COMPOSING MPC WITH LQR AND NEURAL NETWORKS
FOR EFFICIENT AND STABLE CONTROL*

FANGYU WU†, GUANHUA WANG†, SIYUAN ZHUANG†, KEHAN WANG†, ALEXANDER
KEIMER†, ION STOICA†, AND ALEXANDRE BAYEN†

Abstract. Model predictive control (MPC) is a powerful control method that handles dynamical
systems with constraints. However, solving MPC iteratively in real time, i.e., implicit MPC, has been
a challenge for 1) systems with low-latency requirements, 2) systems with limited computational
resources, and 3) systems with fast and complex dynamics. To address this challenge, for low-
dimensional linear systems, a classical approach is explicit MPC; for high-dimensional or nonlinear
systems, a common approach is function approximation using neural networks. Both methods,
whenever applicable, may improve the computational efficiency of the original MPC by several orders
of magnitude. The existing methods have the following disadvantages: 1) explicit MPC does not
apply to higher-dimensional problems or most of the problems with nonlinear constraints; and 2)
function approximation is not guaranteed to find an accurate surrogate policy, the failure of which
may lead to closed-loop instability. To address these issues, we propose a triple-mode hybrid control
scheme, named Memory-Augmented MPC, by combining an efficient linear quadratic regulator, an
efficient neural network, and a costly, fail-safe MPC. From its standard form, we derive two variants
of such hybrid control scheme: one customized for chaotic systems and the other for slow systems. We
prove stability of the control scheme with any arbitrary neural networks and test its computational
performance in simulated numerical experiments.

Key words. Model predictive control, computing methodologies, robotics

AMS subject classifications. 93B45, 68U99, 68T40

1. Introduction. Model predictive control (MPC) is a powerful method for
controlling dynamical systems with constraints [25, 4]. It has been widely applied in
robotics, ranging from ground [5] and aerial [1] vehicle control to humanoid [16] and
quadrupe [3] robot control.

To implement MPC, one has two conventional approaches: 1) implicit MPC,
which solves an optimization problem in real time with a preferably efficient problem
formulation and numerical scheme, 2) explicit MPC, which caches every solution of
the optimization problem offline and looks up the cached solutions online.

Explicit MPC tends to have faster running time than implicit MPC. However,
existing explicit methods have limited use cases. Common explicit MPC methods only
handle MPC with quadratic cost and linear constraints. Within the set of applicable
problems, explicit MPC does not work well for MPC of many variables and complex
constraints, since caching solutions of these problems tends to demand a prohibitively
large amount of memory.

Alternatively, one can approximate an MPC with a neural network (NN) through
supervised learning. It develops a surrogate model of MPC by fitting a neural network
into a trajectory dataset generated using implicit MPC. A well-trained neural network
model behaves similarly to the original MPC model and can achieve similar control
effectiveness at a fraction of the MPC’s computational budget. Nevertheless, to our
best knowledge, we find the NN function approximation approach has yet to fully
address the following open problems: 1) it is difficult, if not impossible, to guarantee
that the approximation will always converge to a solution of least approximation
errors; 2) it is challenging to verify that the approximation will produce a stable

*Submitted to the editors December 15, 2021.
†Department of EECS, UC Berkeley, Berkeley, CA. For correspondence, please contact Fangyu
Wu (fangyuwu@berkeley.edu).
closed-loop dynamics without any knowledge of approximation errors between the original and the approximated control laws.

Motivated by the shortcomings of explicit MPC and NN function approximation, we propose a triple-mode hybrid control scheme, named Memory-Augmented MPC (MAMPC). The core idea is to mix a costly MPC controller with an efficient linear quadratic regulator (LQR) controller and an efficient NN controller, whenever stability permits. We summarize the main contributions of our work below.

- We present a novel controller design, i.e., Memory-Augmented MPC, in its standard form and two modified forms.
- We prove stability of Memory-Augmented MPC without imposing any condition on the approximation errors of the neural network.
- We demonstrate computational performance of Memory-Augmented MPC via four simulated robotic control applications.

The rest of the article is organized as follows. A brief literature review is provided in Section 2 to situate the work in context. Terminology and notations of MPC are introduced in Section 3. Standard and modified versions of MAMPC are presented in Section 4. Theoretical properties of MAMPC, including stability, are proven in Section 5. Four numerical experiments are presented in Section 6 and discussed in Section 7. Lastly, major findings and future work are briefed in Section 8.

2. Related Works. A major limitation of MPC has been a lack of computational efficiency as evident in many robotic applications such as the Atlas humanoid robot [16] and the MIT Cheetah robot [3]. To overcome this shortcoming, we have three basic approaches: 1) developing an efficient implicit MPC policy, 2) precompiling an explicit MPC policy, and 3) approximating the MPC policy by an efficient surrogate policy such as a NN.

Implicit MPC approach implements the MPC policy by developing an efficient numerical scheme for solving the optimization program. This is perhaps the most common approach of implementing a MPC policy. Efficient implicit MPC controller often depends on custom optimization solvers that leverage structures specific to the problems they solve, such as [8, 18]. Implementation of such implicit MPC is usually done in a low-level programming language such as C [10, 19]. Despite of being computationally expensive, recent development in algorithms and hardware enables some implicit MPC controllers to achieve high-frequency performance of approximately 1000 Hz [9, 15].

