two-level hierarchical MPC controller is formulated using successive linearization (SL) that directly accounts for the multitimescale and nonlinear system dynamics to achieve accurate predictive capabilities and computational efficiency. Detailed simulation results show that the proposed hierarchical structure can increase aircraft thermal endurance by at least 21% compared to a centralized approach while significantly reducing the computational cost. The results also show that SL provides a valuable framework for efficiently accounting for nonlinear system dynamics within both levels of the hierarchical MPC formulation.**

***Index Terms*— Fuel thermal management system (FTMS), hierarchical model predictive control (MPC), successive linearization (SL).**

I. Introduction

THE fuel thermal management system (FTMS) plays a major role in the overall thermal energy management of high-performance aircraft by using fuel to: 1) absorb heat from multiple heat sources; 2) remove heat from the aircraft through the combustion of fuel; and 3) store excess heat in the fuel tanks. When the total heat load exceeds the heat rejection capabilities of the FTMS, the temperature of the fuel in the fuel tanks increases and can eventually reach a limit where it is no longer safe to operate the aircraft. The *thermal endurance* of an aircraft refers to the period in which an aircraft can operate before reaching this unsafe fuel temperature limit. A properly designed FTMS controller can maximize the aircraft's thermal endurance, avoiding early temperature violations due to poor control decisions.

Previous research has shown that FTMS operating decisions at the beginning of a mission can significantly affect the thermal endurance of the aircraft [1], [2], and therefore, an effective FTMS controller requires a long planning horizon.

Manuscript received 18 March 2022; revised 21 August 2022; accepted 4 October 2022. Date of publication 27 October 2022; date of current version 25 April 2023. This work was supported by the Office of Naval Research under Award N00014-22-1-2247. Recommended by Associate Editor V. Adetola. *(Corresponding author: Justin P. Koeln.)*

The authors are with the Mechanical Engineering Department, The University of Texas at Dallas, Richardson, TX 75080 USA (e-mail: daniel.leister@utdallas.edu; justin.koeln@utdallas.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCST.2022.3214730.

Digital Object Identifier 10.1109/TCST.2022.3214730

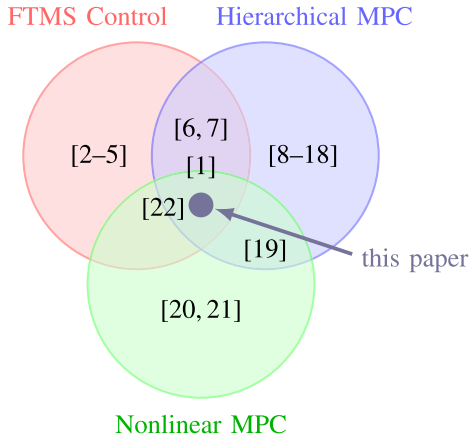

Fig. 1. Placement of the existing literature with respect to major areas relevant to this work.

However, long-term planning may not be effective if the prediction model does not accurately capture the nonlinear dynamics of the system. As shown in [1], the use of a linearized model over a long prediction horizon can result in large prediction errors and reduced thermal endurance. Moreover, when considering the dynamics of fast states or high-frequency disturbances, the controller must simultaneously have a long planning horizon and a fast control update rate. Therefore, this work draws from results in three major areas, as shown in Fig. 1, to achieve a control strategy that can handle these requirements properly.

Capable of predicting state trajectories and directly imposing operational constraints, model predictive control (MPC) is well suited to maximizing thermal endurance subject to the predicted heat loads of the aircraft while accounting for the actuator and state constraints in the FTMS. However, the combination of long prediction horizons, fast control updates, and nonlinear system dynamics prevents most centralized MPC formulations from being implemented in real time. Hierarchical MPC [8] provides a good alternative to handle the multitimescale nature of the problem and has long been developed and implemented in diverse applications, such as control of microgrids [23], path planning [24], [25], and actuator controls [11], [26]. However, this article primarily considers hierarchical MPC approaches where MPC is used at multiple levels of the hierarchy, making coordination between MPC controllers critical. Hierarchical control with multiple levels of MPC controllers has been investigated for applications such as coordination of electricity generators [12], energy

management in electric or hybrid vehicles [18], and thermal management [27], in particular for aircraft [6], [7]. It should be noted that the vast majority of existing works in theoretical hierarchical MPC formulations are limited to linear system models [7], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18] to provide guarantees such as recursive feasibility [10], [12], [13] and stability [11], [13], [14], [15] and, thus, are not directly applicable to this work considering nonlinear system models.

One of the most distinguishing features among hierarchical MPC approaches is the coordination mechanism used to ensure coherent action between the upper and lower levels of the hierarchy. A common choice is to have the upper level controller compute reference trajectories to be tracked by the lower level controllers. In [11], [13], [14], and [17], the lower level deals with actuator dynamics, and therefore, the upper level computes reference input trajectories that are states to be tracked by the lower level. The work in [15] differs slightly from this paradigm in that the upper level sends to the lower level a piecewise constant reference signal that is part of the lower level control input, augmented with the output of a stabilizing linear controller. In [6] and [7], the upper level MPC computes reference *state* trajectories to be tracked by a lower level MPC. The reference tracking mechanism is changed to *waypoint* tracking in [10], where the optimal state computed by the upper level for its next time step is used by the lower level MPC as a terminal state constraint. The rationale behind this approach is to give the lower level more freedom to optimize the short-term operation of the system as opposed to the full reference tracking mechanism. This rationale is extended in [9] by replacing the waypoints with *waysets* as terminal constraints, which are computed online. Here, the system is guaranteed to have a feasible trajectory following the upper level reference trajectory if the lower level terminal states lie within these waysets. This motivated the work in [16], where each point in the wayset has an associated terminal cost, efficiently computed online, which removed some greedy behavior from the lower level controller seen in [9].

Nonlinear MPC (NMPC) [28] is distinguished from linear MPC (LMPC) by directly considering the nonlinear system dynamics in the optimization problem formulation and is mainly used in cases where an LMPCdoes not provide satisfactory performance due to strong nonlinearities in the system dynamics. The resulting nonlinear programs (NLPs) are harder to solve compared to linear programs (LPs) or quadratic programs (QPs) typically found in LMPC, for which very efficient solvers exist. Therefore, NMPCalgorithms can be classified according to the numerical optimization strategy used to solve the underlying NLPs. In the so-called sequential approach, only the control input trajectory is discretized, and the states are obtained by simulation of the discrete nonlinear system dynamics. Since the size of the resulting NLP is considerably reduced, off-the-shelf NLP solvers, such as Interior Point OPTimizer (IPOPT) [29], often can be used [20]. However, such algorithms can behave poorly for unstable or highly nonlinear systems and do not scale well with the prediction horizon and the number of inputs [30]. Conversely, the simultaneous approach, where both the state and input trajectories are part of the decision variables in the NLP, tends to generate much larger and sparser NLPs but can also handle unstable or highly nonlinear systems. This approach also allows algorithms to leverage the structure of the resulting NLPs. As pointed out in [20], numerous techniques developed in the last decades, which explore the structure of the resulting NLPs, have been essential to the improvement of the speed and accuracy of NMPCalgorithms, so as to enable their deployment in real time. However, the implementation of many of these techniques is not trivial, which is evidenced by the various software toolkits developed by the research community to facilitate the use of efficient NMPCmethods [21].

In searching for fast NMPCalgorithms that balance implementation complexity, ease of use, and computational speed, this article explores the use of successive linearization (SL) as an alternative method for the efficient solution of NLPs. SL has found successful application in optimal control problems in aerospace applications [31], [32], [33], [34], which often involves the solution of problems with nonlinear dynamics and nonconvex constraints in real time. The reader is referred to [35] for a thorough review of SL concepts and algorithms. One advantage of SL over some of the existing efficient NMPCimplementations is that SL is relatively simple to implement and requires only the use of an efficient LP or QP solver.

As for the existing literature on FTMS control, the work in [2] is particularly relevant for the careful thermodynamic analysis of the potential advantage of considering a dual-tank system, showing how this topology can extend thermal endurance compared to a single-tank topology. The proposed linear quadratic regulator (LQR) relies on the realization that keeping the temperature of the fuel sent to the engine as close as possible to the upper fuel temperature limit maximizes heat rejection and, therefore, maximizes thermal endurance. A more recent work [5] considered a dual tank topology similar to the one adopted in this work, with an additional valve that allows recirculation fuel to bypass the recirculation tank and be directly mixed with the fuel from the reservoir in the feed line. Huang et al. [5] propose a controller that switches between four configurations optimized for specific plant conditions, aided by PI controllers to accommodate for uncertainties. The controllers in [2] and [5] have the advantage of being less computationally expensive than MPC controllers. However, in [3], it was shown how using LMPCcontrollers supplied with a preview of future disturbances can make the controllers act proactively and extend the thermal endurance, such as precooling the fuel tank when a future increase in heat load is expected, even if the preview is not perfect.

It is also important to note the contributions of works that lie at the intersection of the major areas in Fig. 1. Huang and Doman [4] extend the work in [2] by showing the advantages of using a nonlinear optimal control formulation over the LQR controller. Similar to this work, El Chamie and Lin [22] used SL for NMPCof an aircraft FTMS but considered a single-tank topology. Though the controller in [22] can be considered as part of a hierarchy, only the lowest level is considered, while, in this work, the complete hierarchy

is considered, i.e., the upper and lower level controllers. In addition, the two-tank FTMS model used in this work, based on the model from [2], considers states in multiple time scales, which adds to the complexity of the optimization problems being solved. It should be noted though that El Chamie and Lin [22] considered a time-varying mass flow rate of fuel sent to the engine, whereas this work considers a constant flow rate. Pangborn et al. [6] used linear hierarchical MPC for coordinated control of several aircraft subsystems including the FTMS. The authors show that the hierarchical controller achieves far fewer constraint violations compared to PI controllers while needing less control effort. Fewer constraint violations were also reported in a similar work [7], which includes the control of electrical subsystems, adding to the multitimescale nature of the problem, and proposes a modified MPC architecture, where the nonlinear system dynamics for the upper level is replaced by a switched linear system. The work in [19] is also closely related to this work, where a two-level nonlinear hierarchical MPC was used for the battery thermal management of an electric vehicle. The focus of that study though was not to explore the use of more efficient solution alternatives for the resulting NLPs, and therefore, a general-purpose NLP solver, IPOPT [29], was used.

