IAEA nuclear fusion

PAPER • OPEN ACCESS

Hybrid model predictive control techniques for safety factor profile and stored energy regulation while incorporating NBI constraints

OPEN ACCESS
IOP Publishing | International Atomic Energy Agency Nuclear Fusion
Nucl. Fusion **64** (2024) 086052 (15pp) https://doi.org/10.1088/1741-4326/ad5c7f

Hybrid model predictive control techniques for safety factor profile and stored energy regulation while incorporating NBI constraints

Brian Leard*, Zibo Wang, Sai Tej Paruchuri, Eugenio Schuster and Tariq Rafiq

Lehigh University, Bethlehem, PA, United States of America

E-mail: brian.leard@lehigh.edu

Received 15 January 2024, revised 6 June 2024
Accepted for publication 27 June 2024
Published 8 July 2024

Abstract

A novel hybrid Model Predictive Control (MPC) algorithm has been designed for simultaneous safety factor (q) profile and stored energy (w) control while incorporating the pulse-width-modulation constraints associated with the neutral beam injection (NBI) system. Regulation of the q-profile has been extensively shown to be a key factor for improved confinement as well as non-inductive sustainment of the plasma current. Simultaneous control of w is necessary to prevent the triggering of pressure-driven magnetohydrodynamic instabilities as the controller shapes the q profile. Conventional MPC schemes proposed for q-profile control have considered the NBI powers as continuous-time signals, ignoring the discrete-time nature of these actuators and leading in some cases to performance loss. The hybrid MPC scheme in this work has the capability of incorporating the discrete-time actuator dynamics as additional constraints. In nonlinear simulations, the proposed hybrid MPC scheme demonstrates improved q-profile$+w$ control performance for NSTX-U operating scenarios.

Keywords: hybrid system, model predictive control, discrete constraints, pulse-width modulation, mixed integer quadratic programming, safety factor profile

(Some figures may appear in colour only in the online journal)

1. Introduction

The viability of fusion as a commercial energy source requires reaching advanced tokamak scenarios, which are characterized by a high fusion gain, magnetohydrodynamic (MHD) stable operation, and extended confinement. To realize these conditions, it is necessary to achieve precise control over various plasma properties. One such parameter of interest is the safety factor (q) profile, which is closely related to the poloidal magnetic flux. While this q-profile is shaped, it is pivotal to avoid pressure-induced MHD instabilities, which can be accomplished by simultaneous regulation of the plasma stored energy w. Neutral beam injection (NBI) is a vital actuation method for concurrent regulation of these two properties. By introducing high-energy neutral particles into the plasma core, NBI drives current, heat, and torque, making it an effective tool for regulating both magnetic and kinetic properties of the

[*] Author to whom any correspondence should be addressed.

plasma. During a tokamak discharge, the NBI power level behaves in an on/off Boolean style of operation. However, most control schemes that incorporate NBI, such as robust control [1, 2], Lyapunov control [3], and optimal control [4], regard the NBI power level as a continuously varying signal, contrary to its discrete operational nature. The continuous controller signal is converted into a time request for a power pulse using a technique known as pulse-width modulation (PWM). While this is often an effective method for regulation of plasma properties using NBI, in certain scenarios the omission of the discrete on/off actuator dynamics in the control synthesis can lead to a loss of control performance.

One potential control strategy that is capable of incorporating the discrete nature of NBI is model predictive control (MPC). The MPC algorithm utilizes a model of the system dynamics in order to solve a real-time optimization problem that determines the best actuator trajectories to achieve a specific target state. An advantage of MPC lies in its ability to handle time-varying constraints on both actuators and states, giving it an edge in managing complex systems such as tokamak plasmas. Consequently, MPC has found implementation across various plasma control applications, including positional and shape control [5], and density control [6]. There has already been significant effort to control the safety factor profile or a related plasma property using MPC [7–11]. However, like most other control schemes, past work has neglected any discrete actuator dynamics and assumed that all state and input variables are continuous. There has been limited exploration into modifying the conventional MPC algorithm to incorporate the discrete constraints of the NBI actuators. And yet, there has been extensive work discussing modeling and optimization techniques for MPC algorithms that deal with hybrid systems containing both continuous-time and discrete-time variables [12–14]. Indeed, the validity of these hybrid MPC's has already been demonstrated in other fields such as self-driving cars [15], robotics [16], and building energy management systems [17]. Within the field of plasma control, a hybrid MPC scheme has been applied to incorporate the discrete nature of pellet injection while regulating the electron density [18]. This work aims to leverage similar methods to design a hybrid MPC that considers the discrete-time nature of NBI actuators for simultaneous regulation of the safety factor profile and stored energy.

This paper is structured as follows. Section 2 outlines the control-oriented model that is employed by the hybrid MPC to forecast the evolution of the safety factor profile and stored energy. Section 3 explains the design of conventional MPC algorithms for systems with solely continuous variables. Section 4 discusses the limitations of NBI actuators and the PWM conversion process. Section 5 describes how the continuous MPC is modified into the hybrid MPC by incorporating discrete variables. Section 6 assesses the hybrid MPC's performance in comparison to the traditional MPC in a scenario from the National Spherical Torus Experiment (NSTX-U). Section 7 summarizes the study's findings and deliberates on potential future advancements.

2. Control-oriented modeling of the poloidal magnetic flux and stored energy

In this section, a nonlinear model of the evolution of the poloidal magnetic flux and stored energy is introduced. This model will serve as the basis for developing MPC algorithms discussed in subsequent sections.

The safety factor profile describes the pitch of the magnetic field and is closely related to the poloidal magnetic flux Ψ,

$$q(t,\hat{\rho}) = -\frac{\mathrm{d}\Phi}{\mathrm{d}\Psi} = -\frac{\mathrm{d}\Phi}{2\pi\,\mathrm{d}\psi} = -\frac{\frac{\partial\Phi}{\partial\rho}\frac{\partial\rho}{\partial\hat{\rho}}}{2\pi\frac{\partial\psi}{\partial\hat{\rho}}} = -\frac{B_{\phi,0}\rho_b^2\hat{\rho}}{\theta}, \quad (1)$$

where Φ is the toroidal magnetic flux, $B_{\phi,0}$ represents the vacuum toroidal magnetic field at the tokamak's major radius R_0, and $\psi(t,\hat{\rho})$ is the poloidal stream function, i.e. $\psi(t,\hat{\rho}) = \Psi(t,\hat{\rho})/(2\pi)$. The variable θ is defined as the spatial gradient of the poloidal magnetic flux, $\theta \triangleq \frac{\partial\psi}{\partial\hat{\rho}}$. The symbol $\hat{\rho}$ designates the normalized mean effective minor radius, $\hat{\rho} \triangleq \rho/\rho_b$, where $\rho \triangleq \sqrt{\Phi/(B_{\phi,0}\pi)}$ is the mean effective minor radius, and ρ_b is ρ at the plasma boundary. Shown by (1), it is possible to control the evolution of the safety factor profile by controlling the spatial gradient of the poloidal magnetic flux θ.

The evolution of the poloidal magnetic flux is modeled by a one-dimensional partial differential equation (PDE) known as the magnetic diffusion equation (MDE). This equation is derived by combining Ampere's law, Faraday's law, and Ohm's law, while assuming that plasma properties are toroidally axisymmetric and constant along flux surfaces [19]. Any variable that indexes the flux coordinate can be used as a spatial coordinate. In this work, the normalized mean effective minor radius $\hat{\rho}$ is the spatial coordinate. The MDE is represented as,

$$\frac{\partial\psi}{\partial t} = \frac{\eta(T_\mathrm{e})}{\mu_0\rho_\mathrm{b}^2\hat{F}^2}\frac{1}{\hat{\rho}}\frac{\partial}{\partial\hat{\rho}}\left(\hat{\rho}\hat{F}\hat{G}\hat{H}\frac{\partial\psi}{\partial\hat{\rho}}\right) + R_0\hat{H}\eta(T_\mathrm{e})\frac{\langle\bar{j}_\mathrm{ni}\cdot\bar{B}\rangle}{B_{\phi,0}}, \quad (2)$$

$$\frac{\partial\psi}{\partial\hat{\rho}}(t,0) = 0, \qquad \frac{\partial\psi}{\partial\hat{\rho}}(t,1) = -k_{I\mathrm{p}}I_\mathrm{p}(t), \quad (3)$$

where, $\eta(t,\hat{\rho})$ is the plasma resistivity, μ_0 is the vacuum permeability, $T_\mathrm{e}(t,\hat{\rho})$ is the electron temperature, $\bar{B}$ is the magnetic field, $\hat{F}(\hat{\rho})$, $\hat{G}(\hat{\rho})$, and $\hat{H}(\hat{\rho})$ are equilibrium parameters, $\bar{j}_\mathrm{ni}(t,\hat{\rho})$ is the noninductive current drive, and $\langle\,\rangle$ denotes a flux surface average. The variable $k_{I\mathrm{p}}$ is a constant defined as, $k_{I\mathrm{p}} \triangleq \frac{-\mu_0}{2\pi}R_0\hat{G}(1)\hat{H}(1)$, and I_p represents the plasma current. The noninductive current drive is composed of the current source from the actuators (in this case NBI) as well as the self-generated bootstrap current, i.e. $j_\mathrm{ni} = j_\mathrm{nb} + j_\mathrm{bs}$. The MDE is closed using a series of empirically-derived, control-oriented models for n_e, T_e, and j_nb. Additionally, the Spitzer model is used to calculate the plasma resistivity η and the Sauter model is used to calculate the bootstrap current j_bs. Mathematical details on these models can be found in [20]. Since the safety factor profile is inversely proportional to the spatial gradient of the poloidal magnetic flux, θ, it is desired to take the spatial derivative of the MDE to obtain the evolution of θ. This converts the MDE into,

$$\frac{\partial\theta}{\partial t} = \left[\frac{\mathrm{d}C_{f_1}}{\mathrm{d}\hat{\rho}}\theta + \left(C_{f_1} + \frac{\mathrm{d}C_{f_2}}{\mathrm{d}\hat{\rho}}\right)\frac{\partial\theta}{\partial\hat{\rho}} + C_{f_2}\frac{\partial^2\theta}{\partial\hat{\rho}^2}\right]\hat{u}_{\mathrm{df}} + \sum_{l=1}^{L}\frac{\mathrm{d}C_l}{\mathrm{d}\hat{\rho}}\hat{u}_l + \left(\frac{\mathrm{d}C_{\mathrm{bs}}}{\mathrm{d}\hat{\rho}}\frac{1}{\theta} - C_{\mathrm{bs}}\frac{1}{\theta^2}\frac{\partial\theta}{\partial\hat{\rho}}\right)\hat{u}_{\mathrm{bs}}, \qquad (4)$$

subject to the boundary conditions $\theta(t,0)=0$, $\theta(t,1)=-k_{I_\mathrm{p}}I_\mathrm{p}(t)$. In the aforementioned equation, C_{f_1}, C_{f_2}, C_l, and C_{bs} represent functions of $\hat{\rho}$. The variables $\hat{u}_{\mathrm{df}}$, $\hat{u}_l$, and $\hat{u}_{\mathrm{bs}}$ are time-dependent functions of I_p, the line averaged density $\bar{n}_\mathrm{e}$, and the NBI powers $P_{\mathrm{nb},l}$. The subscript l denotes the number of the NBI, such that $l \in \{1,\ 2,\ \ldots,\ L\}$ where L is the total number of available NBI's. The explicit definitions of these terms can be found in [21], however in this work their relationship is succinctly expressed as

$$\hat{\boldsymbol{u}} = f_{\hat{u}}(\boldsymbol{u}), \qquad (5)$$

where $\hat{\boldsymbol{u}} = [\hat{u}_{\mathrm{df}},\ \hat{u}_1, \ldots\ \hat{u}_L,\ \hat{u}_{\mathrm{bs}}]^T$ can be considered the pseudo-inputs and the variable $\boldsymbol{u} = [I_\mathrm{p},\ P_{\mathrm{nb},1}, \ldots,\ P_{\mathrm{nb},L}]^T \in \mathbb{R}^{N\times 1}$ can be considered the physical inputs to the system. Note that since the plasma current and the powers of NBI actuators are inputs, $N = 1 + L$.