When existing implicit MPC methods do not provide satisfactory latency, one sometimes resorts to explicit MPC, whenever such alternative is applicable. Explicit MPC approach implements the MPC policy by precomputing the solution of the optimization program and looking up the solution in real time. At its core, explicit methods achieve reduced running time at the cost of increased memory footprint. To this end, numerous works has been developed to reduce its space complexity [27, 17]. Nevertheless, due to poor scalability and lack of theory on nonlinear systems, explicit MPC has limited use cases in large-scale robotic applications.

When the explicit MPC method fails to apply to a problem due to scalability or nonlinearity, one may consider another alternative, namely, function approximation. Function approximation approach implements a MPC policy by approximating offline that MPC control law with an efficient surrogate model, such as a neural network, and uses that surrogate model for online deployment. Early works in this direction include [23, 22], which has described a procedure to design a surrogate NN controller and has proposed conditions to guarantee its closed-loop stability. Later development has
focused on guaranteeing different theoretical properties of the NN-controlled closed-loop system. For example, [30] proposes a NN method with probabilistic optimality bounds; [24] has designed a projection operator that can modify any arbitrary NN controller to a stabilizing one; [12] proposes a mixed-integer linear program to certify stability and feasibility of ReLU-activated NN controllers; and [11] has developed a method that guarantees certain performance indicator of a NN-controlled system using probabilistic validation methods. Besides, rather than replacing the original MPC controller, a very recent work [7] uses a NN to warm start an optimization solver, which can then solve the MPC policy faster in real time. For a comprehensive study, readers may consult [31]. Applications of function approximation methods have found successes in practice, such as [21].

It is worth noting that closed-loop stability of the function approximation methods above, except [24], requires reasonable convergence of NN model to the original control law. Nevertheless, training a NN is commonly a high-dimensional nonlinear program. This nonlinear program is solved with faith by a black-box solver, such as [14], which in general does not come with any guarantee on convergence or optimality. Hence, for most of the existing works, their closed-loop stability replies in large part on the faithful assumption that there exists a black-box optimization procedure that can somehow find a good approximation to the original MPC.

In contrary, we propose a novel approach named Memory-Augmented MPC, which can work without any condition on the convergence and optimality of the NN. Compared to the technique in [24], which only applies to discrete-time linear time-invariant plants, our method also works for general nonlinear plants. Concretely, the hybrid controller combines a NN with a costly MPC and an efficient LQR. This hybrid design is an extension of the early dual-mode MPC design [20], which just combines a MPC with a LQR. An advantage of our approach over [20] is that it may provide more reduction in latency owing to the addition of a fast intermediate NN mode. As presented in Section 7, with parallelization, MAMPC is at worse equivalent to [20].

We continue the study by next introducing a basic set of terminology and notations in Section 3.

3. A Primer on MPC. We consider the discrete-time time-invariant constrained dynamical systems of the form

\[ x[i + 1] = f(x[i], u[i]), \quad i \in \mathbb{Z}_{\geq 0} \]

where

\[ x[0] = x_0 \quad \text{for some } x_0 \in \mathbb{R}^n, \]
\[ (x[i], u[i]) \in A, \quad \forall i > 0, \]

where \( A \subset \mathbb{R}^n \times \mathbb{R}^m \) is a closed set defining the system constraints, \( x[i] \) is the state at time index \( i \), \( u[i] \) is the input at time index \( i \), \( f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n \) is continuously differentiable in both its components, with an equilibrium point at \( (0_n, 0_m) \), i.e., \( f(0_n, 0_m) = 0_n \), and the linearized time-invariant system of \( f(\cdot, \cdot) \) around the equilibrium is stabilizable.

Let \( X \) be state constraint set and \( U \) be input constraint set. We consider system constraints of the following form \( A := X \times U \). The system dynamics is a map \( f : A \to X \) and any control law is a map \( u : X \to U \).

Consider a finite planning and control horizon \( N \). Let \( X^{0:N} := (x[0], \ldots, x[N]) \) be a trajectory of the system and \( U^{0:N-1} := (u[0], \ldots, u[N-1]) \) be the corresponding ordered control sequence.
Provided with the dynamical system (3.1), a MPC is an optimal control law implicitly defined through the following set of optimization problems

\[
\min_{x_{i}^{0:N}, u_{i}^{0:N-1}} \sum_{k=0}^{N-1} c(x[k|i], u[k|i]) + c_f(x[N|i]) \\
\text{s.t.: } x[0|i] = x[i], \\
x[k+1|i] = f(x[k|i], u[k|i]), \\
(x[k|i], u[k|i]) \in A, \\
x[N|i] \in X_f, \\
k = 0, \ldots, N - 1,
\]

where \( N \) is planning and control horizon, \( c : A \to \mathbb{R}_+ \) is a continuous stage cost function, \( c_f : X_f \to \mathbb{R}_+ \) is a continuous terminal cost function, \( x[k|i] \) is the predicted state \( k \) steps ahead of present state \( x[i] \), \( u[k|i] \) is the anticipated input that generates \( x[k+1|i] \) from \( x[k|i] \), \( X_f^{1:N} := (x[1|i], \ldots, x[N|i]) \) and \( U_i^{0:N-1} := (u[0|i], \ldots, u[N-1|i]) \) are two sets of optimization variables corresponding to predicted states and anticipated inputs, \( X_f \) is a terminal constraint set containing the origin usually designed to guarantee asymptotic stability. If optimization problem (3.2) is feasible and has an optimal solution \( (X_i^{1:N}, U_i^{0:N-1}) \), then the MPC control law selects the first element of \( U_i^{0:N-1} \) as the input at time index \( i \), that is,

\[
u_{\text{MPC}}(x[i]) := u^*[0|i] = 1^{0:N-1}_i[0].
\]