The review of existing work in the major areas shown in Fig. 1 shows that there is room for improvement of FTMS control strategies for maximization of thermal endurance. Given the requirements for fast update rates, long prediction horizons, and systems constraints, hierarchical MPC is particularly well suited for this task. However, the use of linearized system models limits the ability to maximize closed-loop control performance. This work seeks to fill this gap by proposing the use of NMPCat both levels and using an SL algorithm to solve the related NLPs efficiently. This work is an extension of [1], where the lower level controller considered linearized dynamics. In addition, a more complete model of the FTMS is used, which includes additional states and an actuator. The new model contains a fast state, which adds to the multitimescale nature of the problem, reinforcing the need for a hierarchical control structure.

Therefore, the main contributions of this article are: 1) presenting a fully nonlinear hierarchical MPC framework for systems with multiple timescales that uses SL to compute solutions to the underlying NLPs in real time and 2) demonstrating that the aircraft FTMS thermal endurance maximization problem can be solved in real time using a controller architecture that takes into account the full nonlinear behavior of the system model, i.e., without resorting to linearized models. The proposed control formulation is applied to an aircraft FTMS, but the authors believe that the same concept can be applied to other control applications with timescale separation and nonlinear dynamics, such as the control of microgrids, coordination of electricity generators, and energy management in electric or hybrid vehicles.

The remainder of this article is organized as follows. Section II introduces the FTMS model. Section III presents the proposed nonlinear hierarchical MPC and other controller formulations considered in this article. Section IV details the SL approach used for nonlinear optimization. Section V presents numerical results comparing the proposed hierarchical control framework with the alternative approaches. Finally, conclusions and directions for future work are provided in Section VI.

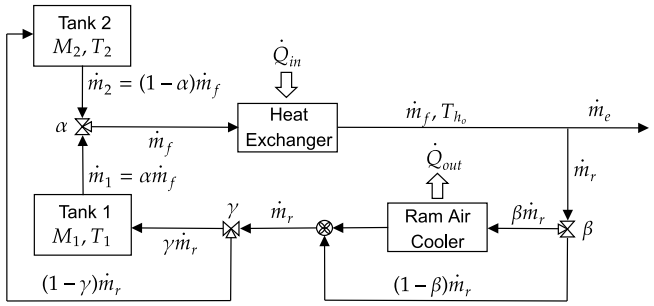

Fig. 2. Dual-tank FTMS architecture modified from [2].

Notations: Subscript k is used for indexing times steps within the prediction horizon of the optimization problems. Superscript j refers to the value of a variable at the jth iteration of the SL algorithm. When not followed by a subscript, N refers generically to the prediction horizon of an MPC controller, as applicable to the context where it appears. Likewise, Δt without a subscript refers generically to the sampling time used in a controller formulation. The variable $\mathbf{X} = \{x_0, x_1, \ldots, x_{N+1}\}$, with $x_k \in \mathbb{R}^n$, refers to a state trajectory, $\mathbf{U} = \{u_0, u_1, \ldots, u_N\}$, with $u_k \in \mathbb{R}^{n_u}$, to a sequence of inputs, $\mathbf{D} = \{d_0, d_1, \ldots, d_N\}$, with $d_k \in \mathbb{R}$, to a disturbance trajectory, and $\mathbf{S} = \{s_0, s_1, \ldots, s_N\}$, with $s_k \in \mathbb{R}^n$, to a slack variables trajectory. The variables n and n_u refer to the state and input dimensions, respectively.

II. Fuel Thermal Management System Model

While different FTMS architectures have been studied, each system: 1) uses pumped fuel flow to collect thermal energy from various heat sources; 2) removes part of the thermal energy from the aircraft by burning a portion of the heated fuel in the aircraft's engine(s); and 3) can cool some of the remaining fuel flow before returning to the fuel tank(s), which serve as thermal energy storage. The specific FTMS architecture used in this article is shown in Fig. 2 and has similar dynamic behavior to the dual-tank system from [2]. In this architecture, fuel drawn from a recirculation tank (Tank 1) and a reservoir tank (Tank 2) is sent to the feed line and absorbs heat from the heat exchanger, which represents heat loads from the full authority digital engine controller (FADEC), the vapor cycle system (VCS), engine, and fuel pump. Some of the heated fuel is sent to the engine, while the remaining is fed to the recirculation loop. Finally, fuel can be either sent to the Ram air cooler or bypass the cooler to return to Tank 1 or Tank 2. The derivation of the dynamic FTMS model in this section largely follows the development in [2] with the following differences: one additional actuator and two additional states are considered, corresponding to the return fuel mass flow rate to Tank 2, the fuel temperature in Tank 2, and the temperature of fuel exiting the heat exchanger.

The proposed model has five states corresponding to the mass of fuel in the recirculation tank, M_1, the mass of fuel in the reservoir tank, M_2, the fuel temperature in the recirculation tank, T_1, the fuel temperature in the reservoir tank, T_2, and the fuel temperature at the output of the heat exchanger, T_{h_o}. The pumped fuel mass flow rate $\dot{m}_f$ and the mass flow to the engine $\dot{m}_e$ are assumed constant, and the recirculation flow rate $\dot{m}_r$ is given by $\dot{m}_r = \dot{m}_f - \dot{m}_e$. While the assumption of constant flow rates may seem restrictive, it allows the numerical results in Section V to best highlight the effects of control decisions made by different controllers without the complications of time-varying engine mass flow rates.

The openings of the three three-way proportional valves create three control inputs: α that denotes the fraction of pumped fuel coming from Tank 1 such that $\dot{m}_1 = \alpha \dot{m}_f$; β that determines the fraction of fuel in the recirculation loop that goes through the Ram air cooler; and γ that is the fraction of the fuel in the recirculation loop that returns to Tank 1 such that the inlet mass flow rates to Tank 1 and Tank 2 are given by

$$\dot{m}_{1,i} = \gamma\,(\dot{m}_f - \dot{m}_e)$$
$$\dot{m}_{2,i} = (1-\gamma)(\dot{m}_f - \dot{m}_e).$$

From the conservation of mass, the ordinary differential equations (ODEs) governing the mass of fuel stored in each of the two tanks are

$$\dot{M}_1 = (\gamma - \alpha)\dot{m}_f - \gamma\,\dot{m}_e$$
$$\dot{M}_2 = (1-\gamma)(\dot{m}_f - \dot{m}_e) - (1-\alpha)\dot{m}_f.$$

From the conservation of energy and the modeling assumptions from [2], the ODEs governing the fuel temperatures in Tanks 1 and 2 are

$$\dot{T}_1 = \gamma\,\frac{(\dot{m}_f - \dot{m}_e)}{m_1}\cdot\left[T_{h_o} - T_1 - \beta(1 - e^{\mathrm{NTU}/\beta})(T_{h_o} - T_w)\right]$$
$$\dot{T}_2 = \frac{(1-\gamma)(\dot{m}_f - \dot{m}_e)}{m_2}\cdot\left[T_{h_o} - T_2 - \beta\left(1 - e^{\mathrm{NTU}/\beta}\right)(T_{h_o} - T_w)\right]$$

where NTU is the negative of the number of heat transfer units for the Ram air cooler and T_w is the cold side wall temperature. Both NTU and T_w are assumed constant.

The ODE for the temperature of the fuel at the output of the heat exchanger, T_{h_o}, is

$$\dot{T}_{h_o} = -\frac{\dot{m}_f}{M_{\mathrm{hx}}}\left[T_{h_o} + \alpha T_1 + (1-\alpha)T_2\right] + \frac{\dot{Q}_{\mathrm{in}}}{M_{\mathrm{hx}} c_v}$$

where M_{hx} is the mass of fuel present in the hot side of the heat exchanger, which is assumed constant.

As described in [2], $\dot{Q}_{\mathrm{in}}$ is the total heat load from several sources and is defined as

$$\dot{Q}_{\mathrm{in}} = \dot{Q}_F + \dot{Q}_{h_v} + \dot{Q}_{h_e} + \left(P_p + K_{Q_h}\dot{m}_f\right)$$

where $\dot{Q}_F$, $\dot{Q}_{h_v}$, and $\dot{Q}_{h_e}$ are the heat loads from the FADEC, the VCS, and the engine lubrication system, respectively. The term $(P_p + K_{Q_h}\dot{m}_f)$ refers to the heat from the fuel pump, assumed to be a linear function of $\dot{m}_f$.

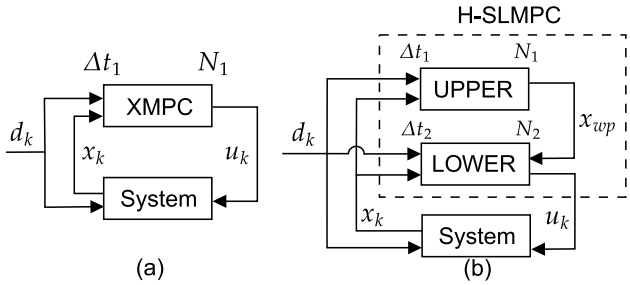

Fig. 3. (a) Centralized and (b) hierarchical closed-loop control schemes with corresponding sampling time Δt and prediction horizon N. XMPC refers to any of the centralized controllers: LMPC, NMPC, or SLMPC. In the H-SLMPC, UPPER refers to the upper level controller and LOWER to the lower level controller.

Combining these equations, the overall nonlinear dynamics of the system can be generically written as

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

where $x = [M_1\ M_2\ T_1\ T_2\ T_{h_o}]^\top$ and $u = [\alpha\ \beta\ \gamma]^\top$. The disturbance $d = \dot{Q}_{h_v}$ is defined to allow for a time-varying heat load from the VCS, which will be used in Section V.

III. Controller Formulations

This work proposes a new hierarchical NMPCformulation, which is based on SL (H-SLMPC), to solve the thermal endurance maximization problem. In Section V, the performance of the proposed H-SLMPC controller is compared to three baseline controllers: a centralized NMPC, a centralized LMPC, and a centralized SL MPC (SLMPC). Fig. 3 shows a schematic representation of each controller, while the remainder of this section details the overall structure of the controllers and the corresponding optimization problem formulations.

The MPC controllers presented in this section are formulated with a receding horizon for notational simplicity. However, the results shown in Section V are based on a slightly modified controller implementation that considers a shrinking horizon. Mission-driven control problems, such as the FTMS thermal endurance maximization, can benefit from a shrinking horizon since the controller is expected to operate for a limited amount of time, i.e., for the duration of the mission, and cannot operate indefinitely due to the monotonically decreasing mass of fuel. The technical details for formulating an MPC controller with a shrinking horizon are provided in [10].

A. Baseline Controllers

1) NMPC: This controller solves the following centralized NMPCoptimization problem at each time step.