The model in (4) is a partial differential equation (PDE). To further simplify the model, it is discretized spatially using the finite difference method. A number of $M+1$ spatial nodes are defined at $\hat{\rho}_m = m/M, m \in \{0,\ 1,\ \ldots,\ M\}$. The variable θ_m denotes θ at the spatial location $\hat{\rho}_m$. It is possible to solve for the exterior nodes θ_0 and θ_M using the boundary conditions. Therefore, the variable $\boldsymbol{\theta}$ is defined as a vector of all interior nodes, i.e. $\boldsymbol{\theta} \triangleq [\theta_1, \theta_3, \ldots \theta_{M-1}] \in \mathbb{R}^{(M-1)\times 1}$. By approximating the spatial derivatives using Taylor series, the PDE (4) is reduced to a series of coupled ordinary differential equations (ODE's), which can be concisely written as

$$\dot{\boldsymbol{\theta}} = f_\theta(\boldsymbol{\theta}, \boldsymbol{u}), \qquad (6)$$

where $\dot{(\cdot)}$ indicates a time derivative.

The plasma stored energy evolution is modeled as an ODE of the form

$$\dot{w} = -\frac{w}{\tau_\mathrm{e}(t)} + P_{\mathrm{tot}}(t) \triangleq f_w(w, \boldsymbol{u}), \qquad (7)$$

where w is the stored energy and τ_e is the confinement time calculated using the IPB98(y,2) scaling law found in [22]. The variable P_{tot} is the total power introduced to the plasma,

$$P_{\mathrm{tot}} = \sum_{l=1}^{L} P_{\mathrm{nb},l} + P_{\mathrm{ohm}} - P_{\mathrm{rad}}, \qquad (8)$$

with P_{ohm} being the Ohmic power and P_{rad} being the radiated power. Mathematical equations for these terms can be found in [20]. It is desired to combine (6) and (7) into one equation, which is done by defining a state variable $\boldsymbol{z} \triangleq [\theta_1,\ \theta_3,\ \ldots,\ \theta_{M-1},\ w]^T \in \mathbb{R}^{M\times 1}$, as well as a corresponding function $F \triangleq [f_\theta^T,\ f_w]^T$. Now, the state dynamics can be succinctly expressed as,

$$\dot{\boldsymbol{z}} = F(\boldsymbol{z}, \boldsymbol{u}). \qquad (9)$$

While this nonlinear model could technically be used by the MPC, it is desired to use optimization techniques that require linear state dynamics. Therefore, the next section will linearize this model before incorporating it into the MPC algorithm.

3. Conventional MPC design

The general MPC algorithm contains three subroutines: prediction, optimization, and receding horizon implementation. The prediction subroutine calculates the response of the system for a predetermined sequence of inputs over a finite time frame, referred to as the prediction horizon. The optimization subroutine iterates on the system response obtained from the prediction routine to solve a constrained optimization problem on a user-defined cost function to determine the optimal sequence of inputs over the prediction horizon. In the receding horizon subroutine, the input value corresponding to the first time step of the optimal input sequence is implemented in the system. At each time step, the MPC algorithm employs these three interconnected subroutines to approach the target state, a process illustrated in figure 1. In this case, the input (green) is varied so that the output (blue) approaches the target (red-dashed). Note that in the figure, during the prediction subroutine the inputs are modified over a period of time known as a control horizon N_{cl}, which can be either equal or shorter than the prediction horizon N_p, and then are kept constant. Once the algorithm identifies the optimal inputs, it only implements the inputs at timestep k, and the control inputs for the future times (green-dashed) are discarded. At the next time step, $k+1$, a new optimization will proceed, identifying the desired inputs for that time step.

The nonlinear model derived in section 2 can be used to predict the states while implementing MPC. However, optimization using nonlinear models can be computationally expensive and therefore are often difficult to implement in a real-time optimization algorithm. To reduce the computational time of the optimization process, the nonlinear system (9) can be linearized around a reference trajectory by employing first-order Taylor series expansion,

$$\dot{\boldsymbol{z}} \approx F(\boldsymbol{z}_{\mathrm{ref}}, \boldsymbol{u}_{\mathrm{ref}}) + \underbrace{\left.\frac{\partial F}{\partial \boldsymbol{z}}\right|_{\boldsymbol{z}_{\mathrm{ref}},\boldsymbol{u}_{\mathrm{ref}}}}_{\hat{A}}(\boldsymbol{z} - \boldsymbol{z}_{\mathrm{ref}}) + \underbrace{\left.\frac{\partial F}{\partial \boldsymbol{u}}\right|_{\boldsymbol{z}_{\mathrm{ref}},\boldsymbol{u}_{\mathrm{ref}}}}_{\hat{B}}(\boldsymbol{u} - \boldsymbol{u}_{\mathrm{ref}}). \qquad (10)$$

Note that the reference vectors $\boldsymbol{u}_{\mathrm{ref}}$ and $\boldsymbol{z}_{\mathrm{ref}}$ represent an equilibrium point of the system, i.e. $F(\boldsymbol{z}_{\mathrm{ref}}, \boldsymbol{u}_{\mathrm{ref}}) = 0$. This establishes a linear time-invariant (LTI) model describing the plasma dynamics. New variables $\tilde{\boldsymbol{z}} \triangleq \boldsymbol{z} - \boldsymbol{z}_{\mathrm{ref}}$, and $\tilde{\boldsymbol{u}} \triangleq \boldsymbol{u} - \boldsymbol{u}_{\mathrm{ref}}$ are defined. Substituting these variables into (10) results in a linear system of the form

$$\dot{\tilde{\boldsymbol{z}}} = \hat{A}\tilde{\boldsymbol{z}}(t) + \hat{B}\tilde{\boldsymbol{u}}(t), \qquad (11)$$

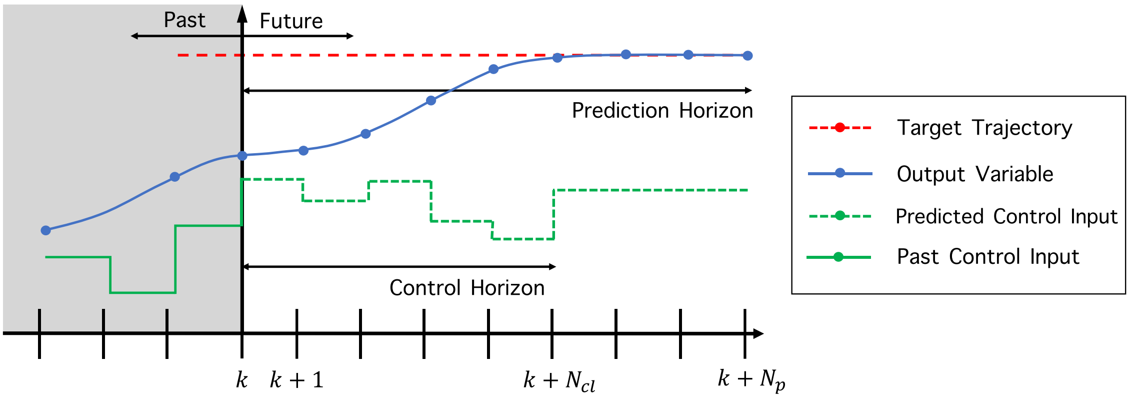

Figure 1. Schematic showing a state and input evolution that are chosen by the MPC over a given prediction horizon, which is N_p timesteps. The input parameters are altered until the timestep N_cl after which they are kept constant.

where $\tilde{z} \in \mathbb{R}^{M\times 1}$ and $\tilde{\boldsymbol{u}} \in \mathbb{R}^{N\times 1}$. In this section, each input can be considered a continuous variable, and a specific input can be denoted $\tilde{u}_n$, where the index $n \in \{1, 2, \ldots, N\}$. However, it is important to note that in later sections a distinction will be made between continuous and discrete inputs. For practical implementation of this model, the system is discretized onto a temporal grid, $t_k = k\Delta t$, $k \in \{0, 1, \ldots\}$, where Δt is a constant time interval. By approximating temporal derivatives as $\dot{\tilde{z}} \approx (\tilde{z}^{k+1} - \tilde{z}^k)/\Delta t$, the system is transformed from continuous-time to a discrete-time framework,

$$\tilde{z}^{k+1} = A\tilde{z}^k + B\tilde{\boldsymbol{u}}^k, \tag{12}$$

where $A = \Delta t\hat{A} + I$ and $B = \Delta t\hat{B}$. The variables $\tilde{z}^k$ and $\tilde{\boldsymbol{u}}^k$ represent the state and input at time t_k.

To mitigate steady-state error, it is desired to incorporate an integrator into the MPC framework. One way to accomplish this is to convert the plant model into velocity form so that the input is the control increment $\Delta\tilde{\boldsymbol{u}}^k$ rather than the control $\tilde{\boldsymbol{u}}^k$. The model in (12) is rewritten into,

$$\Delta\tilde{z}^{k+1} = A\Delta\tilde{z}^k + B\Delta\tilde{\boldsymbol{u}}^k, \tag{13}$$

where, $\Delta\tilde{z}^{k+1} = \tilde{z}^{k+1} - \tilde{z}^k$, and $\Delta\tilde{\boldsymbol{u}}^k = \tilde{\boldsymbol{u}}^k - \tilde{\boldsymbol{u}}^{k-1}$. A new state vector is defined as,

$$\boldsymbol{x} = \left[\Delta\tilde{z}^k, \quad \tilde{z}^k\right]^T, \tag{14}$$

and (13) is combined with the definition of $\Delta\tilde{z}^{k+1}$ to form an enlarged plant,

$$\boldsymbol{x}^{k+1} = \tilde{A}\boldsymbol{x}^k + \tilde{B}\Delta\tilde{\boldsymbol{u}}^k, \tag{15}$$

$$\tilde{z}^k = \tilde{C}\boldsymbol{x}^k, \tag{16}$$

where

$$\tilde{A} = \begin{bmatrix} A & 0_{M\times M} \\ A & I_{M\times M} \end{bmatrix} \in \mathbb{R}^{2M\times 2M}, \quad \tilde{B} = \begin{bmatrix} B \\ B \end{bmatrix} \in \mathbb{R}^{2M\times N},$$
$$\tilde{C} = \begin{bmatrix} 0_{M\times M} \\ I_{M\times M} \end{bmatrix}^T \in \mathbb{R}^{M\times 2M}. \tag{17}$$

The inputs are constrained to model mechanical limitations of the actuators. In this work, the NBI's are subjected to value constraints, and the plasma current is subjected to value and rate constraints,

$$P_\mathrm{nb}^\mathrm{min} \leqslant P_{\mathrm{nb},l} \leqslant P_\mathrm{nb}^\mathrm{max}, \tag{18}$$

$$I_\mathrm{p}^\mathrm{min} \leqslant I_\mathrm{p} \leqslant I_\mathrm{p}^\mathrm{max}, \tag{19}$$

$$\mathrm{d}I_\mathrm{p}^\mathrm{min}/\mathrm{d}t \leqslant \mathrm{d}I_\mathrm{p}/\mathrm{d}t \leqslant \mathrm{d}I_\mathrm{p}^\mathrm{max}/\mathrm{d}t, \tag{20}$$

where $P_\mathrm{nb}^\mathrm{min}$ and $I_\mathrm{p}^\mathrm{min}$ are the minimum possible values of each NBI and the plasma current, and $P_\mathrm{nb}^\mathrm{max}$ and the $I_\mathrm{p}^\mathrm{max}$ are the maximum possible values. Similarly, the variables $\mathrm{d}I_\mathrm{p}^\mathrm{min}/\mathrm{d}t$ and $\mathrm{d}I_\mathrm{p}^\mathrm{max}/\mathrm{d}t$ are the minimum and maximum rates of the plasma current.