This process is repeated at the next time index \( i + 1 \), until some termination criterion is met. As a result, we obtain a closed-loop system

\[
x[i+1] = f(x[i], u_{\text{MPC}}(x[i])), \quad i \in \mathbb{Z}_0,
\]

with initial condition \( x[0] = x_0 \in X_0 := \{x_0 \in \mathbb{R}^n \mid \text{problem (3.2) is feasible}\} \) is the set of all admissible initial states. For well-posedness of MPC, we require \( \bar{X} \subseteq X_0 \). Let \( J(x[i]) := \sum_{k=0}^{N-1} c(x[k|i], u[k|i]) + c_f(x[N|i]) \) be the cumulative cost and \( J^*(x[i]) := \sum_{k=0}^{N-1} c(x^*[k|i], u^*[k|i]) + c_f(x^*[N|i]) \) be the optimal cumulative cost. By Lyapunov stability theorem, we can derive the following sufficient condition for asymptotic stability of MPC as in Theorem 12.2 in [4].

**Theorem 3.1 (Local Asymptotic Stability of MPC).** Assume that
- The stage cost and terminal cost are continuous and positive definite.
- \( X_f \) is control invariant.
- For all \( x \in X_f \),

\[
\min_{u \in \mathbb{R}, (x,u) \in X_f} c(x,u) - c_f(x) + c_f(f(x,u)) \leq 0.
\]

then the MPC problem (3.2) is persistently feasible and the closed-loop system (3.4) is locally asymptotically stable with respect to the origin. In addition, \( J^*(\cdot) \) is a Lyapunov function and \( X_0 \) is a positively invariant region of attraction.

With the above definitions and theorem, we are ready to describe the composition rules of MPC, LQR, and NN in Section 4.
4. Composing MPC, LQR, and NN. Provided with a computationally costly implicit MPC \( u_{\text{MPC}} \) that satisfies Theorem (3.1), we propose a hybrid control scheme, namely, Memory-Augmented Model Predictive Control (MAMPC), \( u_{\text{MAMPC}} : X \rightarrow U \), by augmenting \( u_{\text{MPC}} \) with a LQR controller \( u_{\text{LQR}} \) and a NN controller \( u_{\text{NN}} \), where the LQR controller and the NN controller are independently developed from the implicit MPC. In addition, we present two variants of MAMPC specifically tailored for chaotic systems and slow systems, respectively.

4.1. LQR from MPC. Provided with a MPC, we derive an infinite-time LQR controller by 1) linearizing the system dynamics around the equilibrium, 2) removing stage constraints, terminal constraint, and terminal cost, 3) taking planning and control horizon \( N \) to \( \infty \), and 4) replacing stage cost \( c(\cdot) \) with a positive definite quadratic cost, if it is not already so in the original MPC. Formally, the MPC-induced LQR problem is defined as follows

\[
\min_{X^0 \in \mathbb{R}^n, U^0 \in \mathbb{R}^m} \sum_{k=0}^{\infty} x^T[k|i]Qx[k|i] + u^T[k|i]Ru[k|i]
\]

s.t.: 
- \( x[0|i] = x[i] \),
- \( x[k+1|i] = Ax[k|i] + Bu[k|i] \),
- \( k \in \mathbb{Z}_+ \),

where \( A := \frac{\partial f}{\partial x} \bigg|_{(0, 0)} \in \mathbb{R}^{n \times n} \), \( B := \frac{\partial f}{\partial u} \bigg|_{(0, 0)} \in \mathbb{R}^{n \times m} \) are linearized system dynamics around the equilibrium, and \( Q \in \{ N \in \mathbb{R}^{n \times n} \mid N \succeq 0 \} \), \( R \in \{ M \in \mathbb{R}^{m \times m} \mid M \succeq 0 \} \) are weighting matrices that measure significance of state deviations and control costs, respectively.

It can be shown that the optimal LQR control law for an initial state \( x \) is a linear map

\[
u_{\text{LQR}}(x) = -Kx
\]

for some \( K \in \mathbb{R}^{m \times n} \) [2]. Furthermore, if the linearized system \( (A, B) \) is stabilizable and if \( A - BK \) is Hurwitz, then the closed-loop system

\[
x[i+1] = f(x[i], u_{\text{LQR}}(x[i])),
\]

is locally asymptotically stable near the equilibrium with a positively invariant region of attraction \( \mathcal{R}_{\text{LQR}} \), as shown in Theorem 4.7 in [13]. Specifically, \( \mathcal{R}_{\text{LQR}} \) is a set that if \( x_0 \in \mathcal{R}_{\text{LQR}} \) and \( x[0] = x_0 \), then \( x[i] \in \mathcal{R}_{\text{LQR}}, \forall i \in \mathbb{Z}_+ \) and \( \lim_{i \to \infty} x[i] = 0 \). As implied by local asymptotic stability, the largest possible \( \mathcal{R}_{\text{LQR}} \) is nonempty.

Because the LQR controller only requires one matrix multiplication within a single controller step, it is usually significantly more efficient than the implicit MPC in terms of per-step computation. Hence, it is usually economical to switch to the LQR controller when the state is in the region of attraction of the LQR, that is, \( x[i] \in \mathcal{R}_{\text{LQR}} \).