Problem 1 [NMPC (Baseline Controller)]:

$$\min_{\mathbf{X},\mathbf{U},\mathbf{S}} \ell(\mathbf{U}, \mathbf{S}) \tag{2a}$$
$$\text{s.t. } \forall k \in [0, N_1 - 1]$$
$$x_{k+1} = x_k + \Delta t_1 f(x_k, u_k, d_k) \tag{2b}$$
$$\underline{u} \le u_k \le \bar{u} \tag{2c}$$
$$\underline{x} - s_k \le x_k \le \bar{x} + s_k \tag{2d}$$
$$0 \le s_k \le s_{k+1}. \tag{2e}$$

Here, the total cost is given by

$$\ell(\mathbf{U},\mathbf{S}) = \sum_{k=0}^{N_1-1} \ell_1(u_k, u_{k+1}) + \sum_{k=0}^{N_1} \ell_2(s_k)$$

where

$$\ell_1(u_k, u_{k+1}) = \begin{cases} ||u_0 - u_{-1}||_1, & k = 0 \\ ||u_{k+1} - u_k||_1, & k > 0 \end{cases} \tag{3a}$$

$$\ell_2(s_k) = w_s ||s_k||_1 \tag{3b}$$

with w_s as a weight applied to prioritize minimizing the slack variables relative to the rate-of-change of the inputs. The variable u_{-1} is the control action taken at the previous time step. Equation (2b) refers to the nonlinear dynamics model described in Section II discretized using a forward Euler approximation, (2c) constrains the inputs, and (2d) introduces slack variables on the state constraints that are penalized in the objective function. Inequality constraint (2e) ensures that the slack variables are monotonically increasing, which encourages the controller to delay the constraint violations as much as possible, thereby maximizing the thermal endurance. Due to (2b), the resulting problem is an NLP. It should be noted that the objective function $\ell(\mathbf{U},\mathbf{S})$ is tailored to the FTMS thermal endurance maximization problem, but the SL method used in this work is applicable to problems with general objective function definitions [35].

2) LMPC: The LMPC uses a linear time-invariant (LTI) version of the dynamics (2b) used by the NMPC. The LTI model used in (4b) is obtained by linearizing (1) around a specific point and discretizing using a forward Euler approximation, to be consistent with the discretization approach used in (2b). The resulting centralized linear optimization problem is shown as follows.

Problem 2 [LMPC (Baseline Controller)]:

$$\min_{\mathbf{X},\mathbf{U},\mathbf{S}} \ell(\mathbf{U},\mathbf{S}) \tag{4a}$$

s.t. $\forall k \in [0, N_1 - 1]$

$$x_{k+1} = x_k + \Delta t_1[Ax_k + Bu_k + Wd_k + c] \tag{4b}$$

$$\underline{u} \le u_k \le \bar{u} \tag{4c}$$

$$\underline{x} - s_k \le x_k \le \bar{x} + s_k \tag{4d}$$

$$0 \le s_k \le s_{k+1}. \tag{4e}$$

Here

$$A = \left.\frac{\partial f(x,u,d)}{\partial x}\right|_{(x_0,u_{-1},d_0)} \tag{5a}$$

$$B = \left.\frac{\partial f(x,u,d)}{\partial u}\right|_{(x_0,u_{-1},d_0)} \tag{5b}$$

$$W = \left.\frac{\partial f(x,u,d)}{\partial d}\right|_{(x_0,u_{-1},d_0)} \tag{5c}$$

$$c = f(x_0,u_{-1},d_0) - Ax_0 - Bu_{-1} - Wd_0. \tag{5d}$$

Note that, to improve the LMPC's prediction capabilities, the matrices in (5) are recomputed at every iteration based on the currently available state, input, and disturbance vectors.

The linearized dynamics (4b) make **Problem 2** an LP, which can be solved much more efficiently than the NLP from **Problem 1**. However, as shown in [1], an LTI model of the FTMS may generate very large prediction errors in temperature. For the three-state model considered in [1], temperature prediction errors on the order of 70 K were reported over a long prediction horizon. Even if, in some cases, the prediction errors may cause the LMPC to act conservatively such that the resulting thermal endurance is satisfactory, in other cases, the errors may lead the controller to achieve very poor and unacceptable performance. In [1], the thermal endurance achieved by a similar LMPC was approximately 28% less than that achieved by a similar NMPC in a scenario with a known time-varying disturbance.

3) SLMPC: Following the approach from [36], as described in more detail in Section IV, the SLMPC controller *iteratively* solves the following centralized linear optimization problem at each time step.

Problem 3 [SLMPC (Baseline Controller)]:

$$\min_{\mathbf{X},\mathbf{U},\mathbf{S}} \ell(\mathbf{U},\mathbf{S}) \tag{6a}$$

s.t. $\forall k \in [0, N_1 - 1]$

$$x_{k+1} = x_k + \Delta t_1[A_k x_k + B_k u_k + W_k d_k + c_k] \tag{6b}$$

$$\underline{u} \le u_k \le \bar{u} \tag{6c}$$

$$\underline{x} - s_k \le x_k \le \bar{x} + s_k \tag{6d}$$

$$0 \le s_k \le s_{k+1} \tag{6e}$$

$$\left\| x_{k,s} - x^*_{k,s} \right\|_\infty \le \delta^{(j)}. \tag{6f}$$

Here

$$A_k = \left.\frac{\partial f(x,u,d)}{\partial x}\right|_{(x_k^*,u_k^*,d_k)} \tag{7a}$$

$$B_k = \left.\frac{\partial f(x,u,d)}{\partial u}\right|_{(x_k^*,u_k^*,d_k)} \tag{7b}$$

$$W_k = \left.\frac{\partial f(x,u,d)}{\partial d}\right|_{(x_k^*,u_k^*,d_k)} \tag{7c}$$

$$c_k = f(x_k^*,u_k^*,d_k) - Ax_k^* - Bu_k^* - Wd_k \tag{7d}$$

where x_k^* and u_k^* are vectors of the reference state ($\mathbf{X}^*$) and input ($\mathbf{U}^*$) trajectories, while d_k are vectors of the disturbance trajectory $\mathbf{D}$ provided to the controller. The variable $x_{k,s}$ refers to the scaled version of the state trajectory at time step k, and $x^*_{k,s}$ is the corresponding variable from the reference state trajectory. The variable $\delta^{(j)}$ is the trust radius at iteration j of the SL algorithm. The use of scaled variables in (6f) ensures that each dimension of the state vector has approximately the same weight in the computation of the infinity norm. This, in turn, ensures that deviations from the reference state trajectory are constrained uniformly. The variables $\mathbf{X}^*$, $\mathbf{U}^*$, $\mathbf{D}$, and $\delta^{(j)}$ are given as input parameters to **Problem 3**, and the details of how they are computed are given in Section IV.

Note the two differences between **Problem 2** and **Problem 3**: 1) the linearized dynamics from (6b) and (7) are now time-varying over the prediction horizon and 2) **Problem 3** contains a trust-region constraint (6f). The resulting optimization problem is an LP, solved once at each iteration, and, therefore, benefits from the availability of efficient solvers. Even so, it is important to note that the iterative procedure adopted by this controller aims at finding a solution

that is also a local minimum of **Problem 1**. That is, the iterative procedure seeks a solution to an NLP by solving a sequence of much simpler to solve LPs.

B. Proposed Controller—H-SLMPC

The H-SLMPC differs from other controllers by applying a hierarchy of two controllers, as shown in Fig. 3. Each controller in the hierarchy uses the same iterative procedure as the SLMPC to solve a nonlinear optimization problem based on **Problem 1**.

The upper level controller updates at a relatively slow rate compared to the lower level, which allows it to use a long prediction horizon. While solving **Problem 3** exactly as the SLMPC, the upper level controller does not apply the resulting control action directly to the system. The role of the upper level controller is to provide the lower level with guidance on regions of the state space that should be achieved in the short-term such that long-term optimality and constraint satisfaction is preserved. This guidance is provided through a *waypoint*, which is typically the first predicted point in the state trajectory of the optimal solution computed by the upper level controller [10]. If $\mathbf{X}^*$ is the state trajectory from the solution obtained by the upper level controller, then the waypoint is given by

$$x_{\text{wp}} = x_1^*. \tag{8}$$

Since the upper level computes a new solution at every Δt_1, the currently active waypoint is updated accordingly. The actual implementation in the following numerical simulations uses a waypoint defined as $x_{\text{wp}} = x_2^*$, which is slightly different from (8) for reasons discussed in detail in Section V-C2.

The lower level controller updates at a faster rate, which provides better high-frequency disturbance rejection compared to the upper level controller. The fast update rate usually requires the lower level controller to use a short prediction horizon. Since the actual control action applied to the plant comes from the lower level, there is a clear need for guidance from the upper level through the waypoint mechanism. This allows the H-SLMPC as a whole to be able to reject high-frequency disturbances while seeking optimal operation in the long term. To compute the control action at each time step, the lower level controller applies the SL algorithm to solve a slightly modified version of **Problem 3**, with an additional waypoint-tracking terminal constraint, as detailed in the following.

Problem 4 [H-SLMPC (Proposed Controller)]:

$$\min_{\mathbf{X},\mathbf{U},\mathbf{S}} \ell(\mathbf{U},\mathbf{S}) + w_{\text{wp}}||s_{\text{wp}}||_1 \tag{9a}$$

$$\text{s.t. } \forall k \in [0, N_2 - 1]$$

$$x_{k+1} = x_k + \Delta t_2[A_k x_k + B_k u_k + W_k d_k + c_k] \tag{9b}$$

$$\underline{u} \le u_k \le \bar{u} \tag{9c}$$

$$\underline{x} - s_k \le x_k \le \bar{x} + s_k \tag{9d}$$

$$0 \le s_k \le s_{k+1}, \tag{9e}$$

$$||x_{k,s} - x_{k,s}^*||_\infty \le \delta^{(j)} \tag{9f}$$

$$-s_{\text{wp}} \le \left(x_{N_2} - x_{\text{wp}}\right) \le s_{\text{wp}}. \tag{9g}$$

Here, s_{wp} is the slack from the terminal state to the current waypoint x_{wp}, and w_{wp} is a weighting term that penalizes these deviations. Note the use of a different prediction horizon N_2 and sampling time Δt_2 compared to the upper level controller and centralized controllers. More details on the waypoint coordination mechanism and its implementation with a shrinking horizon can be found in [10].

IV. Successive Linearization

SL, which can be considered part of a broader class of NLP algorithms known as sequential convex programming (SCP) [34], is a technique for solving the NLP resulting from an NMPCformulation. This is done by iteratively linearizing the nonlinear constraints and objective function around a given candidate system trajectory and solving the resulting convex program (**Problem 3** or **Problem 4**) until convergence is achieved. While there are different variants of SL-based algorithms, the approach used in this work is most closely related to the `SCvx` algorithm in [36]. **Algorithm 1** formalizes the specific implementation used in this work at a high level, while the details are discussed throughout this section.