By using a numerical approximation of derivatives, these constraints can be written as,

$$\underline{u}_n \leqslant u_n^k \leqslant \overline{u}_n, \tag{21}$$

$$\underline{\mathrm{d}u}_n \leqslant \frac{u_n^k - u_n^{k-1}}{\Delta t} \leqslant \overline{\mathrm{d}u}_n, \tag{22}$$

where $\underline{u}_n$ is the minimum value of the nth input, $\overline{u}_n$ is the maximum value of that input, $\underline{\mathrm{d}u}_n$ is the minimum rate of that input, and $\overline{\mathrm{d}u}_n$ is the maximum rate of that input. The value constraints on each total input value u_n^k can be converted to a constraint on the input to our system $\Delta\tilde{u}_n^k$,

$$\underbrace{\underline{u}_n - u_{\mathrm{ref},n} - \tilde{u}_n^{k-1}}_{\underline{\Delta\tilde{u}_n^k}} \leqslant \Delta\tilde{u}_n^k \leqslant \underbrace{\overline{u}_n - u_{ref,n} - \tilde{u}_n^{k-1}}_{\overline{\Delta\tilde{u}_n^k}}. \tag{23}$$

Note that while the original value constraints (21) were constant in time, the constraints on our system input depend on previous input values, and are thus time-varying. Using a similar conversion, the rate constraint can be expressed as,

$$\underline{\mathrm{d}u}_n\Delta t \leqslant \Delta\tilde{u}_n^k \leqslant \overline{\mathrm{d}u}_n\Delta t. \tag{24}$$

The value and rate constraints expressed in (23), and (24) can be succinctly written as,

$$\mathcal{A}^k\Delta\tilde{\boldsymbol{u}}^k \leqslant \boldsymbol{b}^k, \tag{25}$$

where $\mathcal{A}^k \in \mathbb{R}^{N\times N}$ and $\boldsymbol{b}^k \in \mathbb{R}^{N\times 1}$. Further explanation of this process can be found in [11].

To reach the desired control objectives, the enlarged plant and input constraints are combined with a user-defined cost function. This creates the full MPC problem statement,

$$\min_{\Delta\tilde{\boldsymbol{u}}^i} J = \sum_{i=k}^{N_\mathrm{p}+k-1} \left(\tilde{z}^{i+1} - \tilde{z}^{i+1}_{\mathrm{tar}}\right)^T Q \left(\tilde{z}^{i+1} - \tilde{z}^{i+1}_{\mathrm{tar}}\right) + \left(\Delta\tilde{\boldsymbol{u}}^i\right)^T R \left(\Delta\tilde{\boldsymbol{u}}^i\right) \tag{26}$$

$$\text{s.t } \boldsymbol{x}^{i+1} = \tilde{A}\boldsymbol{x}^i + \tilde{B}\Delta\tilde{\boldsymbol{u}}^i, \quad \tilde{z}^i = \tilde{C}\boldsymbol{x}^i, \tag{27}$$

$$\mathcal{A}^i \Delta\tilde{u}^i \leqslant \boldsymbol{b}^i. \tag{28}$$

The first term in the cost function (26) measures the deviation between the predicted output ($\Delta\tilde{z}^{i+1}$) and target output ($\Delta\tilde{z}^{i+1}_{\mathrm{tar}}$) for each step in the prediction horizon N_p, while the second term measures the control effort. The variables Q and R are positive-definite weight matrices. Solving this optimization problem yields the optimal input values to approach the target output.

In order to speed up computation, it is desired to rewrite the summation of the cost function into matrix form. First, an extended vector is created that includes input values for all time steps in the prediction horizon

$$\Delta\tilde{\boldsymbol{u}}^{k|N_\mathrm{p}} = \left[\Delta\tilde{\boldsymbol{u}}^k, \quad \Delta\tilde{\boldsymbol{u}}^{k+1}, \quad \ldots, \quad \Delta\tilde{\boldsymbol{u}}^{k+N_\mathrm{p}-1}\right]^T \in \mathbb{R}^{N_\mathrm{p}N\times 1}. \tag{29}$$

Then, the constraints for each time step are combined into a singular inequality,

$$\tilde{\mathcal{A}}\Delta\tilde{\boldsymbol{u}}^{k|N_\mathrm{p}} \leqslant \tilde{\boldsymbol{b}}, \tag{30}$$

where $\tilde{\mathcal{A}} = \mathrm{diag}(\mathcal{A}^k, \ldots, \mathcal{A}^{k+N_\mathrm{p}-1}) \in \mathbb{R}^{N_\mathrm{p}N\times N_\mathrm{p}N}$ and $\tilde{\boldsymbol{b}} = [\boldsymbol{b}^k, \ldots, \boldsymbol{b}^{k+N_\mathrm{p}-1}]^T \in \mathbb{R}^{N_\mathrm{p}N\times 1}$. The system dynamics (27) are substituted into the cost function (26) with the summation rewritten in matrix form. The details of this process can be found in appendix A. The original problem statement (26)–(28) can then be expressed as,

$$\min_{\Delta\tilde{\boldsymbol{u}}^{k|N_\mathrm{p}}} J = \frac{1}{2}\left(\Delta\boldsymbol{u}^{k|N_\mathrm{p}}\right)^T H\Delta\boldsymbol{u}^{k|N_\mathrm{p}} + f\Delta\boldsymbol{u}^{k|N_\mathrm{p}} + J_0, \tag{31}$$

$$\text{s.t. } \tilde{\mathcal{A}}\Delta\tilde{\boldsymbol{u}}^{k|N_\mathrm{p}} \leqslant \tilde{\boldsymbol{b}}, \tag{32}$$

where H and f are matrices built from the system dynamics and cost function. The term J_0 is a constant and therefore can be disregarded in the minimization. By rewriting the optimization problem in this form, it is possible to use quadratic programming (QP) to solve the constrained minimization problem. In this work, the Matlab function *quadprog* solves the QP problem via the active-set method.

4. Pulse-width modulation

In the previous section, the MPC algorithm treated each input to the system as a continuous variable. However, as previously mentioned, each NBI actuator operates in discrete binary steps, either delivering a set power level, $P^{\mathrm{max}}_{\mathrm{nb}}$, or no power. Traditionally, control schemes have implemented MPC algorithms such as the one discussed in section 3 and converted the continuous power requests given by the controller to discrete signals implemented by the actuator using a technique known as pulse-width modulation (PWM). Within a fixed time interval, known as a cycling time (t_c), an NBI actuator turns on for a period of time considered a pulse-width request time ($t_{\mathrm{pw}} \leqslant t_\mathrm{c}$), and turns off for the remainder. The NBI does so in such a way so that the average power across the cycle time is equal to the controller's power level request. In this sense, the power requested by the controller is converted to a time request for the actuator,

$$t_{\mathrm{pw}} = \frac{P_{\mathrm{nb}}\, t_\mathrm{c}}{P^{\mathrm{max}}_{\mathrm{nb}}}, \tag{33}$$

where P_{nb} represents the continuous power level requested by the MPC to a generic NBI actuator. Figure 2 shows how PWM converts a power request to a pulse-time request for a specific cycle time. However, due to mechanical limitations, t_{pw} cannot encompass any value between 0 and t_c. Minimum on and off times ($t^{\mathrm{on}}_{\mathrm{min}}$, $t^{\mathrm{off}}_{\mathrm{min}}$) prevent rapid fluctuation of the power that is either mechanically infeasible or potentially harmful to the actuator. In this work, the minimum times will be considered equivalent to each other and one-fourth the cycle time,

$$t^{\mathrm{on}}_{\mathrm{min}} = t^{\mathrm{off}}_{\mathrm{min}} = t_{\mathrm{min}} = t_\mathrm{c}/4. \tag{34}$$

These minimum times restrict the feasible range of t_{pw}, causing it to become piecewise in nature, with a continuous region nested between two discrete variables. Figure 3 explains how the region of t_{pw} is converted from a continuous domain to a piecewise domain. When the requested power is in the range $1/4 \leqslant P_{\mathrm{nb}}/P^{\mathrm{max}}_{\mathrm{nb}} \leqslant 3/4$, the minimum time constraints do not apply. However, outside that range, the PWM algorithm chooses one of two values. In the $0 \leqslant P_{\mathrm{nb}}/P^{\mathrm{max}}_{\mathrm{nb}} < 1/4$ range, t_{pw} is restricted to be either 0 or t_{min}, and in the $3/4 < P_{\mathrm{nb}}/P^{\mathrm{max}}_{\mathrm{nb}} \leqslant 1$ range, t_{pw} is restricted to be either $t_\mathrm{c} - t_{\mathrm{min}}$, or t_c.

The minimum times constraints, coupled with the PWM conversion, can induce notable deviations between the requested power waveform and the delivered power to the actuator, a phenomena that is not considered by the control algorithm of traditional MPC's such as the one outlined in section 3. This phenomena is demonstrated by figure 4, which shows the impact of these minimum time constraints on a given power signal requested by the controller. The continuous power waveform requested by the controller (black dashed) is converted to a discrete signal using the PWM algorithm for three cases with different minimum times, (a): $t_{\mathrm{min}} = 10$ ms, (b): $t_{\mathrm{min}} = 40$ ms, and (c): $t_{\mathrm{min}} = 100$ ms. As expected, as the minimum time (and correspondingly the cycle time) increases, the length of the average pulse increases. The pulses for $t_{\mathrm{min}} = 10$ ms (red shaded) are barely imperceptible, while the pulses for $t_{\mathrm{min}} = 40$ ms (blue shaded) are wider and more noticeable, and this trend continues for $t_{\mathrm{min}} = 100$ ms (purple shaded). When t_{pw} is in the continuous region (green region in figure 3), the time constraints are not in effect, and the average

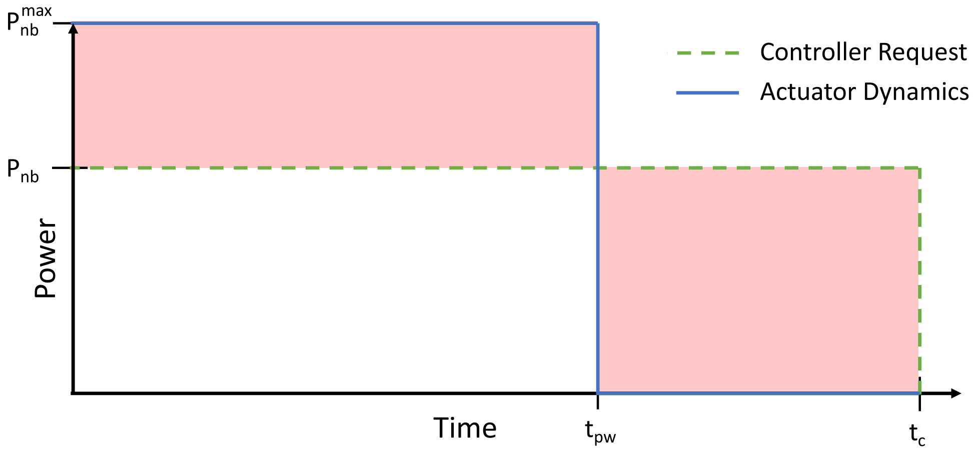

Figure 2. Conversion of the controller's continuous power request (P_{nb}) to a pulse width request time (t_{pw}) for a single cycle time.