4.2. NN from MPC. Provided with a MPC, we derive a NN controller 1) by imitating the MPC policy \( u_{\text{MPC}}(x) \) through supervised learning and 2) optionally by interacting with the environment through reinforcement learning, as illustrated in Figure 1.
In the *imitation* phase, the NN training is a supervised learning problem usually of the following mean squared form

\[
\min_W \frac{1}{M} \sum_{x \in \mathcal{D}} \left\| u_{\text{MPC}}(x) - \phi(x \mid W) \right\|_2^2,
\]

where \( \phi : \mathbb{R}^n \rightarrow \mathbb{R}^m \) is a NN model with weights \( W \) and \( \mathcal{D} \) is a set of \( M \) states randomly sampled from \( \mathcal{X}_0 \).

Therefore, the resulting NN control law is

\[
u_{\text{NN}}(x) := \phi(x \mid \overline{W}),
\]

where \( \overline{W} \) is a suboptimal solution to problem (4.4), generally estimated via a gradient method such as Adam [14].

In the *adaptation* phase, it is possible to fine tune the NN further through a reinforcement learning (RL) algorithm. The training setup varies based on the specific choice of the RL algorithm. A comprehensive review of reinforcement learning is clearly beyond the scope of this work. Interested readers may refer to the standard text on this topic [28]. Lastly, we remark that this phase may be skipped if the NN trained in the imitation phase is sufficiently effective for the target control application or if the RL algorithm does not bring any improvement to the trained NN.

In practice, people find that implicit MPC controllers can be well approximated by a more efficient NN controller [23]. However, it is not easy to certify stability of the closed-loop system without bounding approximation errors between the NN controller and the original MPC controller.

\[
x[i + 1] = f(x[i], u_{\text{NN}}(x[i])).
\]

To solve the problem, we show that by composing the NN controller with the MPC and the LQR in a hybrid control scheme, we can prove the local asymptotic stability of the closed-loop system at an acceptable cost of computational efficiency, even if the NN is random. Next, we present the basic form of our method, i.e., the standard MAMPC.

### 4.3. Standard Memory-Augmented MPC

The hybrid control scheme combines the LQR controller and the NN controller with the original MPC controller. At every control step, if the state is in the region of attraction of the LQR, apply the LQR controller; else, we simulate the closed-loop system (4.6) for up to \( N_{\text{LQR}} \) steps:
if there exists a step $j \leq N_{\text{LQR}}$ such that the state reaches within the region of attraction of LQR and that up until the $j^{th}$ step the simulated system does not violate any stage constraint of the MPC, apply the NN controller; otherwise, if the state is within the admissible initial states of the MPC, apply the implicit MPC.

Formally, the hybrid controller is defined as follows

$$u_{\text{MAMPC}}(x) := \begin{cases} 
  u_{\text{LQR}}(x), & \text{if } x \in \mathcal{X}_{\text{LQR}}, \\
  u_{\text{NN}}(x), & \exists i = 1, \ldots, N_{\text{LQR}}, \ y[i] \in \mathcal{X}_{\text{LQR}} \text{ and} \\
  \forall j = 0, \ldots, i, \ (y[j], u_{\text{NN}}(y[j])) \in \mathcal{A}, \\
  u_{\text{MPC}}(x), & \text{otherwise,}
\end{cases}$$

where $y$ is the simulated state by numerically stepping: $y[i] = f(y[i-1], u_{\text{NN}}(y[i-1]))$ with $y[0] = x$, $\mathcal{X}_{\text{LQR}}$ is designed to be a subset of $\mathcal{X}_0$, and $N_{\text{LQR}} \in \mathbb{Z}_{>0}$ is the verification horizon of $\mathcal{X}_{\text{LQR}}$.

A pseudocode of the $u_{\text{MAMPC}}(x)$ is described in Algorithm 4.1. As we show in Section 5, the stability of NN relies on verifying whether the system reaches within $\mathcal{X}_{\text{LQR}}$ through forward simulation. However, this approach is ineffective for systems that are sensitive to initial conditions and systems that require a large number of steps to converge to the origin. To address these two challenges, we introduce the following two variants of MAMPC.

### 4.4. Alternating-Authority Memory-Augmented MPC

The first variant of MAMPC is designed specifically for chaotic systems. A chaotic system is loosely defined as a system that is very sensitive to initial condition and control input. Qualitatively speaking, the same chaotic system that begins with two slightly different initial conditions and control sequences will arrive at significantly different terminal states. As a result, it is very challenging to stabilize a chaotic system with any NN controller because the function approximator will inevitably produce random control errors which can easily deter the system from its stabilizing trajectory. In this case, it is useful to modify the hybrid control scheme to bound error accumulation induced by the NN by periodically alternating between the NN controller and the MPC controller. We name this modified version of MAMPC as alternating-authority MAMPC,
which is formally described below.

\[
(4.8) \quad u^{AA}_{\text{MAMPC}}(x) := \begin{cases} 
  u_{\text{LQR}}(x), & \text{if } x \in \mathcal{R}_{\text{LQR}}, \\
  u_{\text{NN}}(x), & \text{if } x \in X_0 \setminus \mathcal{R}_{\text{LQR}} \quad \text{and} \\
  i \pmod{i_d} \neq 0, & \text{otherwise,}
\end{cases}
\]

where \(i_d \in \mathbb{Z}_{\geq 1}\) is the period of MPC defaulting and \(\mathcal{R}_{\text{LQR}} \subset X_0\).