In the remainder of this section, the superscript $*$ refers to the variable value from the last *valid iteration*, which is an iteration for which the solution to **Problem 3** or **Problem 4** was not rejected. Reference input and state trajectories are the results of *valid iterations* and, therefore, are designated by $\mathbf{U}^*$ and $\mathbf{X}^*$, respectively. The objective function value of the solution to the linearized problem (**Problem 3** or **Problem 4**) at the current iteration is given by L. The variables $\mathbf{X}_l$ and $\mathbf{S}_l$ refer to the state and slack trajectories obtained from the solution to the *linearized* problem, while $\mathbf{X}_{\text{nl}}$ and $\mathbf{S}_{\text{nl}}$ refer to the corresponding trajectories obtained from the original *nonlinear* dynamics. The objective function value computed using the actual input and state trajectories of the current iteration, $(\mathbf{U}_{\text{new}}, \mathbf{X}_{\text{nl}})$, is given by J. The initial guess for the reference input trajectory is $\mathbf{U}_0$.

A. Computation of a Reference System Trajectory

Algorithm 1 (Lines 2 and 9): In this step, the predicted system trajectory is computed based on the nonlinear discrete-time dynamics (2b) given an initial state x_0, a sequence of inputs $\mathbf{U}$, and a sampling time Δt_{ref}. This results in a corresponding state trajectory $\mathbf{X}$, a slack variables trajectory $\mathbf{S}$, which captures the state constraint violations at each time step, and the associated true objective function cost J^* (Line 2) or J (Line 9). Note that Δt_{ref} used to compute the reference trajectory must be small enough, such that the forward Euler discretization is still stable, and does not necessarily match the controller sampling time. See the Appendix for more details. In this work, a value of $\Delta t_{\text{ref}} = 2$ s was used for SL-based controllers.

Much like any other algorithm for solving NLPs, the SL method might converge to a local minimum, which depends on the initial guess provided. In this case, the initial guess is given by the initial reference trajectory $\mathbf{U}_0$, which, together with x_0, allows the computation of the initial reference state trajectory and slack variables.

Algorithm 1 SL

Require: $\mathbf{U}_0, x_0, \delta_0, \Delta t_{ref}$
1: **Initialize:**
$\mathbf{U}^* \leftarrow \mathbf{U}_0$, $\delta^{(0)} \leftarrow \delta_0$, relinearize $\leftarrow$ **true**
2: $\mathbf{X}^*, \mathbf{S}^*, J^* \leftarrow$ COMPUTE REFERECE TRAJECTORY$(x_0, \mathbf{U}^*, \Delta t_{ref})$ ▷ see Section IV-A
3: $j = 0$
4: **while true do**
5: **if** relinearize **then**
6: $\mathbb{A}, \mathbb{B}, \mathbb{W}, \mathbb{C} \leftarrow$ COMPUTE LTV MATRICES$(\mathbf{X}^*, \mathbf{U}^*)$ ▷ see Section IV-B
7: **end if**
8: $\mathbf{X}_l, \mathbf{U}_{new}, \mathbf{S}_l, L \leftarrow$ SOLVE LP$(\mathbf{X}^*, \mathbb{A}, \mathbb{B}, \mathbb{W}, \mathbb{C}, \delta^{(j)})$ ▷ see Section IV-C
9: $\mathbf{X}_{nl}, \mathbf{S}_{nl}, J \leftarrow$ COMPUTE REFERENCE TRAJECTORY$(x_0, \mathbf{U}_{new}, \Delta t_{ref})$ ▷ see Section IV-A
10: **if** STOP CONDITION$(j, J^*, L, x^*_{k,s}, x_{k,s})$**==true then** ▷ see Eq. (11a)
11: **break**
12: **else**
13: $\delta^{(j+1)}, J^*, \mathbf{X}^*, \mathbf{U}^*,$ relinearize $\leftarrow$ TRUST-REGION UPDATE$(\delta^{(j)}, J^*, \mathbf{X}^*, \mathbf{U}^*, J, L, \mathbf{X}_{nl}, \mathbf{U}_{new})$ ▷ see Section IV-E
14: **end if**
15: $j \leftarrow j + 1$
16: **end while**
17: **return** $\mathbf{U}^*$

B. Computation of Linear Time-Varying (LTV) System Matrices From a Reference Trajectory

Algorithm 1 (Line 6): In this step, the current reference state and input trajectories $(\mathbf{X}^*, \mathbf{U}^*)$ are used to obtain the discrete-time LTV system matrices based on (7). The discrete-time LTV model matrices are grouped as follows:

$$\mathbb{A} = \{A_k, k = 0, \ldots, N\} \tag{10a}$$
$$\mathbb{B} = \{B_k, k = 0, \ldots, N\} \tag{10b}$$
$$\mathbb{W} = \{W_k, k = 0, \ldots, N\} \tag{10c}$$
$$\mathbb{C} = \{c_k, k = 0, \ldots, N\}. \tag{10d}$$

If the sampling time Δt_{ref} used to obtain the reference state and input trajectories is smaller than the controller's sampling time, the procedure outlined in the Appendix is used to obtain the LTV model matrices for the controller.

C. Solving the LP

Algorithm 1 (Line 8): In this step, the algorithm calls an external LP solver using the current reference state trajectory $\mathbf{X}^*$, the matrices that represent the LTV model ($\mathbb{A}$, $\mathbb{B}$, $\mathbb{W}$, and $\mathbb{C}$), and the current trust radius $\delta^{(j)}$. The solver will return the new input $\mathbf{U}_{\text{new}}$, the computed state trajectory for the linearized system $\mathbf{X}_l$, and slack variables $\mathbf{S}_l$ for this trajectory. These result in the new objective function value L for the linearized problem. Though these values are exact for the linearized system dynamics, they are approximations of the original nonlinear dynamics. The actual values, considering the nonlinear dynamics, will be obtained in Line 9 using $\mathbf{U}_{\text{new}}$ and (2b) to compute $\mathbf{X}_{\text{nl}}$, $\mathbf{S}_{\text{nl}}$, and J.

D. Stopping Condition

Algorithm 1 (Line 10): One of the factors that distinguish different SCP algorithms is the condition used to decide when iterations have converged to an acceptable solution and, therefore, stop the iterative process [35]. In this work, the following stopping condition is used:

$$C_1 \text{ AND } (C_2 \text{ OR } C_3 \text{ OR } C_4) \tag{11a}$$

where

$$C_1 = j \geq j_{\min} \tag{11b}$$
$$C_2 = \max_k ||x^*_{k,s} - x_{k,s}||_\infty \leq \epsilon_3, \quad k = 1, \ldots, N \tag{11c}$$
$$C_3 = J^* \leq \epsilon_1 \tag{11d}$$
$$C_4 = |\Delta L| \leq \epsilon_2 \tag{11e}$$

where $\Delta L = J^* - L$ measures the improvement predicted by the linear optimizer. Here, (11b) ensures that the algorithm will run a minimum number of iterations. Condition (11c) detects whether changes in the state trajectory at the current iteration are sufficiently small to stop and is suggested in [35] when the cost function depends on the inputs. Conditions (11d) and (11e) were included to handle cases where numerical precision issues could lead to inconsistency: if the algorithm were allowed to proceed to the next iteration with very low values of J^* or $|\Delta L|$ at the current iteration, numerical errors in the model or in the solver solution at the next iteration could have created situations where $\Delta L < 0$, causing the algorithm to fail. The values used for the parameters $j_{\min}$, ϵ_1, ϵ_2, and ϵ_3 are provided in Section V-D.

E. Trust-Region Update

Algorithm 1 (Line 13): For SL methods, iteratively solving the optimization problem using LTV models can effectively identify local optima of the original nonlinear optimization problem as long as solutions to the linearized problem do not deviate significantly from the reference trajectory used for the linearization. Therefore, deviations are bounded by the parameter δ in (6f) and (9f). The trust-region created by δ should be small, to minimize linearization error, but large enough to achieve convergence in a few iterations. Since this tradeoff depends on the nonlinearity in the system model and the effect of this nonlinearity on the cost function, Line 13 implements Algorithm 2, using the formulation defined for the `SCvx` algorithm in [36], to adapt the trust-region size $\delta^{(j)}$ used in (6f) and (9f) at each iteration of the SL algorithm.

Algorithm 2 Trust-Region Update

Require: $\delta, J^*, \mathbf{X}^*, \mathbf{U}^*, J, L, \mathbf{X}_{nl}, \mathbf{U}_{new}$
$\Delta J \leftarrow J^* - J$
$\Delta L \leftarrow J^* - L$
$\rho \leftarrow \frac{\Delta J}{\Delta L}$
if $\rho < \rho_0$ **then** ▷ reject and decrease trust radius
  $\delta \leftarrow \frac{\delta}{\alpha}$
  relinearize $\leftarrow$ **false**
else ▷ accept, this is a valid iteration
  if $\rho < \rho_1$ **then**
    $\delta \leftarrow \frac{\delta}{\alpha}$ ▷ decrease trust radius
  else if $\rho < \rho_2$ **then**
    ▷ do nothing, keep same trust radius
  else
    $\delta \leftarrow \alpha\delta$ ▷ $\rho \geq \rho_2$, increase trust radius
  end if
  $\delta \leftarrow min\left(max\left(\delta, \delta_{min}\right), \delta_{max}\right)$
  $J^* \leftarrow J$
  $\mathbf{X}^* \leftarrow \mathbf{X}_{nl}$
  $\mathbf{U}^* \leftarrow \mathbf{U}_{new}$
  relinearize $\leftarrow$ **true**
end if
return $\delta, J^*, \mathbf{X}^*, \mathbf{U}^*$, relinearize

TABLE I
SYSTEM PARAMETERS

Variable	Description	Units	Nominal Value*	Lower Bound	Upper Bound
M_1	Recirculation tank mass	kg	200	50	2850
M_2	Reservoir tank mass	kg	2850	50	2850
T_1	Recirculation fuel temperature	K	288	280	333
T_2	Reservoir fuel temperature	K	288	280	333
T_{h_o}	Heat Exchanger output fuel temperature	K	288	280	421
$\dot{m}_f$	Pumped fuel flow rate	kg/s	1.0	-	-
$\dot{m}_e$	Engine fuel flow rate	kg/s	0.26	-	-
α	Recirculation fuel fraction	-	-	0	1.0
β	Cooler fuel fraction	-	-	0	1.0
γ	Tank 1 inlet fuel fraction	-	-	0	1.0
c_v	Fuel specific heat	J/(kg·K)	2,010	-	-
$\dot{Q}_F$	FADEC heat input	W	1,000	-	-
$\dot{Q}_{h_v}$	VCS heat input	W	65,000	-	-
$\dot{Q}_{h_e}$	Engine heat input	W	10,000	-	-
P_p	Fuel pump power	W	50,000	-	-
K_{Q_h}	Fuel pump heat input coeff.	W/kg	-6,618	-	-
NTU	Negative of the number of Heat Transfer Units	-	-0.37	-	-
T_w	Heat exchanger cold wall temperature	K	238	-	-

*Value also serves as the initial state for M_1, M_2, T_1, T_2, and T_{h_o}.