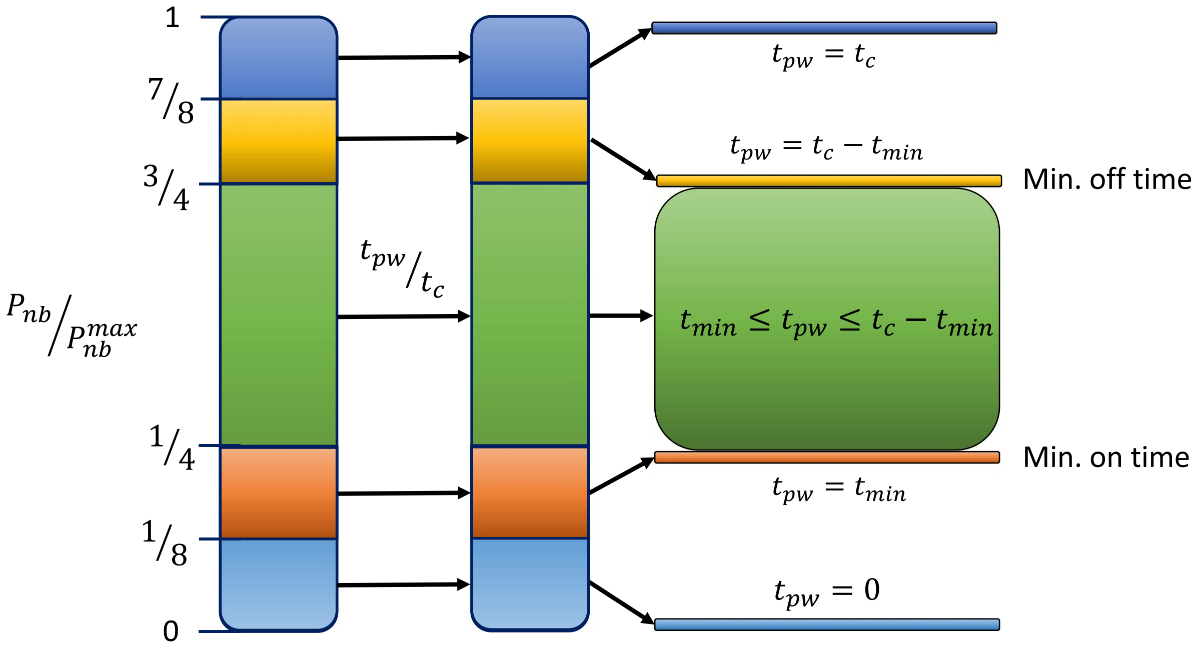

Figure 3. The PWM filtering converts a continuous power level P_{nb} to a request time t_{pw}. However, due to mechanical constraints of NBIs, if t_{pw} falls in either the yellow and dark blue region or the orange and light blue region, it must be rounded to one of two discrete values. This introduces a piecewise nature to the range of t_{pw}.

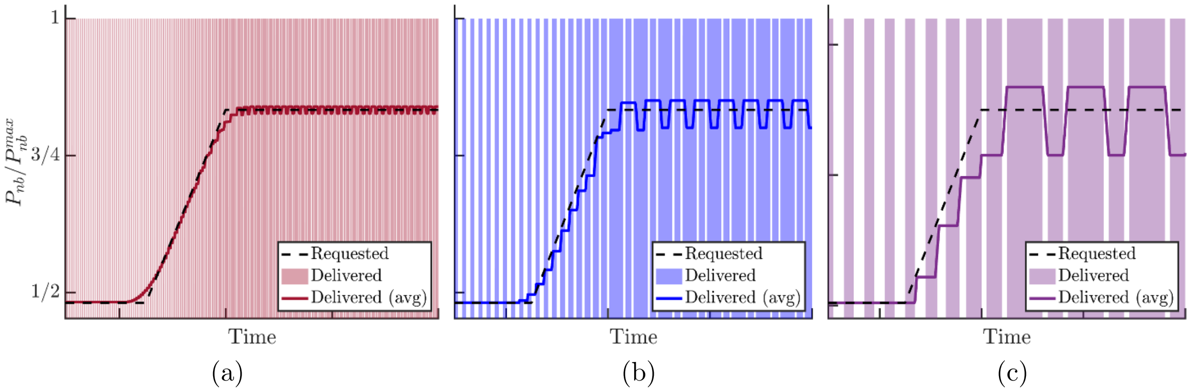

Figure 4. Diagram demonstrating the impact of PWM time constraints on the actuator response. The power requested by the controller is converted to a discrete signal using PWM while increasing the minimum pulse time constraints. The delivered power as well as the average delivered power are shown for: (*a*) $t_{\mathrm{min}} = 10$ ms (*b*) $t_{\mathrm{min}} = 40$ms, (*c*) $t_{\mathrm{min}} = 100$ ms.

PWM power over a single cycle time can match the requested power from the MPC. However, once a power level is requested that causes t_{pw} to leave the continuous region, in this case $P_{\text{nb}}/P_{\text{nb}}^{\max} > 3/4$, the requested power cannot be perfectly recreated due to the minimum time constraints. This causes an oscillation to occur in the average delivered power. The magnitude and frequency of this oscillation is dependent on the size of these time constraints. When the constraints are relatively small, the oscillation is not significant. When $t_{\min} = 10$ ms, the average delivered power (red line) closely tracks the requested power without substantial deviation. However, when $t_{\min} = 40$ ms (blue line), the oscillations have grown in magnitude, and there is now a clear deviation between the requested power and average delivered power for a specific time. When $t_{\min} = 100$ ms (purple line), these oscillations are even larger in both magnitude and pulse width. This demonstrates that, in certain cases, the minimum time constraints lead to a noticeable discrepancy between the controller requested power and the average delivered power, which can potentially diminish the control efficacy. Consequently, a modification of the MPC algorithm has been developed that considers the discrete nature of the NBI actuators with the overall goal of enhancing controller performance.

5. Hybrid MPC design

The hybrid MPC scheme takes the conventional MPC scheme in section 3 and incorporates discrete constraints originating from the operational behavior of NBI actuators. First, it is beneficial to separate the input vector into continuous-time and discrete-time inputs,

$$\Delta\tilde{\boldsymbol{u}}^k = \left[\Delta\tilde{\boldsymbol{u}}_{\text{c}}^k, \quad \Delta\tilde{\boldsymbol{u}}_{\text{d}}^k\right]^T, \tag{35}$$

where $\Delta\tilde{\boldsymbol{u}}_{\text{c}}^k \in \mathbb{R}^{N_{\text{c}}\times 1}$ contains the continuous inputs, and $\Delta\tilde{\boldsymbol{u}}_{\text{d}}^k \in \mathbb{R}^{N_{\text{d}}\times 1}$ contains the discrete inputs. Note that $N_{\text{c}} + N_{\text{d}} = N$, and in this work $N_{\text{c}} = 1$ (only the plasma current is continuous) and $N_{\text{d}} = L$ (the discrete inputs are equal to the number of NBI's). The vector $\boldsymbol{l} = [1, 2, \ldots, L]$ indexes each discrete input (NBI power).

At each time step, for $l \in \{1, \ldots, L\}$, each discrete input is restricted to two values,

$$\Delta\tilde{u}_{N_{\text{c}}+l}^k \in \left\{\underline{\Delta\tilde{u}}_{N_{\text{c}}+l}^k, \overline{\Delta\tilde{u}}_{N_{\text{c}}+l}^k\right\}, \tag{36}$$

which guarantees that the total power of the associated NBI actuator will either be 0 or $P_{\text{nb}}^{\max}$. Note that the variables $\underline{\Delta\tilde{u}}_{N_{\text{c}}+l}^k$ and $\overline{\Delta\tilde{u}}_{N_{\text{c}}+l}^k$ are identical to the minimum and maximum inputs values identified in (23). However, these variables are indexed differently in (36) because this Boolean constraint is only applicable to the discrete inputs.

Next, the minimum time constraints of the actuator are incorporated. This requires memory of prior input values. Specifically, the controller must know t_l^{sw}, the last time step when the lth NBI switched on or off. Before optimizing the inputs, the controller verifies whether the time width of each NBI power pulse has exceeded the minimum pulse time, in other words checks the condition, $t_k - t_l^{\text{sw}} < t_{\min}$. If this condition is true, this input is subject to the time constraint, and the total power of the lth NBI is not varied from the previous time step. Therefore, $\tilde{u}_{N_{\text{c}}+l}^k = \tilde{u}_{N_{\text{c}}+l}^{k-1}$, and consequently $\Delta\tilde{u}_{N_{\text{c}}+l} = 0$. To succinctly express these constraints, new variables are defined,

$$\text{if } t^k - t_l^{\text{sw}} \geqslant t_{\min} \text{ set } s_l^k = 1, \text{ else } s_l^k = 0, \tag{37}$$

$$v_l^k = 1 - s_l^k, \tag{38}$$

which are combined into the vectors,

$$\boldsymbol{s}^k = \left[s_1^k, \quad \ldots, \quad s_L^k\right] \in \mathbb{R}^{1\times L}, \quad \boldsymbol{v}^k = \left[v_1^k, \quad \ldots, \quad v_L^k\right] \in \mathbb{R}^{1\times L}. \tag{39}$$

These vectors are used to create the matrices $\hat{S}^k = \text{diag}\left([0_{1\times N_{\text{c}}}, \boldsymbol{s}^k]\right) \in \mathbb{R}^{N\times N}$, and $\hat{V}^k = \text{diag}\left([0_{1\times N_{\text{c}}}, \boldsymbol{v}^k]\right) \in \mathbb{R}^{N\times N}$. Finally, the matrices $S^k \in \mathbb{R}^{\hat{L}\times N}$ and $V^k \in \mathbb{R}^{(L-\hat{L})\times N}$ are created by combining all non-zero rows of $\hat{S}^k$ and $\hat{V}^k$, where the variable $\hat{L} \leqslant L$ denotes the number of active discrete inputs that are not subject to the time constraint at a particular timestep. The matrix S^k is now associated with these active discrete inputs, effectively picking them from the input vector $\Delta\tilde{\boldsymbol{u}}^k$, and the matrix V^k is associated with all discrete inputs that are subject to the time constraint (and thus removed from the optimization). As the k superscript indicates, these time-constraints are dynamic and therefore the s_l^k and v_l^k variables, as well as their associated matrices, are fluctuating throughout time.

It is also necessary to introduce a vector $\boldsymbol{p}^k$ that indexes the NBIs that are active in the optimization, i.e. not subject to the minimum time constraint. This vector is formed by first multiplying each s_l^k by the corresponding index of the discrete input, which can be succinctly written as,

$$\hat{\boldsymbol{p}}^k = \boldsymbol{s}^k * \boldsymbol{l} \in \mathbb{R}^{1\times L}, \tag{40}$$

where the $(*)$ symbol denotes element-wise multiplication. Then the vector $\boldsymbol{p}^k = [p_1^k, p_2^k, \ldots, p_{\hat{L}}^k] \in \mathbb{R}^{1\times\hat{L}}$ is constructed by removing all zero values from $\hat{\boldsymbol{p}}^k$. The end result is a subset of the vector $\boldsymbol{l}$ that has removed the indices of all discrete inputs currently subject to the time constraint. It is important to note that this $\boldsymbol{p}^k$ vector is both non-sequential and, as indicated by the superscript k, changes at each time step.

Finally, the input constraints (28) are now only relevant for the continuous variables. A new matrix is defined,

$$T = \left[I_{N_{\text{c}}} \quad 0_{N_{\text{c}}\times L}\right] \in \mathbb{R}^{N_{\text{c}}\times N}. \tag{41}$$

By multiplying this matrix T to either side of the inequality (28), the constraint applies only to the continuous variables in the input vector.

The Hybrid MPC problem statement can now be expressed as,

$$\min_{\Delta\tilde{\boldsymbol{u}}^i} J = \sum_{i=k}^{N_\mathrm{p}+k-1} \left(\tilde{\boldsymbol{z}}^{i+1} - \tilde{\boldsymbol{z}}_\mathrm{tar}^{i+1}\right)^T Q \left(\tilde{\boldsymbol{z}}^{i+1} - \tilde{\boldsymbol{z}}_\mathrm{tar}^{i+1}\right) + \left(\Delta\tilde{\boldsymbol{u}}^i\right)^T R \left(\Delta\tilde{\boldsymbol{u}}^i\right) \tag{42}$$

$$\text{s.t}\, \boldsymbol{x}^{i+1} = \tilde{A}\boldsymbol{x}^i + \tilde{B}\Delta\tilde{\boldsymbol{u}}^i, \quad \tilde{\boldsymbol{z}}^i = \tilde{C}\boldsymbol{x}^i, \tag{43}$$

$$T\mathcal{A}^i\,\Delta\tilde{\boldsymbol{b}}^i \leqslant T\boldsymbol{b}^i, \tag{44}$$

$$S^k\Delta\tilde{\boldsymbol{u}}^i \in \left\{\underline{\Delta\tilde{u}}_{\mathrm{p}_1^k}^k, \overline{\Delta\tilde{u}}_{\mathrm{p}_1^k}^k\right\} \times \cdots \times \left\{\underline{\Delta\tilde{u}}_{\mathrm{p}_{\hat{L}}^k}^k, \overline{\Delta\tilde{u}}_{\mathrm{p}_{\hat{L}}^k}^k\right\}, \tag{45}$$

$$V^k\Delta\tilde{\boldsymbol{u}}^i = 0_{(L-\hat{L})\times 1}. \tag{46}$$

This optimization problem statement modifies (26)–(28) by adding constraints related to the Boolean nature of the inputs (45), and the minimum time constraint (46). Additionally, the continuous input constraints (44) now only apply to the continuous inputs, $\Delta\tilde{\boldsymbol{u}}_\mathrm{c}^k$.