A pseudocode of the \(u^{AA}_{\text{MAMPC}}(x)\) is described in Algorithm 4.2.

**Algorithm 4.2 Alternating-Authority MAMPC**

**Input:** Control \(x\)

**Output:** State \(x\)

1: if \(x \in \mathcal{R}_{\text{LQR}}\) then
2: \(u \leftarrow u_{\text{LQR}}(x)\)
3: else if \(i \pmod{i_d} \neq 0\) and \(u_{\text{NN}}(x)\) keeps \(f\) inside \(X\) without violating \(U\) then
4: \(u \leftarrow u_{\text{NN}}(x)\)
5: else
6: \(u \leftarrow u_{\text{MPC}}(x)\)
7: end if

The main variation from the standard MAMPC is that we do not verify if the system falls within \(\mathcal{R}_{\text{LQR}}\) in \(N_{\text{LQR}}\) steps, as forward simulation is not as reliable for chaotic systems. This implies that \(i_d\) should be chosen as a small positive number. Otherwise, prediction produced by forward simulation will be a reliable estimate of the future. However, note that reducing \(i_d\) will prolong running time. Therefore, a rule of thumb of designing \(i_d\) should be to choose it as large as the stability permits.

In addition, we remark that the closed-loop system has a smaller region of attraction compared to standard MAMPC, since we do not verify whether the system enters \(\mathcal{R}_{\text{LQR}}\) through forward simulation.

**4.5. Way-Point Memory-Augmented MPC.** Another variant of MAMPC is designed specifically for slow systems. A slow system is loosely defined as a system that requires a substantial number of steps to steer close to the origin. For slow systems, it is difficult to choose an effective verification horizon \(N_{\text{LQR}}\): a small \(N_{\text{LQR}}\) may be too short to predict convergence of the NN controller; a large \(N_{\text{LQR}}\) may incur computational overhead and thus render the hybrid control inefficient. To address this problem, we propose a second variant of MAMPC by introducing a “way-point” set, \(\mathcal{D}_{\text{WP}}\), and enabling the NN controller to be applied as long as the anticipated trajectory falls with that way-point set. We name this variant of MAMPC as way-
point MAMPC, which is formulated as follows.

\[
\begin{align*}
    u_{\text{MAMPC}}(x) = \begin{cases} 
    u_{\text{LQR}}(x), & \text{if } x \in R_{\text{LQR}}, \\
    u_{\text{LQR}}(x), & \text{if } x \in D_{\text{WP}} \setminus R_{\text{LQR}} \quad \text{and} \\
    u_{\text{NN}}(x), & \exists i = 1, \ldots, N_{\text{LQR}}, y[i] \in R_{\text{LQR}} \quad \text{and} \\
    \forall j = 0, \ldots, i, \quad (y[j], u_{\text{NN}}(y[j])) \in A_{\text{WP}}, \\
    u_{\text{MPC}}(x), & \text{if } x \in \mathbb{R}^n \setminus D_{\text{WP}} \quad \text{and} \\
    u_{\text{NN}}(x), & \exists i = 1, \ldots, N_{\text{WP}}, y[i] \in D_{\text{WP}} \quad \text{and} \\
    \forall j = 0, \ldots, i, \quad (y[j], u_{\text{NN}}(y[j])) \in A, \\
    u_{\text{MPC}}(x), & \text{otherwise},
    \end{cases}
\end{align*}
\]

where \( D_{\text{WP}} \) is a compact set that contains \( R_{\text{LQR}} \), \( A_{\text{WP}} := D_{\text{WP}} \times U \), and \( N_{\text{WP}} \in \mathbb{Z}_{>0} \) is the verification horizon of \( D_{\text{WP}} \).

Algorithm 4.3 Way-Point MAMPC

**Input:** State \( x \)

**Output:** Control \( u \)

1. if \( x \in R_{\text{LQR}} \) then
2. \( u \leftarrow u_{\text{LQR}}(x) \)
3. else if \( x \in D_{\text{WP}} \setminus R_{\text{LQR}} \) then
4. if \( u_{\text{NN}} \) steers \( f \) into \( R_{\text{LQR}} \) within \( N_{\text{LQR}} \) steps without violating \( A_{\text{WP}} \) then
5. \( u \leftarrow u_{\text{NN}}(x) \)
6. else
7. \( u \leftarrow u_{\text{MPC}}(x) \)
8. end if
9. else
10. if \( u_{\text{NN}} \) steers \( f \) into \( D_{\text{WP}} \) within \( N_{\text{WP}} \) steps without violating \( A \) then
11. \( u \leftarrow u_{\text{NN}}(x) \)
12. else
13. \( u \leftarrow u_{\text{MPC}}(x) \)
14. end if
15. end if
16. return \( u \)

At minimum, we require that the way-point set contains the region of attraction of the LQR, that is, \( R_{\text{LQR}} \subset D_{\text{WP}} \). Additional improvement of the way-point set design can be performed through search. For example, one can parameterize \( D_{\text{WP}} \) with one or more parameters and search for a set of parameters that produces the most efficient closed-loop performance.

Besides, note that introduction of a way-point set may reduce the area of the region of attraction of the closed-loop system.

With the three MAMPC hybrid control schemes defined above, we are ready to analyze their theoretical properties in Section 5.