Based on the latest solution ($\mathbf{X}_l, \mathbf{U}_{\text{new}}$) to the linearized optimization problem (**Problem 3** or **Problem 4**), the improvement over the last valid solution is computed. This is done for the improvement planned by the solver, ΔL, and for the actual improvement, $\Delta J = J^* - J$. Except when numerical errors are significant, which are handled according to the procedure described in Section IV-D, the definition of the linearized optimization problems ensures that $\Delta L \geq 0$ at every iteration. This fact results from an important characteristic of the LTV approximation of the nonlinear dynamics: if $u_k = u_k^*$, then the approximation is exact, and $x_k = x_k^*$ for all $k = 0, \ldots, N$. Therefore, in the worst case, the solver can find a solution where this condition holds and $L = J^*$. Otherwise, a better solution is found where $\Delta L > 0$. This fact, though, does not guarantee that the actual improvement ΔJ is nonnegative. In Algorithm 2, with $\rho_0 = 0$, when $\Delta J < 0$, the current iteration is rejected since the linearized problem does not seem to approximate the original problem well enough. If, otherwise, $\Delta J > 0$, this is considered a *valid iteration*, where a better solution, considering the nonlinear dynamics, was found, and the current valid objective function value J^* is updated, as well as the current reference trajectories ($\mathbf{X}^*, \mathbf{U}^*$). Meanwhile, the trust radius δ is updated according to ρ, the relative magnitude between ΔJ and ΔL. If $\rho < \rho_1$, it is considered that the actual improvement is low enough compared to ΔL such that a reduction in the trust radius is necessary to avoid "overstepping" in the next iteration [35]. If $\rho_1 < \rho < \rho_2$, ΔJ and ΔL are close enough to justify maintaining the current trust radius. Finally, if $\rho \geq \rho_2$, the actual improvement was considerably greater than expected by the optimizer, and the trust radius can be increased to allow for larger steps and faster convergence.

It should be noted that the trust-region has an important role in addressing the issues of "artificial unboundedness" and "artificial infeasibility" introduced by the linear approximation of the dynamics in **Problem 3** and **Problem 4**. The reader is referred to the excellent tutorial on SCP methods in [35] for a discussion on these issues and how they motivate the use of the trust region. The values used for the parameters ρ_0, ρ_1, ρ_2, α, and δ_0 are provided in Section V-D.

V. NUMERICAL EXAMPLES

All the numerical results in this article were generated using MATLAB and YALMIP [37] on a desktop computer with a 3.2-GHz i7-8700 processor and 16 GB of RAM. NLP problems for the NMPC were solved with IPOPT [29], while all LP optimization problems were solved with Gurobi [38]. Based largely on the model used in [2], Table I shows the values of the FTMS model parameters in the equations leading to (1).

A. Test Cases

Three different test cases are proposed to compare the controllers' performance. In Case 1, the controllers are simulated with the nominal conditions from Table I. In Case 2, the large low-frequency disturbance shown in Fig. 4(a) is applied. Here, a perfect preview of the entire disturbance profile is provided to the controllers. This is intended to test long-term planning abilities under changing conditions. Case 3 is used to test the controllers' ability to reject high-frequency *measured* disturbances, i.e., the controllers are only aware of the current disturbance value, considering the disturbance constant and persistent for the entire prediction horizon. In the high-frequency disturbance case, the Q_{h_v} heat load assumes random values ranging from 20% to 170% of the nominal value and changes every 10 s. A sample from this disturbance profile is shown in Fig. 4(b).

It should be noted that, although the disturbance profiles in Case 2 and Case 3 were artificially generated and real disturbance profiles may look very different, the simulated

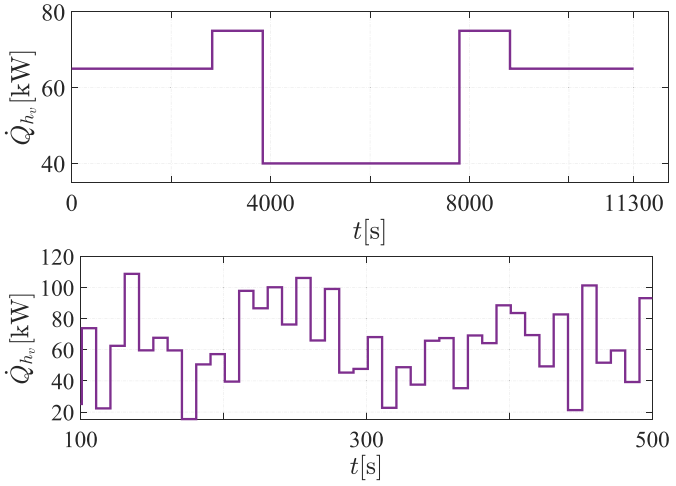

Fig. 4. Profile of the low-frequency disturbance in Case 2 (upper) and sample of the high-frequency disturbance in Case 3 (lower).

profiles here capture some of the core characteristics of the disturbances expected in real missions. Average heat loads in different mission phases can be drastically different, as simulated in Case 2, while rapid changes in heat loads can also occur within a mission phase, as simulated in Case 3.

B. Performance Measure

While the ultimate goal of studying the proposed controllers is the maximization of thermal endurance, this measure alone may not fully capture the capabilities of each controller for that purpose. A complementary metric is proposed here.

Average Temperature Violation (ATV):

$$\text{ATV} = \frac{\sum_{k=0}^{N_{\text{sim}}} \Delta t_{\text{sim}} v_k}{T_{\text{sim}}} \tag{12a}$$

$$v_k = |v_{T_1,k}| + |v_{T_2,k}| + |v_{T_{h_o},k}| \tag{12b}$$

where N_{sim} is the total number of steps in the simulation, Δt_{sim} is the simulation sampling time, and T_{sim} is the total simulation time. The actual constraint violations for the temperatures at time step k are given by v_k. Each component is computed as

$$v_{T_i,k} = \max\big((T_i - \bar{T}_i), 0\big) + \max\big((\underline{T}_i - T_i), 0\big)\big) \tag{13}$$

with T_i as a placeholder for T_1, T_2, or T_{h_o} and $\bar{T}_i$, $\underline{T}_i$ for the corresponding upper and lower bounds, respectively.

Fig. 5 shows an example that illustrates the usefulness of the ATV. Here, Controller A has a smaller ATV compared to Controller B, which is better, but also a misleadingly shorter thermal endurance t_A due to small constraint violations early in the mission. Therefore, both thermal endurance and ATV will be considered when comparing controller performance.

C. Control Horizon and Sampling Time Considerations

1) Centralized Controllers: Two factors determine the sampling time to be used by the controllers, namely, the ability to handle the high-frequency disturbance from Case 3 and the stability and accuracy of the discretization scheme used in the optimization problem formulation. In this application, the high-frequency disturbance calls for a sampling time

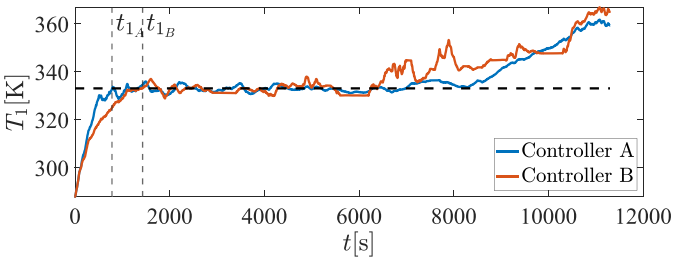

Fig. 5. Illustrative example of how the thermal endurance alone may not fully capture a controller's capability in minimizing constraint violations.

TABLE II
CONTROLLER PARAMETERS

Controller	N_1/N_2	$\Delta t_1/\Delta t_2$ [s]	w_s	w_{wp}	Δt_{ref} [s]
LMPC	1130	10	1000	-	2
NMPC	10	2	1000	-	-
SLMPC	100	10	1000	-	2
H-SLMPC	113/20	100/10	1000	1000	2

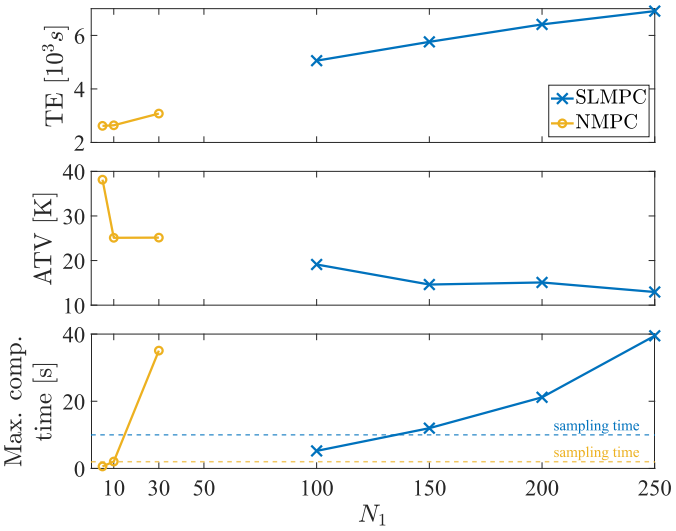

Fig. 6. Effect of increasing the prediction horizon N_1 on the performance of the SLMPC and NMPC when the high-frequency disturbances are applied to the system (Case 2).

$\Delta t_1 \leq 10$ s. Simulations performed with the complete FTMS model in (1) show that significant discretization errors are introduced when using $\Delta t_1 > 2$ s based on the forward Euler discretization (2b) used in this work. This limit is imposed by the heat exchanger output temperature state, T_{h_o}, which has dynamics much faster than the other system states. This results in the need for control of dynamics at different timescales, which requires both a fast sampling time and a long prediction horizon for good performance. However, for the LMPC and SLMPC, the matrices of a model with a small sampling time, Δt_{ref}, can be combined to obtain a model with a sampling time, Δt_1, larger than the limit imposed by the discretization, as shown in the Appendix. Table II summarizes the values used in the simulations.