The addition of the constraint (45) introduces Boolean variables into the fixed-horizon optimization problem, causing the associated QP problem in section 3 to become a mixed-integer quadratic programming (MIQP) problem. Therefore, new optimization techniques must be employed to efficiently solve this type of problem. The first technique used, Penalty Term Homotopy (PTH), remodels the discrete constraints as a penalty term in the to-be-minimized cost function, converting the MIQP problem to a QP problem. The second technique, Branch and Bound (BNB), methodically introduces the discrete constraints as the algorithm systematically searches segments of the solution space. Detailed descriptions of both of these methods can be found in appendices B and C.

6. Simulation testing in COTSIM

Two hybrid MPC's were developed in this work, one solving the MIQP problem using penalty-term homotopy and another using branch and bound. Both of these hybrid MPC's have been tested against a continuous MPC using pulse width modulation. The Control Oriented Transport SIMulator (COTSIM) [23] was used to simulate the plasma dynamics. The COTSIM model was based on the output of the plasma transport code TRANSP (run 204738S33), which itself was predicated on the NSTX-U discharge 204738 from the 2016 experimental campaign. There are two NBI systems on NSTX-U with three beamlines each, so the model considered six NBI inputs, i.e. $L = 6$, each having a power level of $P_\mathrm{nb}^\mathrm{max} = 2.1$ MW, [24]. As mentioned, $N_\mathrm{c} = 1$, since the plasma current is the only continuous input. The prediction horizon was set to four time steps ahead, i.e. $N_\mathrm{p} = 4$. The weighting matrices were set at $Q = \mathrm{diag}(50I_{M-1},\ 10)$ and $R = 0.01I_N$, where I_{M-1} and I_N were chosen as identity matrices of sizes $M-1$ and N, respectively.

Two simulation studies were run. The goal of each controller was to track a given q-profile+w evolution, which is collectively deemed a target output. For each simulation study, a reference trajectory ($\boldsymbol{u}_\mathrm{ref}, \boldsymbol{z}_\mathrm{ref}$) was created by setting the inputs to constant values and running a COTSIM simulation. Then, a target was created by varying the input waveforms and running a new simulation that produces a different plasma-state evolution for each simulation case. The goal of each MPC is to use this reference trajectory, as discussed in section 2, in order to modify the input variables to recreate the target outputs.

The first simulation study, Test 1, keeps I_p identical to the reference to isolate the influence of the NBI waveform evolutions on the desired targets. The NBIs are turned on at 1 second, at which simultaneously feedback is turned on. The PWM constraints are set to, $t_\mathrm{min} = 50$ ms and $t_\mathrm{c} = 200$ ms. It should be emphasized that these PWM constraints heavily impact the resultant performance of each controller. Figure 5 shows the power waveforms for the 6th NBI, and figure 6 shows the evolution of the safety factor at five spatial locations as well as the stored energy evolution. To demonstrate how the PWM conversion process can reduce the performance of the continuous MPC controller, both the continuous MPC with a hypothetically continuous NBI (blue dashed) and a continuous MPC working with a realistic NBI via PWM (pink) are shown. As expected, the continuous MPC with a continuous NBI is able to nearly perfectly recreate the target actuator waveforms. However this is not possible once PWM is applied, leading to a significant reduction in control performance, as evidenced by the reduced tracking by the pink line in figure 6. Notably, the continuous MPC+PWM exhibits significant oscillatory behavior, a degree of which was expected due to the Boolean nature of the NBI actuator. However, for the case of the stored energy w, this degrades tracking effectiveness to potentially unacceptable levels. The PTH hybrid MPC (green) and the BNB hybrid MPC (cyan) each demonstrate a significant improvement on the continuous MPC+PWM. In the case of the stored energy, oscillatory behavior is completely eliminated and the target is tracked effectively throughout the simulation. The safety factor profile tracking still exhibits a degree of oscillation, however notably these oscillations are diminished in magnitude, become more regular, and are largely absent of any steady state error. Additionally, figure 5 shows that the pulsed waveforms of each hybrid MPC is more regular than the continuous MPC+PWM. While both hybrid MPC's exhibit very similar behavior in steady state, the BNB hybrid MPC shows slight improvement over the PTH hybrid MPC in the transient region ($1-2$ s).

The second simulation study, Test 2, modifies the plasma current to demonstrate how each hybrid MPC performs with varying continuous and discrete inputs. The PWM constraints are reduced to $t_\mathrm{min} = 30$ ms, and $t_\mathrm{c} = 120$ ms. The plasma current magnitude constraint was set at 0.5 MW $\leqslant I_\mathrm{p} \leqslant$ 2 MW and the rate constraint is set at -0.2 MW s^{-1} $\leqslant \mathrm{d}I_\mathrm{p}/\mathrm{d}t \leqslant 0.5$ MW s^{-1} . The I_p trajectory used to create the target is purposefully violating the I_p rate constraint, in order to demonstrate how each controller performs when the target is unachievable. Figure 7 shows this plasma current evolution as well as the waveform for one of the NBI powers. Similar to the previous case, the continuous MPC paired with a theoretically continuous NBI closely matches the target trajectories, however the plasma current evolution is not perfectly recreated due to the rate constraint. Figure 8 presents the evolution of

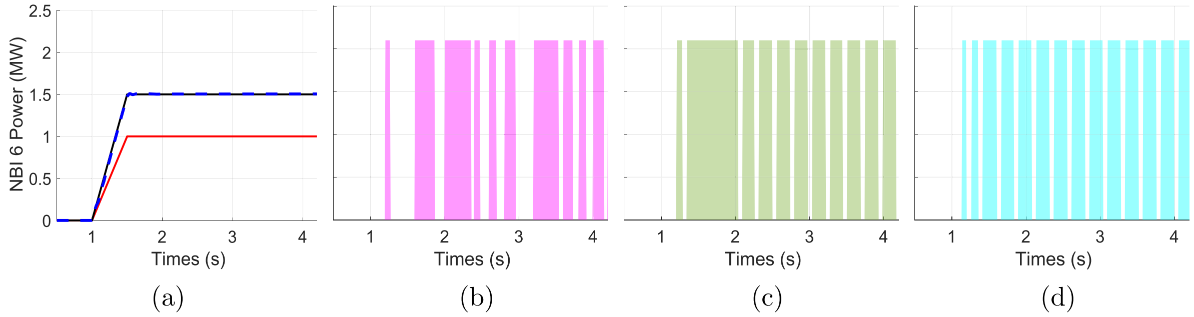

Figure 5. NBI 6 power evolutions: (*a*) shows the reference (red), target (black), and conventional MPC without PWM (blue dashed), (*b*) shows the conventional MPC with PWM (pink), (*c*) shows the PTH hybrid MPC (green), and (*d*) shows the BNB hybrid MPC (cyan).

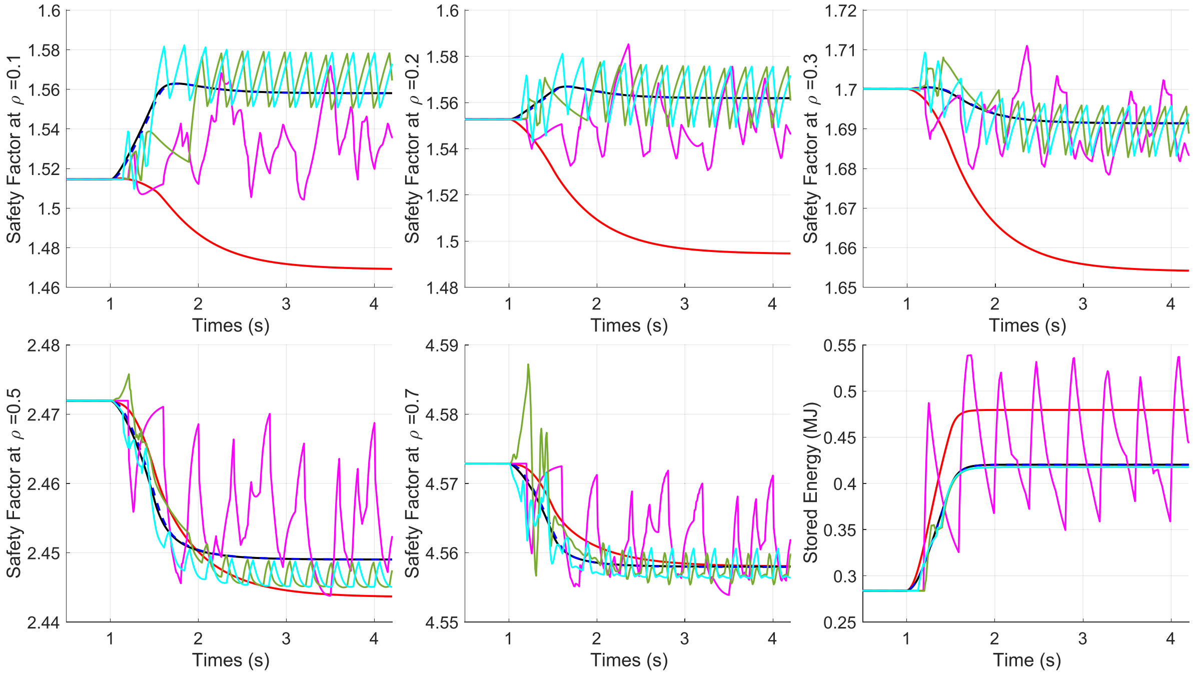

Figure 6. Time evolution of the safety factor at $\hat{\rho} = 0.1$, $\hat{\rho} = 0.2$, $\hat{\rho} = 0.3$, $\hat{\rho} = 0.5$, $\hat{\rho} = 0.7$, and the stored energy for the reference case (red), target (black), conventional MPC without PWM (blue dashed), conventional MPC with PWM (pink), PTH hybrid MPC (green), and BNB hybrid MPC (cyan).

the safety factor at five spatial locations, as well as the stored energy. While the reduction of the PWM constraints leads to some improvement with the conventional MPC+PWM, there is still noticeable oscillatory behavior. Once again, each hybrid MPC demonstrates superior control than the continuous MPC+PWM yet in distinct ways. The PTH hybrid MPC significantly dampens the magnitude of the oscillations caused by the PWM dynamic for both the safety factor profile and the stored energy evolutions, however some oscillatory behavior does persist. The BNB hybrid MPC demonstrates superior dampening of the oscillatory behavior for the stored energy evolution and the safety factor profile toward the plasma edge. However, the dampening of the oscillations for the safety factor toward the very core of the plasma is not significantly improved over MPC+PWM. This behavior deserves more attention and is being further investigated.