5. Theoretical Analysis. In this section, we prove the stability of the three MAMPC methods proposed above. In addition, we remark on robustness under model uncertainties and necessity of the fail-safe mode.

5.1. Stability of MAMPC. We show that the three MAMPC methods are locally asymptotically stable with slightly different stability properties.
We first prove that standard MAMPC is locally asymptotically stable in $X_0$, as stated in the theorem below.

**Theorem 5.1 (Stability of Standard MAMPC).** For every MPC problem of the form (3.2), there always exists a $u_{\text{MAMPC}}$ of the form (4.7). Furthermore, the closed-loop system

\[(5.1)\quad x[i + 1] = f(x[i], u_{\text{MAMPC}}(x[i])), \quad i \in \mathbb{Z}_0,\]

with $x[0] = x_0 \in X_0$ is locally asymptotically stable in $X_0$.

**Proof.** Existence of MAMPC depends on existence of LQR and NN. A LQR controller of the form (4.1) can always be derived from (3.2) because 1) $f$ is continuously differentiable, so it can always be linearized to produce the $A, B$ matrices in (4.1); 2) the linearized system $(A, B)$ is by assumption stabilizable, so region of attraction of LQR is nonempty, i.e., $\mathcal{R}_{\text{LQR}} \neq \emptyset$; 3) as for $Q, R$ we only require $Q \succeq 0, R \succ 0$, which is always possible; 4) lastly, optimization problem (4.1) is just a relaxation of problem (3.2), which can always be solved.

Meanwhile, a NN controller trivially exists because we do not require NN to satisfy any properties other than the basic definition of a NN. This completes the first part of the proof on existence.

To prove local asymptotic stability of standard MAMPC in $X_0$, we partition $X_0$ into three regions as follows

$$X_0 = \mathcal{R}_{\text{LQR}} \cup \mathcal{R}_{\text{NN}} \cup \mathcal{R}_{\text{MPC}},$$

where $\mathcal{R}_{\text{NN}}$ is the set of states where the NN is invoked and $\mathcal{R}_{\text{MPC}}$ is the rest of states in $X_0$, i.e., $\mathcal{R}_{\text{MPC}} := X_0 \setminus (\mathcal{R}_{\text{LQR}} \cup \mathcal{R}_{\text{NN}})$. By construction, $\mathcal{R}_{\text{LQR}}, \mathcal{R}_{\text{NN}}, \mathcal{R}_{\text{MPC}}$ are mutually disjoint.

For every state $x[0] \in \mathcal{R}_{\text{LQR}}, u_{\text{LQR}}$ is invoked and the closed-loop system (5.1) will be equivalent to (4.3), which is locally asymptotically stable.
For every state $x[0] \in R_{NN}$, $u_{NN}$ is invoked and the closed-loop system (5.1) will be equivalent to (4.6) until the state enters $R_{LQR}$ in no more than $N_{LQR}$ steps. The system will stay in $R_{LQR}$ and will be taken to the origin by $u_{LQR}$ as $i \to \infty$. Consequently, the closed-loop system (5.1) is locally asymptotically stable in $R_{MPC} \cup R_{NN}$.

For every state $x[0] \in R_{MPC}$, $u_{MPC}$ is invoked and the closed-loop system (5.1) will be equivalent to (3.4) for at least one time step. In the next time step, if the state enters $R_{LQR} \cup R_{NN}$, then the closed-loop system will converge to the origin as it is locally asymptotically stable in $R_{LQR} \cup R_{NN}$; otherwise, $R_{MPC}$ will be invoked again. Because the closed-loop system (3.4) is locally asymptotically stable, even if MPC never brings the state into $R_{NN}$, it will eventually steer the system into $R_{LQR}$ in finite time, which will then be taken to the origin by $u_{LQR}$. Consequently, the closed-loop system (5.1) is locally asymptotically stable in $X_0$.

This completes the second part of the proof on stability.

To provide geometric intuition on how a MAMPC achieves local asymptotic stability, we illustrate schematically in Figure 2a how a MAMPC could steer a system from an initial state to the equilibrium at zero. A state in $R_{MPC}$ is first steered by $u_{MPC}$ into $R_{NN}$, which is then steered by $u_{NN}$ into $R_{LQR}$, after which is steered to zero by $u_{LQR}$. Note that the example demonstrated in Figure 2a is only a hypothetical particular case of MAMPC closed-loop behaviors.

Next, for alternating-authority MAMPC, the closed-loop system no longer has guaranteed local asymptotic stability in $X_0$, because it does not check whether NN shoots inside $R_{LQR}$. Rather, we can only show that the system is locally asymptotically stable in $R_{LQR}$ and will never escape $X_0$, as stated the theorem below.

**Theorem 5.2 (Stability of Alternating-Authority MAMPC).** For every MPC problem of the form (3.2), there always exists a $u_{MAMPC}^{AA}$ of the form (4.8). Furthermore, the closed-loop system

$$x[i + 1] = f(x[i], u_{MAMPC}^{AA}(x[i])), \quad i \in \mathbb{Z}_{\geq 0},$$

with $x[0] = x_0 \in X_0$ is locally asymptotically stable in $R_{LQR}$. Besides, for every initial condition $x[0] \in X_0$, $x[i] \in X_0$ for all $i \in \mathbb{Z}_+.$

**Proof.** The first part of the proof on existence is identical to the one in Theorem 5.1.

To prove stability of alternating-authority MAMPC, we partition $X_0$ into two disjoint regions as follows

$$X_0 = R_{LQR} \cup R_{AA},$$

where $R_{AA} := X_0 \setminus R_{LQR}$.