For the prediction horizon, Fig. 6 shows the processing times and performance of the SLMPC and NMPC considering the sampling time defined for each for the most computationally demanding scenario, Case 3. The controllers' computation

time exceeds their sampling times for $N_1 > 100$ and $N_1 > 10$ for the SLMPC and NMPC, respectively. Moreover, for both controllers, there is a reduction in constraint violations as N_1 increases. Therefore, $N_1 = 100$ will be used for the SLMPC and $N_1 = 10$ for the NMPC. Fig. 6 also suggests that SL is indeed a better option for solving the NLPs arising in NMPCcompared to the use of generic NLP solvers, such as IPOPT, since the SLMPC seems to scale better with the prediction horizon length. For the LMPC, even using a horizon covering the whole mission profile, $N_1 = 1130$, the maximum computation time never exceeded the sampling time, and therefore, this value was used in the simulations.

2) Hierarchical Controller—H-SLMPC: From Fig. 6, it can be seen that solving **Problem 3** with $N_1 = 100$ takes less than 10 s. Since this problem is also solved by the upper level controller in the H-SLMPC, a sampling time $\Delta t_1 = 100$ s will be used for this controller with a horizon $N_1 = 113$. This allows the upper level controller in the H-SLMPC to predict through the whole duration of the mission while keeping the computation times low enough for computation in real time.

Conversely, the lower level requires a small sampling time and a short prediction horizon to cope with fast dynamics and high-frequency disturbances. In [10], the prediction horizon of the lower level (N_2) is set such that the lower level controller predicts up to the next update of the upper level controller, i.e., $N_2 = (\Delta t_1/\Delta t_2)$, and the lower level controller is formulated such that it follows a waypoint given by the upper level by means of a terminal constraint. However, Raghuraman et al. [16] show how a waypoint coordination mechanism between the upper and lower level controllers can induce greedy, short-sighted, behavior in the lower level controller. An example is shown in Fig. 7(a), where the highly oscillating behavior of the control inputs in some periods of the simulation can be explained by the greedy behavior of the lower level.

In [16], waysets with coordinating terminal costs computed online are used to overcome such greedy behavior issues and shown to work for linear systems, but the efficient online computation of waysets is not yet possible for nonlinear systems in general. In this work, this issue is handled with the use of an extended horizon. Here, a shrinking horizon is also used but with $N_2 = 2(\Delta t_1/\Delta t_2)$, and the waypoint corresponds to the state predicted by the upper level *two steps ahead* with $x_{\text{wp}} = x_2^*$ [different from (8)]. This way, when the lower level reaches the point where the upper level sends a new waypoint, the lower level still has half of its original shrinking horizon remaining, i.e., $(N_2/2)$ steps. At this point, the lower level horizon is reset to the original length N_2 and starts to steer the system toward the new waypoint. This way, any significant changes between subsequent waypoints can be handled more smoothly since the lower level still has $(N_2/2)$ steps to make the necessary adjustments to the control action. Fig. 7(b) shows the response when using the extended horizon for the same scenario simulated in Fig. 7(a). It can be seen that the control response now is smooth, and effective coordination between controllers is still achieved. A natural side effect of using the extended control horizon in the lower level is the increased computational effort associated with approximately twice as many decision variables in the optimization problem. All H-SLMPC results in Section V-D make use of the extended horizon for the lower level controller.

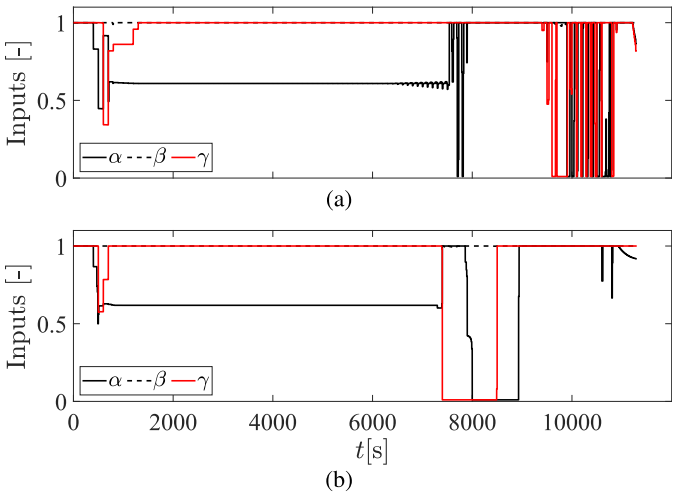

Fig. 7. Profile of H-SLMPC control inputs with (a) nonextended horizon and (b) extended horizon when no disturbances are applied to the system.

D. Controllers' Performance Evaluation

This section compares the performance of all four controllers in the test cases described in Section V-A. The MPC-related controller parameters used are summarized in Table II. It should be noted that, for the NMPC and SL-based controllers, the initial guess provided by IPOPT and the SL algorithm can have a considerable effect on the computation time. To minimize the effect of the initial guess on the results, for $k > 0$, the result from the previous time step is used as the initial guess for the current step for both algorithms. For the NMPC, IPOPT was called once before simulating the first time step to obtain a reasonable initial guess. For the SL-based controllers, a rough initial guess with a constant input trajectory with $u_k = [0.7\ 0.7\ 0.7]^T$ was used. Also, the following SL-related parameters for the SL-based controllers were used: $j_{\min} = 2$, $\epsilon_1 = 10^{-3}$, $\epsilon_2 = 10^{-3}$, $\epsilon_3 = 5 \cdot 10^{-2}$, $\rho_0 = 0$, $\rho_1 = 0.25$, $\rho_2 = 0.9$, $\alpha = 2$, and $\delta_0 = 0.4$.

Fig. 8 shows the trajectories obtained for the controllers in each scenario. The columns of this figure correspond to the three different test cases, while the rows correspond to the four different control approaches. Each set of results has three subplots, where the control decisions of the three three-way valves are present on top, the fuel mass in the two tanks in the middle, and the fuel temperatures in the tanks and outlet of the heat exchanger on the bottom. Consistent with the results in [2], in general, the H-SLMPC tries to increase the temperature T_1 up to the upper limit early in the mission. This allows the system to send a larger fraction of the hotter fuel from Tank 1 to the engine most of the time, maximizing heat rejection. This behavior is reproduced by the LMPC, which also has a prediction horizon that spans the whole mission. Conversely, the NMPC and SLMPC, with their short horizons, cannot predict far enough to make good decisions and, therefore, have temperature violations much

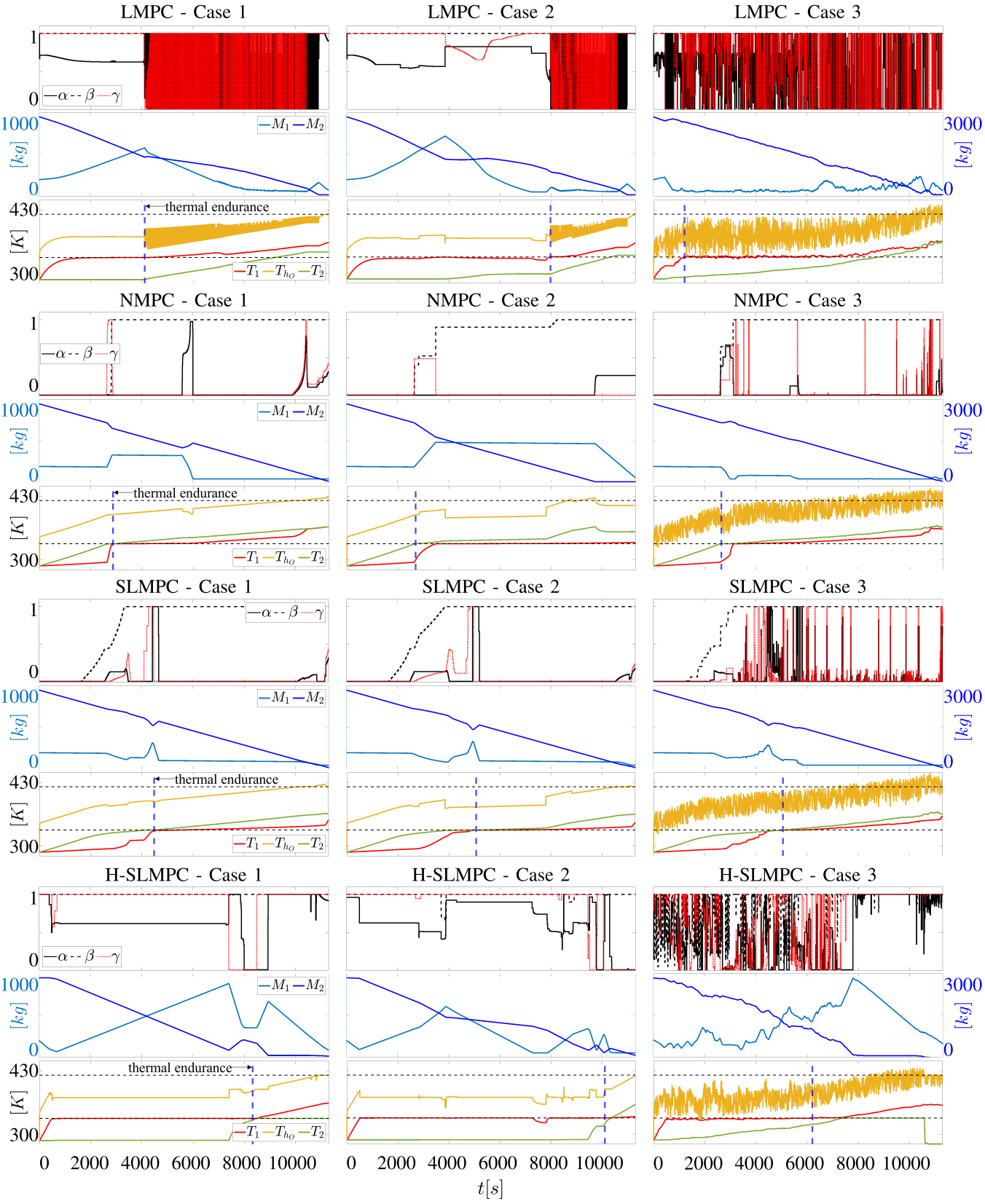

Fig. 8. State and input trajectories of all four controllers for the three test cases. The horizontal dashed lines in the temperature plots show the upper limits of 421 K for T_{h_o} and 333 K for T_1 and T_2, while the vertical dashed lines show the thermal endurance.

earlier. This difference between the controllers with short and long horizons can also be seen with respect to how they use the Ram air cooler: the H-SLMPC and LMPC tend to keep $\beta = 1.0$ throughout the mission, while the NMPC and SLMPC controllers waste the opportunity to remove heat from the system at the beginning of the mission by leaving β with a

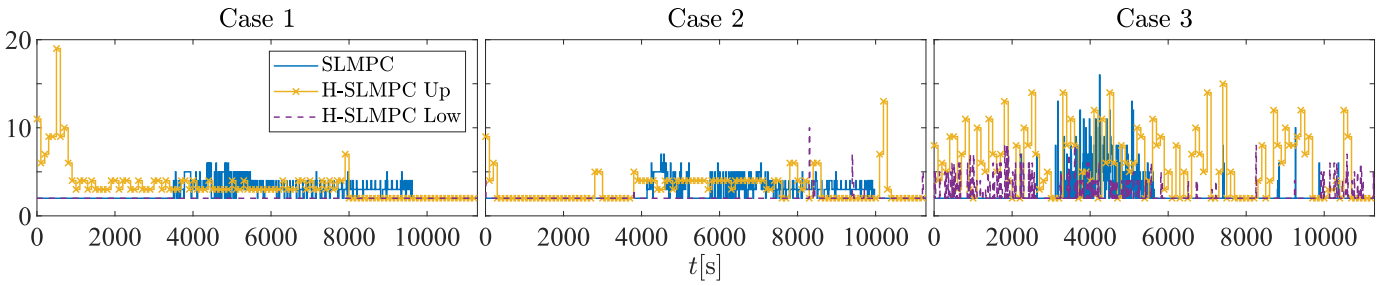

Fig. 9. Number of SL iterations until convergence for Cases 1–3.