For these hybrid MPC schemes to be real-time viable, the computational time of the controller must be below a certain threshold. This work sought to keep the computational times below 10 ms, which was the sampling time of the controllers in this simulation study. Table 1 shows the maximum computational times of each MPC algorithm running the simulation on a Macbook Pro with an M1 Pro chip (10-core CPU, 16-core GPU). It was found that the size of the minimum time constraints did not have significant impact on the maximum computational times. The MPC schemes are solving a singular QP problem at each sampling time, and therefore run faster than the hybrid MPCs, which are each solving multiple QP problems. The continuous MPC runs in less than 1 ms, and the PWM dynamics have only minimal impact. The PTH hybrid MPC meets the 10 ms threshold, while the BNB MPC, which is significantly slower, does not. This is expected, since the PTH algorithm computation time scales $\mathcal{O}(N_\mathrm{d})$ while the BNB algorithm scales $\mathcal{O}(2^{N_\mathrm{d}})$, with N_d being the number of binary decision variables. Consequently, the PTH solver appears more conducive to real-time application. However, this should

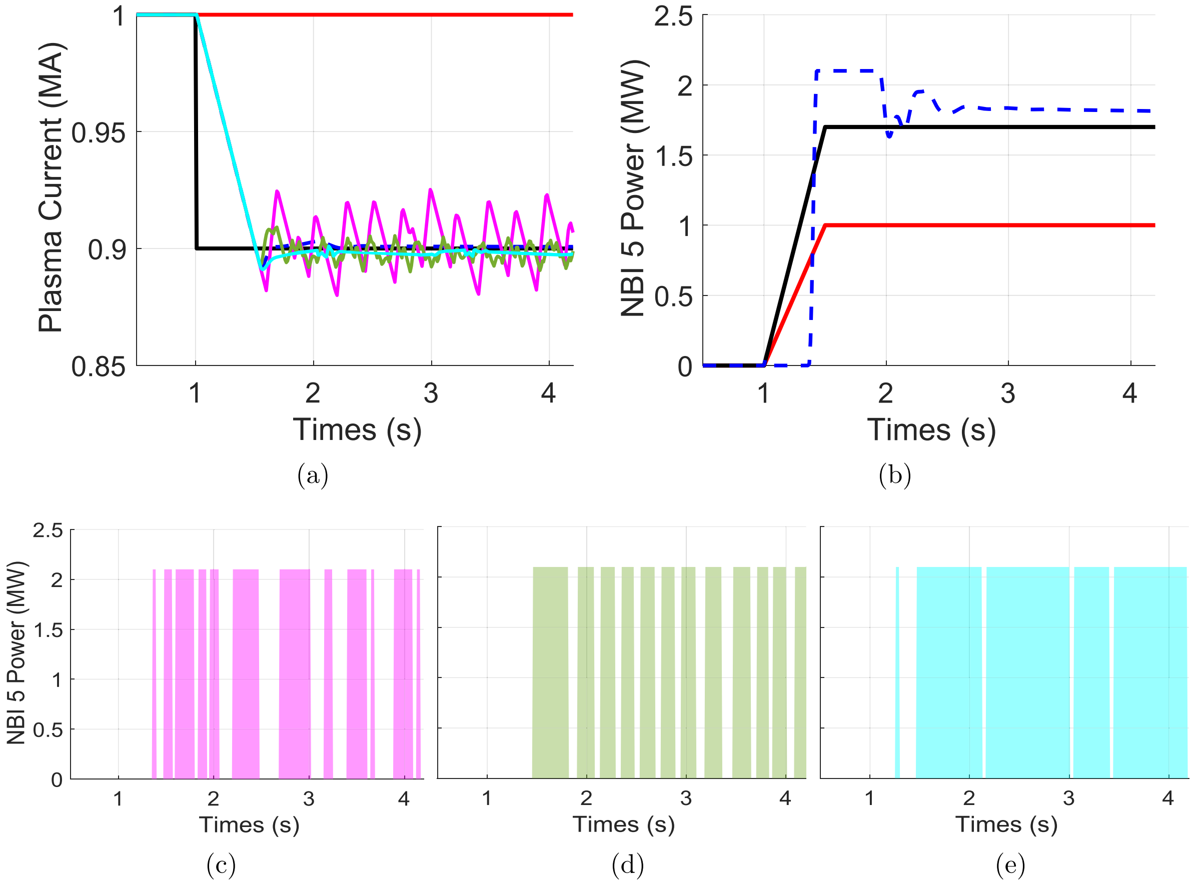

Figure 7. (*a*) Plasma Current evolution, (*b*)–(*d*) NBI 5 power evolutions. Subplot (*b*) shows the reference (red), target (black), and conventional MPC without PWM (blue dashed), (*c*) shows the conventional MPC with PWM (pink), (*d*) shows the PTH hybrid MPC (green), and (*e*) shows the BNB hybrid MPC (cyan).

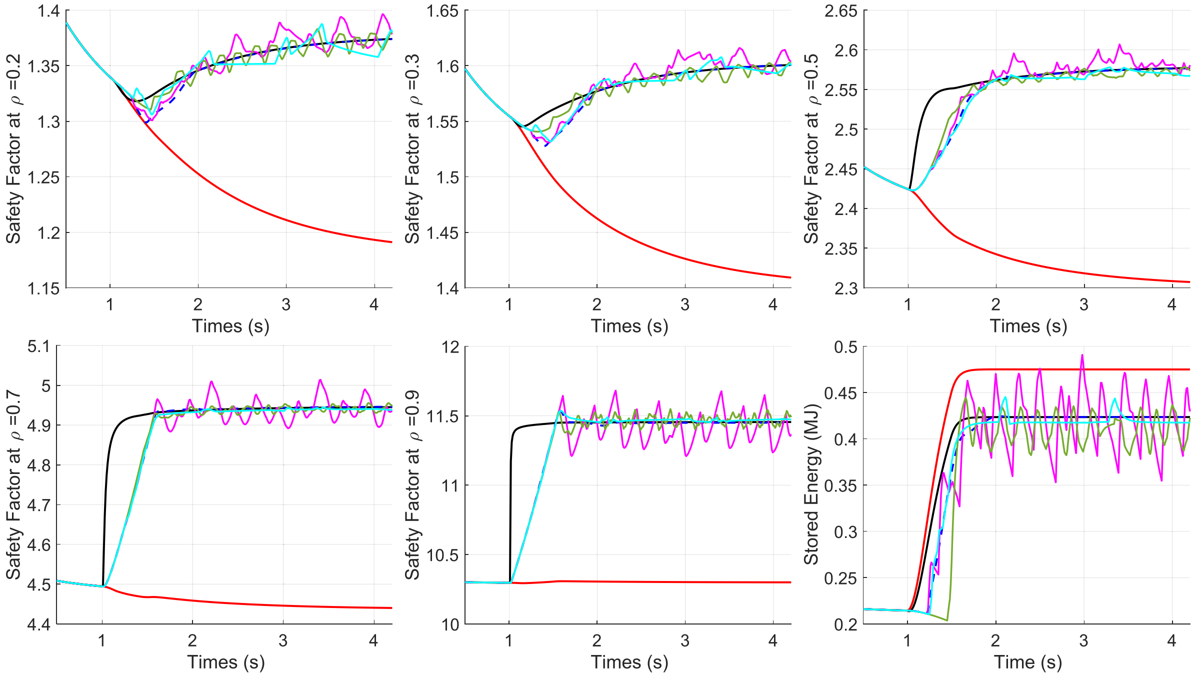

Figure 8. Time evolution of the safety factor at $\hat{\rho} = 0.2$, $\hat{\rho} = 0.3$, $\hat{\rho} = 0.5$, $\hat{\rho} = 0.7$, $\hat{\rho} = 0.9$, and the stored energy for each controller. Note that the legend is the same as figure 6.

Table 1. Computational Times of Each MPC algorithm.

Controller	MPC	MPC+PWM	PTH MPC	BNB MPC
Max Comp. Time	0.8 ms	0.9 ms	6.4 ms	67.5 ms

not exclude the possibility of using the BNB method in real-time. Implementing BNB into the PCS could significantly reduce the computation time of the algorithm. Additionally, techniques such as parallel programming could be exploited to reduce the overall computation time. A larger sampling time could also be used for this control problem. The needed sampling time is dependent on the timescale of the system, and for relatively slow responses (such as magnetic profiles), a larger sampling time is possible. Faster computation, possibly combined with a larger sampling time, could make the BNB approach feasible in real-time.

7. Conclusions and future work

A hybrid MPC strategy has been devised to control the safety factor profile and stored energy evolution while incorporating discrete actuator constraints inherent to NBI. The hybrid MPC accounts for the Boolean operation of the inputs as well as minimum pulse times due to mechanical limitations. The addition of these constraints necessitates modifications to the traditional optimization algorithm to incorporate discrete components of the system. Both penalty term homotopy and branch and bound are used to solve the optimization problem associated with the hybrid system. During simulation studies using COTSIM for an NSTX-U scenario, both hybrid MPC's demonstrate improved control of the target objectives over a continuous MPC+PWM. The hybrid MPC schemes significantly reduce any oscillatory behavior and eliminate steady state error if it occurred. Future work involves testing these hybrid MPC algorithms on an experimental case for NSTX-U. Additionally, by including transport models for additional properties of the plasma impacted by NBI such as the electron temperature or toroidal rotation, the hybrid MPC schemes presented in this work could be tested on alternative profiles or even for simultaneous profile control. Finally, the control design could be expanded even further by incorporating potential time delays in the NBI system if they are significant.

Acknowledgment

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Fusion Energy under Award Number DE-SC0021385.

Disclaimer

This report was prepared as an account of work sponsored by an agency of the US Government. Neither the US Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the US Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the US Government or any agency thereof.

Appendix A. Conversion of the MPC problem to standard QP form

To solve the fixed-horizon optimal control problem (26)–(28), it is posed in standard QP form. Similar to (29), the states are stacked for each time step in the prediction horizon,

$$\tilde{z}^{k+1|N_\mathrm{p}} = \begin{bmatrix}\tilde{z}^{k+1}, & \tilde{z}^{k+2}, & \dots, & \tilde{z}^{k+N_\mathrm{p}}\end{bmatrix}^T \in \mathbb{R}^{(M*N_\mathrm{p})\times 1}. \quad \text{(A.1)}$$

It is possible to replace the $\tilde{z}^i$ terms in the cost function (26) using the linear model, (27). This is done by creating the state transition matrices,

$$O_N = \begin{bmatrix}\tilde{C} & \tilde{C}\tilde{A} & \tilde{C}\tilde{A}^2 & \dots & \tilde{C}\tilde{A}^{N_\mathrm{p}-1}\end{bmatrix}^T \in \mathbb{R}^{N_\mathrm{p}M\times 2M}, \quad \text{(A.2)}$$

$$F_N = \begin{bmatrix} \tilde{C}\tilde{B} & 0 & 0 & \dots & 0 & 0 \\ \tilde{C}\tilde{A}\tilde{B} & \tilde{C}\tilde{B} & 0 & \dots & 0 & 0 \\ \tilde{C}\tilde{A}^2\tilde{B} & \tilde{C}\tilde{A}\tilde{B} & \tilde{C}\tilde{B} & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & \vdots & & \ddots & \vdots \\ \tilde{C}\tilde{A}^{N_\mathrm{p}-1}\tilde{B} & \tilde{C}\tilde{A}^{N_\mathrm{p}-2}\tilde{B} & \dots & \dots & \tilde{C}\tilde{A}\tilde{B} & \tilde{C}\tilde{B} \end{bmatrix} \in \mathbb{R}^{N_\mathrm{p}M\times N_\mathrm{p}N}. \quad \text{(A.3)}$$

The state dynamics can now be expressed as,

$$\tilde{z}^{k+1|N_\mathrm{p}} = O_N\tilde{A}x^k + F_N\Delta\tilde{u}^{k|N_\mathrm{p}}. \quad \text{(A.4)}$$

New weighting matrices $\tilde{Q}$ and $\tilde{R}$ are created that match the dimensions of $N\Delta\tilde{u}^{k|N_\mathrm{p}}$,

$$\tilde{Q} = \begin{bmatrix} Q & 0 & \dots & 0 \\ 0 & Q & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & Q \end{bmatrix} \in \mathbb{R}^{N_\mathrm{p}M\times N_\mathrm{p}M},$$

$$\tilde{R} = \begin{bmatrix} R & 0 & \dots & 0 \\ 0 & R & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & R \end{bmatrix} \in \mathbb{R}^{N_\mathrm{p}N\times N_\mathrm{p}N}, \quad \text{(A.5)}$$

and the state equation is substituted into the cost function, producing the matrices

$$H = F_N^T \tilde{Q} F_N + \tilde{R}, \tag{A.6}$$

$$f = \left(x^k\right)^T \tilde{A}^T O_N^T \tilde{Q} F_N - \left(\tilde{z}_{\mathrm{tar}}^{k+1|N_\mathrm{p}}\right)^T \tilde{Q} F_N, \tag{A.7}$$

where $H \in \mathbb{R}^{N_\mathrm{p}N \times N_\mathrm{p}N}$ and $f \in \mathbb{R}^{1 \times N_\mathrm{p}N}$. These matrices are then used to calculate the cost function while conducting the minimization of the standard QP problem (31) and (32).