For every state $x[0] \in R_{LQR}$, the closed-loop system (5.2) is identical to closed-loop system (5.1), which is locally asymptotically stable, as shown in Theorem (5.1).

Next, we prove by contradiction that the system will never escape $X_0$. Suppose there exists an escaping control $\bar{u}$ such that $\bar{x} \in X_0$ but $f(\bar{x}, \bar{u}) \notin X_0$. This escaping control $\bar{u}$ must be produced by either $u_{MPC}$, $u_{NN}$, or $u_{LQR}$, that is, $\bar{u} = u_{MPC}(\bar{x})$, $\bar{u} = u_{NN}(\bar{x})$, or $\bar{u} = u_{LQR}(\bar{x})$. If $\bar{u} = u_{MPC}(\bar{x})$, then $u_{MPC}$ is not persistently feasible in $X_0$, which is not possible because we assume that $u_{MPC}$ satisfies Theorem (3.1). If $\bar{u} = u_{NN}(\bar{x})$, then the following switch condition must hold: $f(\bar{x}, \bar{u}) \in X \subseteq X_0$. However, this is not possible since $\bar{u}$ is assumed to be an escaping control, that is, $f(\bar{x}, \bar{u}) \notin X_0$. If $\bar{u} = u_{LQR}(\bar{x})$, then there must be $\bar{x} \in R_{LQR}$. Since $R_{LQR}$ is
positively invariant, we must have \( f(\bar{x}, \bar{u}) \in \mathcal{R}_{LQR} \subset \mathcal{X}_0 \), which is not possible since \( f(\bar{x}, \bar{u}) \not\in \mathcal{X}_0 \). Therefore, the system will never escape \( \mathcal{X}_0 \).

This completes the second part of the proof on stability.

An illustration of how the alternating-authority MAMPC could steer a system to the equilibrium is provided in Figure 2b. Note that marginal stability may be an understatement in practice: a well-trained NN, combined with a modest alternation period \( i_d \), could very well result in a closed-loop system that is asymptotically stable.

If analytical guarantee of asymptotic stability is absolutely required, one can replace the admissible control set \( \mathcal{X}_0 \) with a time-varying set \( \mathcal{S}[i] \), where

\[
\mathcal{R}_{LQR} \subset \mathcal{S}[i+1] \subset \mathcal{S}[i] \subset \mathcal{X}_0, \quad \forall i = 1, \ldots, n - 1,
\]

for some \( n \in \mathbb{Z}_{\geq 1} \) with \( \mathcal{S}[0] = \mathcal{X}_0 \) and \( \mathcal{S}[n] = \mathcal{R}_{LQR} \). Geometrically, \( \mathcal{S} \) can be viewed as a set that gets successively shrunk from \( \mathcal{X}_0 \) toward \( \mathcal{R}_{LQR} \). Such design will invoke the fail-safe MPC if the systems is “stuck” outside of \( \mathcal{R}_{LQR} \).

Lastly, similar to alternating-authority MAMPC, the closed-loop system of a way-point MAMPC is locally asymptotically stable in \( \mathcal{R}_{LQR} \) and will eventually enter and stay within a level set that describes \( \mathcal{D}_{WP} \), as stated in the following theorem.

**Theorem 5.3 (Stability of Way-Point MAMPC).** For every MPC problem of the form (3.2), there always exists a \( u_{MAMPC}^{WP} \) of the form (4.9). Furthermore, the closed-loop system

\[
\dot{x}[i + 1] = f(x[i], u_{MAMPC}^{WP}(x[i])) , \quad i \in \mathbb{Z}_{\geq 0},
\]

with \( x[0] = x_0 \in \mathcal{X}_0 \) is locally asymptotically stable in \( \mathcal{R}_{LQR} \). Besides, there exists a time index \( j \in \mathbb{Z}_{\geq 0} \) such that for all \( i \geq j \)

\[
x[i] \in \mathcal{D}_{WP}^+ := \{ x \in \mathbb{R}^n | V(x) < \max_{q \in \partial \mathcal{D}_{WP}} V(x) \},
\]

where \( V : \mathbb{R}^n \rightarrow \mathbb{R}_+ \) is a Lyapunov function of the closed-loop system (3.4) and \( \partial \mathcal{D}_{WP} \) is the boundary of the compact set \( \mathcal{D}_{WP} \), which by construction contains \( \mathcal{R}_{LQR} \).

**Proof.** The first part of the proof on existence is identical to the one in Theorem 5.1.

To prove stability of way-point MAMPC, we partition \( \mathcal{X}_0 \) into five disjoint regions as follows

\[
\mathcal{X}_0 = \mathcal{R}_{LQR} \cup \mathcal{R}_{NN}^i \cup \mathcal{R}_{MPC}^i \cup \mathcal{R}_{NN}^o \cup \mathcal{R}_{MPC}^o,
\]

where \( \mathcal{R}_{NN}^i \) is the set of states in or on the boundary of \( \mathcal{D}_{WP} \), where the NN is invoked; \( \mathcal{R}_{NN}^o \) is the set of states in \( \mathcal{X}_0 \setminus \mathcal{D}_{WP} \) where the NN is invoked; \( \mathcal{R}_{MPC}^i := \mathcal{D}_{WP} \setminus (\mathcal{R}_{LQR} \cup \mathcal{R}_{NN}^i) \); and \( \mathcal{R}_{MPC}^o := (\mathcal{X}_0 \setminus \mathcal{D}_{WP}) \setminus \mathcal{R}_{NN}^o \).