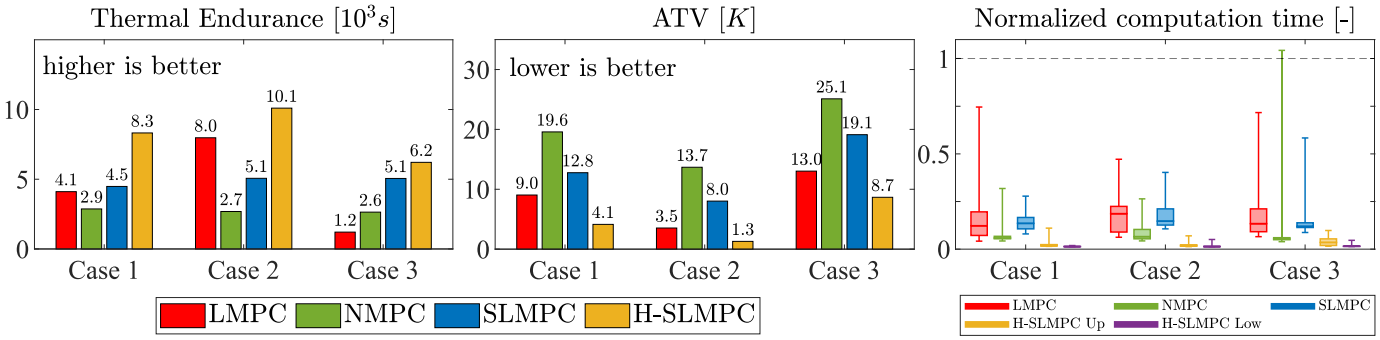

Fig. 10. Performance comparison of the controllers for Cases 1–3. The computation times are normalized with respect to each controller's sampling time.

lower value. Also, in Cases 1 and 2, toward the end of the mission, the H-SLMPC controller decreases γ to direct more of the hot fuel in the recirculation loop to Tank 2. This prevents T_1 from violating the constraint early. A very undesirable feature of the LMPC is the highly oscillatory behavior that occurs once constraints are violated although it uses the same control input smoothing term in the objective function as used by the other controllers.

In Case 2, the H-SLMPC plans and reacts to the foreseen large changes in the heat load. Around 3000 s, when the first large increase in the load hits the system, the controller decreases α, thereby drawing more fuel from Tank 2. This causes the mix of fuel being sent to the heat exchanger to be cooler than before since, at this moment, $T_2 < T_1$, which helps keep the temperatures below the predefined limit. The SLMPC, in contrast, already has $T_2 > T_1$ when this first increase in heat load is approaching. The controller does slightly increase α and γ, drawing more fuel from Tank 1 and increasing the amount of hot fuel sent to Tank 1. The small prediction horizon though prevents the SLMPC from making these changes more aggressively, and soon, around 5000 s, the temperature constraints are violated. When the disturbance switches to a low level around 4000 s, the H-SLMPC increases α again to increase the fraction of hotter fuel from Tank 1 to be sent to the engine. This compensates for the lower thermal load and keeps T_1 close to the limit, maximizing the heat rejection of the system. Finally, *before* the heat load is increased again at 8000 s, the controller reduces α to increase the fraction of the cooler fuel from Tank 2 in the recirculation loop and so precool the fuel in Tank 1, anticipating and avoiding a temperature violation due to the sudden increase in the heat load. The failure of the SLMPC in maximizing the thermal endurance in Case 1 and Case 2 is highlighted by the almost identical controller responses even though the heat load profile from $\dot{Q}_{h_v}$ changes drastically from one case to the other. For Case 2, the LMPC presents a similar behavior to the H-SLMPC, though with a less pronounced precooling around 8000 s, which seems to cause the early constraint violation when the heat load is increased again. Overall, the same trends observed for each controller in Case 1 and Case 2 are also present in Case 3. With the high-frequency disturbance present now, the control inputs from the controllers change more frequently. The inputs from the SLMPC are less aggressive compared to the H-SLMPC likely due to the fact that the H-SLMPC tends to keep T_1 closer to the upper limit, which requires stronger actions to prevent a temperature violation. Here, the LMPC, though with a long prediction horizon, fails to prevent a constraint violation early in the simulation when subject to the high-frequency disturbance. Fig. 8 shows how keeping T_{h_o} below its upper bound does not seem to pose a big challenge to the controllers. Constraint violations of this state happen but, in general, well after the controllers are not anymore capable of keeping T_1 or T_2 close to their upper bound.

For the SL-based controllers, Fig. 9 shows the number of iterations that the SL algorithm requires converging at each time step for the three test cases. As the algorithm uses the solution from the previous iteration as the initial guess for the next iteration, in many time steps, very few iterations are required for convergence, particularly in Cases 1 and 2. The unknown disturbances in Case 3 naturally cause an increase in the number of iterations since the optimal trajectory frequently deviates considerably from the initial guess due to the frequent disturbances.

The overall performance of the controllers is summarized in Fig. 10. The NMPC, with a very low prediction horizon, consistently performs worse than other controllers. The LMPC performs better than the SLMPC, except for the thermal endurance in Case 3. Apparently, the much shorter horizon of the SLMPC nullifies the potential benefits of solving an NLP, as opposed to solving an LP with a long horizon with the LMPC. Notably, the H-SLMPC has a better performance in all aspects for all test cases: the difference in thermal endurance compared to the next best controller ranges from 21% to 84% through all test cases. For the ATV, the H-SLMPC performs better by 37%–67%. As discussed earlier, the sampling time of each controller was chosen such that they are real-time implementable, which is confirmed in Fig. 10 (the only exception is with the NMPC for Case 3, which has a maximum computation time slightly above the sampling time). Fig. 10 essentially shows how the H-SLMPC captures the best characteristics of the other controllers: it has a prediction horizon covering the whole mission, solves optimization problems with the full nonlinear model at each iteration, and uses very little of the available computation time.

VI. Conclusion

A nonlinear hierarchical MPC framework was presented to maximize the thermal endurance of aircraft through control of the FTMS. Due to the multitimescale nonlinear dynamics of this system and the need for long prediction horizons and fast controller updates, an NMPCformulation solved using general-purpose nonlinear programming solvers fails to effectively maximize thermal endurance. Therefore, a nonlinear hierarchical approach was proposed, where SL is used to solve the nonlinear optimization problems for both the upper and lower level controllers through iteratively linearizing the dynamics and solving LPs. In a series of simulated scenarios, the SL hierarchical MPC outperforms centralized linear, nonlinear, and SLMPCcontrollers in terms of larger thermal endurance, fewer constraint violations, and reduced computational cost. Motivated by the practical performance of SL hierarchical MPC, future work will focus on developing theoretical guarantees for robust constraint satisfaction and additional applications exhibiting multitimescale nonlinear dynamics.

Appendix
Derivation of Linear Model With Slow Sampling Time

This section shows how an LTV model with a slow sampling time can be obtained from the matrices of a model with a fast sampling time. Here, the superscripts s and f are used for variables related to a model with *slow* and *fast* sampling times, respectively.

Consider a discrete-time LTV model with slow sampling time Δt^s, with $k = 0, \ldots, N_s$

$$\begin{aligned} x_{k+1} &= x_k + \Delta t^s \left[A_k^s x_k + B_k^s u_k + W_k^s d_k + c_k^s \right] \\ &= \bar{A}_k^s x_k + \bar{B}_k^s u_k + \bar{W}_k^s d_k + \bar{c}_k^s \end{aligned} \tag{14a}$$

where $\bar{A}_k^s = (I + \Delta t^s A_k^s)$, $\bar{B}_k^s = \Delta t^s B_k^s$, $\bar{W}_k^s = \Delta t^s W_k^s$, and $\bar{c}_k^s = \Delta t^s c_k^s$.

Similarly, consider the model of the same system with fast sampling time Δt^f, with $k = 0, \ldots, N_f$

$$x_{k+1} = \bar{A}_k^f x_k + \bar{B}_k^f u_k + \bar{W}_k^f d_k + \bar{c}_k^f \tag{15}$$

with $\bar{A}_k^f = (I + \Delta t^f A_k^f)$, $\bar{B}_k^f = \Delta t^f B_k^f$, $\bar{W}_k^f = \Delta t^f W_k^f$, and $\bar{c}_k^f = \Delta t^f c_k^f$.

Let $r = (\Delta t^s / \Delta t^f)$ be the ratio of the slow to the fast sampling time, with $r \in \mathbb{Z}_+$. If, for the fast model, u_k is considered to be constant between corresponding updates of the slow model, i.e., u_k can only change its value at every r step of the fast model, then the following relations are true:

$$\bar{A}_k^s = \prod_{i=k\cdot r}^{(k+1)\cdot r-1} A_i^f \tag{16a}$$

$$\bar{B}_k^s = \bar{B}_{(k+1)\cdot r-1}^f + \sum_{i=1}^{r-1} \left(\prod_{m=(k+1)\cdot r-i}^{(k+1)\cdot r-1} \bar{A}_m^f \right) \bar{B}_{(k+1)\cdot r-i-1}^f \tag{16b}$$

$$\bar{W}_k^s = \bar{W}_{(k+1)\cdot r-1}^f + \sum_{i=1}^{r-1} \left(\prod_{m=(k+1)\cdot r-i}^{(k+1)\cdot r-1} \bar{A}_m^f \right) \bar{W}_{(k+1)\cdot r-i-1}^f \tag{16c}$$

$$\bar{c}_k^s = \bar{c}_{(k+1)\cdot r-1}^f + \sum_{i=1}^{r-1} \left(\prod_{m=(k+1)\cdot r-i}^{(k+1)\cdot r-1} \bar{A}_m^f \right) \bar{c}_{(k+1)\cdot r-i-1}^f. \tag{16d}$$

Acknowledgment

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Office of Naval Research.