Appendix B. Penalty-term homotopy

Penalty-term homotopy (PTH) incorporates discrete constraints as a large penalty term within the cost function J, which allows for the hybrid problem to be solved using traditional QP [18, 25]. In this sense, these constraints are relaxed, since the hard input constraint on discrete variables (45) is removed. Therefore, discrete inputs could technically take non-viable values. However, as the QP solver searches for a minimum, it will be influenced by the penalty term which will direct the solution to viable input values.

With the addition of the penalty term, the cost function in (42) is transformed into,

$$J_\beta = J + \sum_{i=k}^{N_\mathrm{p}+k-1} \underbrace{\left(\hat{S}^k \Delta\tilde{\boldsymbol{u}}^i - \hat{S}^k \underline{\Delta\tilde{\boldsymbol{u}}}^i\right)^T \hat{\beta}_j \left(\hat{S}^k \overline{\Delta\tilde{\boldsymbol{u}}}^i - \hat{S}^k \Delta\tilde{\boldsymbol{u}}^i\right)}_{\text{Penalty Term}}, \tag{B.1}$$

where $\hat{\beta}_j = \mathrm{diag}(\beta_j, \ldots, \beta_j) \in \mathbb{R}^{N \times N}$, and J is the original cost function in (42). The variable β_j is a scalar penalty term. As will be discussed, this value is methodically iterated upon, and each iteration is indexed using the subscript $j \in \{0, 1, \ldots\}$. Multiplying $\Delta\tilde{\boldsymbol{u}}^i$ by the matrix $\hat{S}^k$ removes all continuous inputs as well as any discrete inputs that are subject to the time constraint, since these inputs are not relevant to the penalty term. If the variable β_j is large, any nonviable discrete input will cause the penalty term to dominate the cost function, dissuading the QP algorithm from choosing that as an optimal solution. However, it is also important to ensure that the penalty term does not cause the influence of the other terms in the cost function to become negligible. Therefore, an iterative procedure is followed, shown in figure B1. The variable β_j is first set at 0, and the QP problem is solved. Each discrete input is then checked to determine if it is sufficiently close to a viable value,

$$|\Delta\tilde{u}^i_{N_\mathrm{c}+l} - \underline{\Delta\tilde{u}}^i_{N_\mathrm{c}+l}| \text{ or } |\Delta\tilde{u}^i_{N_\mathrm{c}+l} - \overline{\Delta\tilde{u}}^i_{N_\mathrm{c}+l}| \leqslant \epsilon, \tag{B.2}$$

where ϵ denotes the acceptable error margin. If this condition is not met for every active discrete input, then β_j is increased using the equation,

$$\beta_j = \beta_0 \beta_v^{j-1}, \tag{B.3}$$

where β_0 and $\beta_v > 1$ are constants that determine the initial value and growth rate of β_j. At the first iteration, $j = 1$, and thus $\beta_j = \beta_0$. With each iteration, j increases by 1, causing β_j to grow at a rate dependent on β_v. This iteration ends once the condition (B.2) is met for each discrete input. At this point, the algorithm has determined optimal inputs that also satisfy the discrete constraints of the system.

To efficiently solve the QP problem at each iteration, the optimization problem incorporating the cost function summation in (B.1) is converted to standard form, similar to (31) and (32). The optimization problem becomes,

$$\min_{\Delta\tilde{\boldsymbol{u}}^{k|N_\mathrm{p}}} \quad J_\beta = \frac{1}{2}\left(\Delta\boldsymbol{u}^{k|N_\mathrm{p}}\right)^T H_\beta \Delta\boldsymbol{u}^{k|N_\mathrm{p}} + f_\beta \Delta\boldsymbol{u}^{k|N_\mathrm{p}} + J_0, \tag{B.4}$$

$$\text{s.t.} \quad \tilde{\mathcal{A}} \Delta\tilde{\boldsymbol{u}}^{k|N_\mathrm{p}} \leqslant \tilde{\boldsymbol{b}}, \tag{B.5}$$

$$\tilde{V}^k \Delta\tilde{u}^{k|N_\mathrm{p}} = 0_{N_\mathrm{p}\left(L-\hat{L}\right)\times 1}, \tag{B.6}$$

where H_β and f_β are H and f modified due to the cost function penalty term,

$$H_\beta = H - 2\tilde{\beta}_j \tag{B.7}$$

$$f_\beta = f + \left(\tilde{S}^k \underline{\Delta\tilde{u}}^{k|N_\mathrm{p}} + \tilde{S}^k \overline{\Delta\tilde{u}}^{k|N_\mathrm{p}}\right)^T \tilde{\beta}_j. \tag{B.8}$$

The matrices $\tilde{V}^k$, $\tilde{S}^k$, and $\tilde{\beta}_j$ are built by stacking V^k, $\hat{S}^k$, and $\hat{\beta}_j$, respectively, for each time step in the prediction horizon,

$$\tilde{V}^k = \underbrace{\begin{bmatrix} V^k & 0 & \ldots & 0 \\ 0 & V^k & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & V^k \end{bmatrix}}_{N_\mathrm{p}\left(L-\hat{L}\right)\times N_\mathrm{p}N}, \quad \tilde{S}^k = \underbrace{\begin{bmatrix} \hat{S}^k & 0 & \ldots & 0 \\ 0 & \hat{S}^k & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \hat{S}^k \end{bmatrix}}_{N_\mathrm{p}N\times N_\mathrm{p}N},$$

$$\tilde{\beta}_j = \underbrace{\begin{bmatrix} \hat{\beta}_j & 0 & \ldots & 0 \\ 0 & \hat{\beta}_j & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \hat{\beta}_j \end{bmatrix}}_{N_\mathrm{p}N\times N_\mathrm{p}N}. \tag{B.9}$$

Since the discrete inputs are now treated as continuous variables, the inequality (B.5) is identical to (32). The equality constraint (B.6) originates from (46) and handles the restrictions imposed by the minimal pulse times.

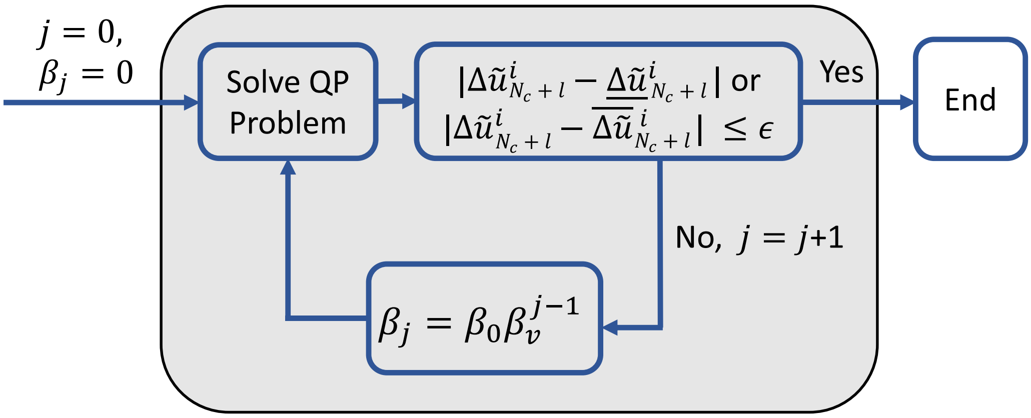
$j = 0$,
$\beta_j = 0$
Solve QP Problem
$|\Delta\tilde{u}^i_{N_c+l} - \underline{\Delta\tilde{u}}^i_{N_c+l}|$ or
$|\Delta\tilde{u}^i_{N_c+l} - \overline{\Delta\tilde{u}}^i_{N_c+l}| \leq \epsilon$
Yes
End
No, $j = j+1$
$\beta_j = \beta_0\beta_v^{j-1}$

Figure B1. Diagram of the PTH iterative procedure. The QP problem is repeatedly solved while increasing the penalty term β_j, until all discrete input variables are sufficiently close to viable inputs (within the threshold ϵ).

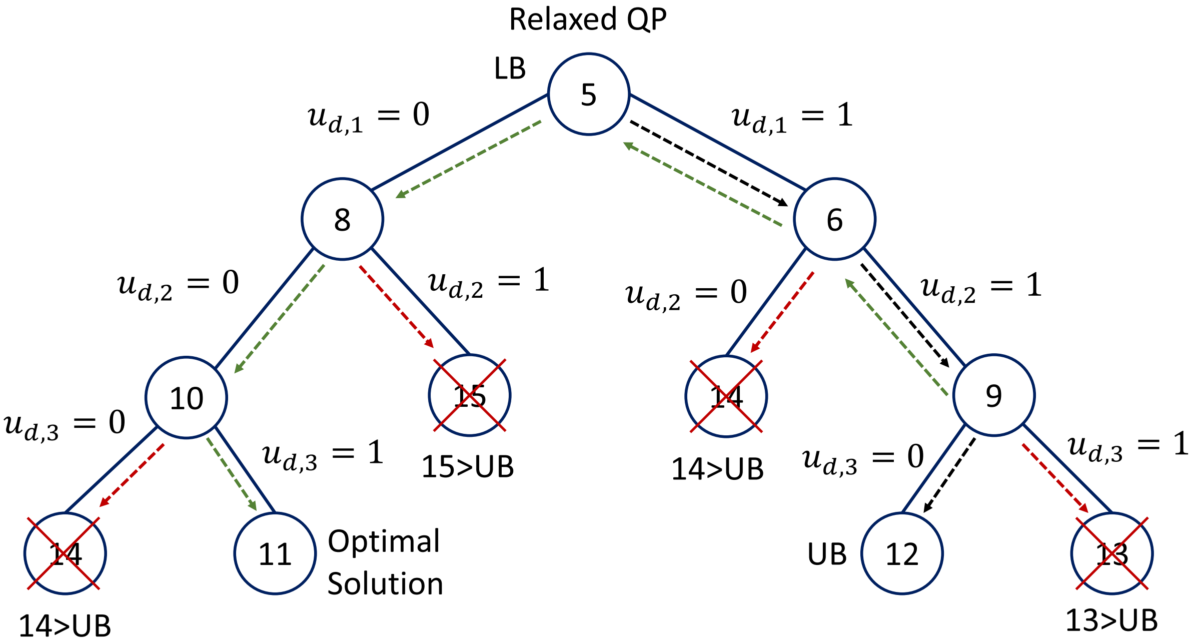

Figure C1. Diagram of the Branch and Bound Procedure. The numbers within the circles indicate the minimum cost function associated with the solution to the QP problem. The algorithm begins with a relaxed QP and slowly unrelaxes discrete variables until it arrives at a viable upper bound (UB) of the solution. The algorithm then systematically searches all unexplored branches, removing any branches whose associated cost function is greater than the UB.