References

[1] D. D. Leister and P. J. Koeln, "Nonlinear hierarchical MPC for maximizing aircraft thermal endurance," in *Proc. Dyn. Syst. Control Conf.*, vol. 1, Oct. 2020, pp. 1–10.

[2] D. B. Doman, "Fuel flow topology and control for extending aircraft thermal endurance," *J. Thermophys. Heat Transf.*, vol. 32, no. 1, pp. 35–50, Jan. 2018.

[3] C. H. Pangborn, E. J. Hey, O. T. Deppen, G. A. Alleyne, and S. T. Fisher, "Hardware-in-the-loop validation of advanced fuel thermal management control," *J. Thermophys. Heat Transf.*, vol. 31, no. 4, pp. 901–909, 2017.

[4] P. G. Huang and D. B. Doman, "Thermal management of single- and dual-tank fuel-flow topologies using an optimal control strategy," *J. Thermal Sci. Eng. Appl.*, vol. 10, no. 4, Aug. 2018, Art. no. 041019.

[5] G. P. Huang, D. B. Doman, M. W. Oppenheimer, A. Tipton, and D. O. Sigthorsson, "Control of a switched mode fuel thermal management system," *J. Thermophys. Heat Transf.*, vol. 36, no. 1, pp. 13–27, Jan. 2022.

[6] H. C. Pangborn, J. P. Koeln, M. A. Williams, and A. G. Alleyne, "Experimental validation of graph-based hierarchical control for thermal management," *J. Dyn. Syst., Meas., Control*, vol. 140, no. 10, Jun. 2018, Art. no. 101016.

[7] J. P. Koeln, H. C. Pangborn, M. A. Williams, M. L. Kawamura, and A. G. Alleyne, "Hierarchical control of aircraft electro-thermal systems," *IEEE Trans. Control Syst. Technol.*, vol. 28, no. 4, pp. 1218–1232, Jul. 2020.

[8] R. Scattolini, "Architectures for distributed and hierarchical model predictive control—A review," *J. Process Control*, vol. 19, no. 5, pp. 723–731, May 2009.

[9] J. Koeln, V. Raghuraman, and B. Hencey, "Vertical hierarchical MPC for constrained linear systems," *Automatica*, vol. 113, Mar. 2020, Art. no. 108817.
[10] J. P. Koeln and A. G. Alleyne, "Two-level hierarchical mission-based model predictive control," in *Proc. Annu. Amer. Control Conf. (ACC)*, Jun. 2018, pp. 2332–2337.
[11] R. Scattolini, P. Colaneri, and D. De Vito, "A switched MPC approach to hierarchical control," *IFAC Proc. Volumes*, vol. 41, no. 2, pp. 7790–7795, Jan. 2008.
[12] M. Farina, X. Zhang, and R. Scattolini, "A hierarchical MPC scheme for coordination of independent systems with shared resources and plug-and-play capabilities," *IEEE Trans. Control Syst. Technol.*, vol. 28, no. 2, pp. 521–532, Mar. 2020.
[13] C. Vermillion, A. Menezes, and I. Kolmanovsky, "Stable hierarchical model predictive control using an inner loop reference model and λ-contractive terminal constraint sets," *Automatica*, vol. 50, no. 1, pp. 92–99, Jan. 2014.
[14] R. Scattolini and P. Colaneri, "Hierarchical model predictive control," in *Proc. 46th IEEE Conf. Decis. Control*, Dec. 2007, pp. 4803–4808.
[15] D. Barcelliy, A. Bemporadz, and G. Ripaccioliy, "Hierarchical multi-rate control design for constrained linear systems," in *Proc. 49th IEEE Conf. Decis. Control (CDC)*, Dec. 2010, pp. 5216–5221.
[16] V. Raghuraman, V. Renganathan, T. H. Summers, and J. P. Koeln, "Hierarchical MPC with coordinating terminal costs," in *Proc. Amer. Control Conf. (ACC)*, Jul. 2020, pp. 4126–4133.
[17] B. Picasso, D. De Vito, R. Scattolini, and P. Colaneri, "An MPC approach to the design of two-layer hierarchical control systems," *Automatica*, vol. 46, pp. 823–831, May 2010.
[18] M. Joševski and D. Abel, "Multi-time scale model predictive control framework for energy management of hybrid electric vehicles," in *Proc. IEEE Conf. Decis. Control*, Dec. 2014, pp. 2523–2528.
[19] M. R. Amini, I. Kolmanovsky, and J. Sun, "Hierarchical MPC for robust eco-cooling of connected and automated vehicles and its application to electric vehicle battery thermal management," *IEEE Trans. Control Syst. Technol.*, vol. 29, no. 1, pp. 316–328, Jan. 2021.
[20] M. Diehl, H. J. Ferreau, and N. Haverbeke, "Efficient numerical methods for nonlinear MPC and moving horizon estimation," in *Nonlinear Model Predictive Control: Towards New Challenging Applications* (Lecture Notes in Control and Information Sciences), L. Magni, D. M. Raimondo, and F. Allgöwer, Eds. Berlin, Germany: Springer, 2009, pp. 391–417.
[21] H. Deng and T. Ohtsuka, "ParNMPC—A parallel optimisation toolkit for real-time nonlinear model predictive control," *Int. J. Control*, vol. 95, no. 2, pp. 390–405, Feb. 2022.
[22] M. E. Chamie and F. Lin, "Successive linearization in model-based control for thermal management systems," in *Proc. Annu. Amer. Control Conf. (ACC)*, Jun. 2018, pp. 1540–1545.
[23] D. Y. Yamashita, I. Vechiu, and J.-P. Gaubert, "A review of hierarchical control for building microgrids," *Renew. Sustain. Energy Rev.*, vol. 118, Feb. 2020, Art. no. 109523.
[24] J. L. Vazquez, M. Bruhlmeier, A. Liniger, A. Rupenyan, and J. Lygeros, "Optimization-based hierarchical motion planning for autonomous racing," in *Proc. IEEE RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 2397–2403.
[25] Q. Hu, J. Xie, and C. Wang, "Dynamic path planning and trajectory tracking using MPC for satellite with collision avoidance," *ISA Trans.*, vol. 84, pp. 128–141, Jan. 2019.
[26] T. Bäthge, M. Kögel, S. D. Cairano, and R. Findeisen, "Contract-based predictive control for modularity in hierarchical systems," *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 499–504, Jan. 2018.
[27] C. Vermillion, J. Sun, and K. Butts, "Predictive control allocation for a thermal management system based on an inner loop reference model—Design, analysis, and experimental results," *IEEE Trans. Control Syst. Technol.*, vol. 19, no. 4, pp. 772–781, Jul. 2011.
[28] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, Jun. 2000.
[29] R. H. Byrd, M E. Hribar, and J. Nocedal, "An interior point algorithm for large-scale nonlinear programming," *SIAM J. Optim.*, vol. 9, no. 4, pp. 877–900, Jan. 1999.
[30] M. Cannon, "Efficient nonlinear model predictive control algorithms," *Annu. Rev. Control*, vol. 28, no. 2, pp. 229–237, Jan. 2004.
[31] M. Szmuk and B. Acikmese, "Successive convexification for 6-DoF Mars rocket powered landing with free-final-time," in *Proc. AIAA Guid., Navigat., Control Conf.*, Jan. 2018, p. 0617.
[32] Z. Wang and M. J. Grant, "Constrained trajectory optimization for planetary entry via sequential convex programming," *J. Guid., Control, Dyn.*, vol. 40, no. 10, pp. 2603–2615, Oct. 2017.
[33] J. Casoliva, "Spacecraft trajectory generation by successive approximation for powered descent and cyclers," Ph.D. thesis, Dept. Mech. Aerosp. Eng., Univ. California, Irvine, CA, USA, 2013.
[34] D. Malyuta, Y. Yu, P. Elango, and B. Acçkmeşe, "Advances in trajectory optimization for space vehicle control," *Annu. Rev. Control*, vol. 52, pp. 282–315, Jan. 2021.
[35] D. Malyuta et al., "Convex optimization for trajectory generation," 2021, *arXiv:2106.09125*.
[36] Y. Mao, M. Szmuk, and B. Açikmeşe, "Successive convexification of non-convex optimal control problems and its convergence properties," in *Proc. IEEE Conf. Decis. Control (CDC)*, Dec. 2016, pp. 3636–3641.
[37] J. Löfberg, "YALMIP: A toolbox for modeling and optimization in MATLAB," in *Proc. IEEE Int. Symp. Comput. Aided Control Syst. Design*, Sep. 2004, pp. 284–289.
[38] *Gurobi Optimizer Reference Manual*, Gurobi Optim., Houston, TX, USA, 2021.

Daniel D. Leister (Graduate Student Member, IEEE) received the B.Sc. degree in electrical engineering and the M.Sc. degree in chemical engineering from the Universidade de São Paulo, São Paulo, Brazil, in 2007 and 2014, respectively. He is currently pursuing the Ph.D. degree in mechanical engineering with The University of Texas at Dallas, Richardson, TX, USA.

From 2007 to 2019, he worked on industrial automation systems at ABB Ltda, São Paulo. His current research involves the development of hierarchical and robust nonlinear model predictive control techniques.

Justin P. Koeln (Member, IEEE) received the B.S. degree in mechanical and aerospace engineering from Utah State University, Logan, UT, USA, in 2011, and the M.S. and Ph.D. degrees in mechanical science and engineering from the University of Illinois at Urbana–Champaign, Champaign, IL, USA, in 2013 and 2016, respectively.

He was an NSF Graduate Research Fellow and a Summer Faculty Fellow with the Air Force Research Laboratory, Wright-Patterson Air Force Base, OH, USA. He is currently an Assistant Professor with the Mechanical Engineering Department, The University of Texas at Dallas, Richardson, TX, USA. His research interests include dynamic modeling and control of thermal management systems, model predictive control, set-based methods, and hierarchical and distributed control for electrothermal systems.

Dr. Koeln was a recipient of the 2022 Office of Naval Research Young Investigator Award.