Appendix C. Branch and bound

Branch and bound (BNB) is one of the most common techniques used to solve MIQP problems. It involves systematically varying discrete variables to explore subsets within the solution space, eliminating any subsets (or branches) that display suboptimal performance. To demonstrate this method, figure C1 outlines the procedure for a sample case. In this case, the vector $u = [u_\mathrm{c},\ u_\mathrm{d}]$ are the inputs to a generic system. This vector contains various continuous inputs u_c and three discrete inputs $u_\mathrm{d} = [u_{\mathrm{d},1}, u_{\mathrm{d},2}, u_{\mathrm{d},3}]$ that are limited between 0 and 1. The goal is to find the combination of discrete and continuous inputs that minimize a cost function. This method initially relaxes the integer constraint on each discrete variable, and then solves the resulting QP problem. The cost function associated with the solution establishes a lower bound (LB) on the solution, which in the case of the example is 5. This LB is the minimal cost function of the problem without any discrete constraints considered. Now, the discrete input constraints are methodically introduced one at a time back into the problem, which can be termed ‘unrelaxing’ a variable. The first discrete input $u_{\mathrm{d},1}$ is now set at both 0 and 1. Two branches have now been created, one associated with $u_{\mathrm{d},1} = 0$, and another with $u_{\mathrm{d},1} = 1$. The associated QP problem for each branch is solved, keeping all other discrete variables relaxed. One of these branches will yield a lower associated cost function solution to the QP problem. In this example case, setting $u_{\mathrm{d},1} = 1$ produces a lower minimal cost function value of 6, so the algorithm chooses the $u_{\mathrm{d},1} = 1$ branch to continue exploring, and temporarily disregards all sections of the solution space where $u_{\mathrm{d},1} = 0$. This process is repeated until all discrete variables are unrelaxed. A viable solution has been arrived at, which is now considered an upper bound (UB) for exploring other potential branches. In figure C1, the dashed black line shows the path that the algorithm took to arrive at the upper bound by successively choosing each branch with the lowest associated cost function. Once this upper bound is reached, any previously unexplored branch undergoes reevaluation. If, at any point, the cost function of a branch surpasses the upper bound, that branch and all sub-branches are eliminated. The green dashed line shows how the optimizer retraces its steps to evaluate past branches, and the red dashed line shows how branches and sub-branches are removed once the associated cost function to that branch becomes greater than the upper bound. This cycle continues until all branches have been explored or removed, marking the point at which the optimal solution is identified. In this example case, the optimal solution is $u_{\mathrm{d},1} = 0$, $u_{\mathrm{d},2} = 0$, and $u_{\mathrm{d},3} = 1$ with an associated cost function minimum of 11. Note how this optimal solution is distinct from the upper bound. A more detailed explanation of the procedure can be found in [26].

The hybrid MPC using branch and bound to solve the MIQP problem maintains the general hybrid MPC problem statement, (42)–(46). As discussed, the BNB procedure reduces the MIQP problem to a series of QP problems. To represent one of these QP problems, it is necessary to define the Boolean variable $\lambda_l \in \{1, 0\}$, whose value is determined by which branch the algorithm is currently searching, i.e. whether the lth NBI for this branch is on or off. These variables are combined into a vector,

$$\hat{\boldsymbol{\lambda}}_{p_j} = \begin{bmatrix} 0_{N_\mathrm{c}\times 1}, & \lambda_1 s_1^k, & \ldots, & \lambda_{p_j} s_{p_j}^k, & 0 & \ldots & 0 \end{bmatrix} \in \mathbb{R}^{N\times 1}, \tag{C.1}$$

$$\hat{\Lambda}_{p_j} = \mathrm{diag}\left(\hat{\boldsymbol{\lambda}}_{p_j}\right), \tag{C.2}$$

and $\Lambda_{p_{\hat{l}}}$ is constructed from the nonzero rows of $\hat{\Lambda}_{p_{\hat{l}}}$. Note that variable p_j^k is used to denote a value within the index of active discrete inputs $\boldsymbol{p}^k$. The $\Lambda_{p_{\hat{l}}}$ variable is used to set the active unrelaxed discrete inputs $\Delta\tilde{u}^k_{p_1^k}$ to $\Delta\tilde{u}^k_{p_j^k}$ to a certain combination indicative of the branch. Note that the discrete variables $\Delta\tilde{u}^k_{p_{j+1}^k} \ldots \Delta\tilde{u}^k_{p_{\hat{L}}^k}$ remain relaxed, and are thus treated as continuous variables. At a given branch the QP problem becomes,

$$\min_{\Delta\tilde{\boldsymbol{u}}^{k|N_\mathrm{p}}} J = \frac{1}{2}\left(\Delta\boldsymbol{u}^{k|N_\mathrm{p}}\right)^T H\Delta\boldsymbol{u}^{k|N_\mathrm{p}} + f\Delta\boldsymbol{u}^{k|N_\mathrm{p}} + J_0, \tag{C.3}$$

$$\text{s.t. } \tilde{\mathcal{A}}\Delta\tilde{u}^{k|N_\mathrm{p}} \leqslant \tilde{\boldsymbol{b}}, \tag{C.4}$$

$$\tilde{S}^k_{p_j^k}\Delta\tilde{\boldsymbol{u}}^{k|N_\mathrm{p}} = \tilde{\Lambda}_{p_j^k}\left(\underline{\Delta\tilde{\boldsymbol{u}}}^{k|N_\mathrm{p}}\right) + \left(\tilde{S}^k_{p_j^k} - \tilde{\Lambda}_{p_j^k}\right)\left(\overline{\Delta\tilde{\boldsymbol{u}}}^{k|N_\mathrm{p}}\right), \tag{C.5}$$

$$\tilde{V}^k\Delta\tilde{u}^{k|N_\mathrm{p}} = 0_{N_\mathrm{p}\left(L-\hat{L}\right)\times 1}, \tag{C.6}$$

which is similar to the continuous QP problem, (31) and (32), except for the addition of the binary constraint (C.5) and the time-constraint (C.6). The binary constraint is derived from (45), and is effectively setting the unrelaxed discrete inputs to one of the two allowable values, determined by the Boolean variables in $\tilde{\Lambda}_{p_{\hat{l}}}$. This constraint is the only thing that is changing as the BNB algorithm searches for the optimal values of the discrete inputs. The matrix $\tilde{S}_{p_j^k}$ in (C.5) is composed of stacking the matrices $S_{p_j^k}$,

$$\tilde{S}^k_{p_j^k} = \begin{bmatrix} S^k_{p_j^k} & 0 & \ldots & 0 \\ 0 & S^k_{p_j^k} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & S^k_{p_j^k} \end{bmatrix} \in \mathbb{R}^{N_\mathrm{p}\hat{L}\times N_\mathrm{p}N}, \tag{C.7}$$

and $S^k_{p_j^k}$ is formed from removing the zero rows of $\hat{S}^k_{p_j^k}$,

$$\boldsymbol{s}^k_{p_j^k} = \begin{bmatrix} 0_{N_\mathrm{c}\times 1}, & s_1^k, & \ldots, & s^k_{p_j^k}, & 0 & \ldots & 0 \end{bmatrix} \in \mathbb{R}^{N_\mathrm{p}N\times 1}, \tag{C.8}$$

$$\hat{S}^k_{p_j^k} = \mathrm{diag}\left(\boldsymbol{s}^k_{p_j^k}\right). \tag{C.9}$$

ORCID iDs

Brian Leard https://orcid.org/0000-0003-4141-080X
Zibo Wang https://orcid.org/0000-0003-1762-0663
Sai Tej Paruchuri https://orcid.org/0000-0003-2372-3888
Eugenio Schuster https://orcid.org/0000-0001-7703-6771
Tariq Rafiq https://orcid.org/0000-0002-2164-1582

References

[1] Wang S., Witrant E. and Moreau D. 2021 Robust control of q-profile and β_p using data-driven models on EAST *Fusion Eng. Des.* **162** 112071

[2] Wang H. and Schuster E. 2019 Robust control of the current profile and plasma energy in EAST *Fusion Eng. Des.* **146** 688–91

[3] Pajares A. and Schuster E. 2021 Current profile and normalized beta control via feedback linearization and Lyapunov techniques *Nucl. Fusion* **61** 036006

[4] Ilhan Z.O., Boyer M.D. and Schuster E. 2019 TRANSP-based closed-loop simulations of current profile optimal regulation in NSTX-Upgrade *Fusion Eng. Des.* **146** 555–8

[5] Tartaglione G., Ariola M., Bie W., Di Grazia L.E., Mattei M. and Mele A. 2022 Plasma magnetic control for DEMO tokamak using MPC *2022 IEEE Conf. on Control Technology and Applications (CCTA)* vol 23 pp 825–30

[6] Bosman T.O.S.J., Van Berkel M. and De Baar M.R. 2021 Model-based electron density profile estimation and control, applied to ITER *J. Phys. Commun.* **5** 115015

[7] Wang Z., Wang H., Schuster E., Luo Z., Huang Y., Yuan Q., Xiao B., Humphreys D. and Paruchuri S.T. 2023 Implementation and initial testing of a model predictive controller for safety factor profile and energy regulation in the EAST Tokamak *2023 American Control Conf. (ACC)* pp 3276–81

[8] Maljaars E., Felici F., de Baar M., van Dongen J., Hogeweij G., Geelen P. and Steinbuch M. 2015 Control of the tokamak safety factor profile with time-varying constraints using MPC *Nucl. Fusion* **55** 023001

[9] Ou Y. and Schuster E. 2009 Model predictive control of parabolic PDE systems with dirichlet boundary conditions via Galerkin model reduction *2009 American Control Conf.* pp 1–7

[10] Maljaars B., Felici F., de Baar M. and Steinbuch M. 2015 Model predictive control of the current density distribution and stored energy in Tokamak fusion experiments using trajectory linearizations *5th IFAC Conf. on Nonlinear Model Predictive Control NMPC 2015* vol 48 pp 314–21

[11] Ilhan Z.O., Wehner W.P. and Schuster E. 2016 Model predictive control with integral action for the rotational transform profile tracking in NSTX-U *2016 IEEE Conf. on Control Applications (CCA)* (IEEE) pp 623–8

[12] Camacho E., Ramirez D., Limon D., Muñoz De La Peña D. and Alamo T. 2010 Model predictive control techniques for hybrid systems *Annu. Rev. Control* **34** 21–31

[13] Bemporad A. and Morari M. 1999 Control of systems integrating logic, dynamics and constraints *Automatica* **35** 407–27

[14] Balbis L., Ordys A.W., Grimble M.J. and Pang Y. 2007 Tutorial introduction to the modelling and control of hybrid systems *Int. J. Modelling Identif. Control* **2** 259

[15] Sun X., Cai Y., Wang S., Xu X. and Chen L. 2019 Optimal control of intelligent vehicle longitudinal dynamics via hybrid model predictive control *Robot. Auton. Syst.* **112** 190–200

[16] Hogan F.R. and Rodriguez A. 2020 Reactive planar non-prehensile manipulation with hybrid model predictive control *Int. J. Robot. Res.* **39** 755–73

[17] Khakimova A., Kusatayeva A., Shamshimova A., Sharipova D., Bemporad A., Familiant Y., Shintemirov A., Ten V. and Rubagotti M. 2017 Optimal energy management of a small-size building via hybrid model predictive control *Energy Build.* **140** 1–8

[18] Orrico C.A., van Berkel M., Bosman T.O.S.J., Heemels W.P.M.H. and Krishnamoorthy D. 2023 Mixed-integer MPC strategies for fueling and density control in fusion Tokamaks *IEEE Control Syst. Lett.* **7** 1897–902

[19] Hinton F.L. and Hazeltine R.D. 1976 Theory of plasma transport in toroidal confinement systems *Rev. Mod. Phys.* **48** 239–308

[20] Barton J. *et al* 2015 Physics-model-based nonlinear actuator trajectory optimization and safety factor profile feedback control for advanced scenario development in DIII-D *Nucl. Fusion* **55** 093005

[21] Wang H. 2019 Model-based control of the current density profile in the Experimental Advanced Superconducting Tokamak (EAST) *PhD Dissertation* Lehigh University, Ann Arbor

[22] ITER Physics Expert Groups on Confinement and Transport and Confinement Modelling and Database 1999 ITER Physics Basis Editors *Nucl. Fusion* **39** 2175

[23] Pajares A. 2019 Integrated control in tokamaks using nonlinear robust techniques and actuator sharing strategies *PhD Dissertation* Lehigh University, United States – Pennsylvania

[24] Gerhardt S., Andre R. and Menard J. 2012 Exploration of the equilibrium operating space for NSTX-Upgrade *Nucl. Fusion* **52** 083020

[25] Sager S. 2005 *Numerical Methods for Mixed-Integer Optimal Control Problems* (Der Andere Verlag Lübeck)

[26] Borrelli F., Bemporad A. and Morari M. 2017 *Predictive Control for Linear and Hybrid Systems* (Cambridge University Press)