For every state \( x \in \mathcal{R}_{LQR} \), the closed-loop system (5.4) is identical to closed-loop system (5.1), which is locally asymptotically stable, as shown in Theorem (5.1).

Next, we prove that the system will eventually enter and stay within \( \mathcal{D}_{WP} \).

For every state \( x[0] \in \mathcal{X}_0 \setminus \mathcal{D}_{WP} \), either NN or MPC will steer the system into \( \mathcal{D}_{WP}^+ \) in finite time. If NN is ever invoked, the system (5.4) will be taken into \( \mathcal{D}_{WP} \) in no more than \( N_{WP} \) steps. Otherwise, MPC will bring the system (5.4) into \( \mathcal{D}_{WP} \) in finite time since the MPC is locally asymptotically stable in \( \mathcal{X}_0 \).

For every state \( x[0] \in \mathcal{D}_{WP} \), we prove by contradiction that there exists a \( j \geq 0 \) such that for all \( i \geq j, x[i] \in \mathcal{D}_{WP}^+ \). Suppose there exists a \( x[0] = \bar{x} \in \mathcal{D}_{WP} \) such
that for every $k \geq 0$, there exists an $i \geq k$ such that $x[i] \notin D^+_{WP}$. Without loss of
generality, define a control $\bar{u}$ such that $x[i-1] \in D^+_{WP}$ but $x[i] = f(x[i-1], \bar{u}) \notin D^+_{WP}$.

This escaping control $\bar{u}$ must be produced by either $u_{MPC}$, $u_{NN}$, or $u_{LQR}$, that is, $\bar{u} = u_{MPC}(x[i-1])$, $\bar{u} = u_{NN}(x[i-1])$, or $\bar{u} = u_{LQR}(x[i-1])$. If $\bar{u} = u_{MPC}(x[i-1])$, then the close-loop system (5.4) is equivalent to system (3.4). Because $x[i] \notin D^+_{WP}$, we have

$$V(x[i]) = V(f(x[i-1], u_{MPC}(x[i-1]))) > V(x[i-1]),$$

but this is not possible because $V$ is a Lyapunov function of the system (3.4), i.e.,

$$V(f(x[i-1], u_{MPC}(x[i-1]))) \leq V(x[i-1]).$$

If $\bar{u} = u_{NN}(x[i-1])$, then $f(x[i-1], u_{NN}(x[i-1])) \notin D_{WP}$, which is not possible because invocation of NN implies that $f(x[i-1], u_{NN}(x[i-1])) \in D_{WP}$. If $\bar{u} = u_{LQR}(x[i-1])$, then by assumption $\lim_{i \to \infty} x[i]$ does not exist. However, this is impossible because $\lim_{i \to \infty} x[i] = 0$ by the local asymptotic stability of the system (5.4) in $\mathcal{R}_{LQR}$. Consequently, the assumption must be false.

This completes the second part of the proof on stability. 

An illustration of how the way-point MAMPC could steer a system to the equilibrium is provided in Figure 2c. As in the alternating-authority MAMPC, to always achieve asymptotic stability, one can successively shrink $D_{WP}$ until $D_{WP} = \mathcal{R}_{LQR}$ in the same way as presented in equation (5.3).

### 5.2. Remark on Robustness

In some cases, MAMPC may be robustified to account for model uncertainties. To this end, we propose a robustification procedure to improve the robustness of MAMPC using the idea of bounding error margin. We demonstrate the procedure by applying it to the standard MAMPC.

Provided with the true system dynamics as $f$, we distinguish $\tilde{f}$ as the model of the dynamical system. Define the model uncertainties of a controller $u$ with initial condition $x[i]$ at time step $i+k$ as

$$\Delta x_u[k|i] := y[k] - \bar{y}[k],$$

where

$$y[0] = x[i], \quad y[k] = f(y[k-1], u(y[k-1])), \quad \bar{y}[0] = x[i], \quad \bar{y}[k] = \tilde{f}(\bar{y}[k-1], u(\bar{y}[k-1])).$$

When $\|\Delta x_u[k|i]\| = \alpha > 0$, it is possible that $y[k] \notin \mathcal{R}_{LQR}$ but $\bar{y}[k] \in \mathcal{R}_{LQR}$, potentially leading to a failure of MAMPC. To address this problem, we define a set operator $\text{Ero}_d$ as follows

$$\text{Ero}_d(\mathcal{X}) := \{x \in \mathcal{X} \mid \|x - q\| \geq \delta, \forall q \in \partial \mathcal{X}\}$$

for some $\delta > 0$, where $\mathcal{X}$ is an arbitrary set and $\partial \mathcal{X}$ denotes the boundary of the set $\mathcal{X}$. We claim that for $\|\Delta x_u[k|i]\| = \alpha > 0$, if $\bar{y}[k] \in \text{Ero}_d(\mathcal{R}_{LQR})$ with $\delta \geq \alpha$, then there must be $y[k] \in \mathcal{R}_{LQR}$. Proof of the claim is a direct application of the triangle inequality of norm. We can therefore incorporate the set operator $\text{Ero}_d$ to enhance the robustness of MAMPC with respect to model uncertainties.
We can apply the above procedure to robustify the standard MAMPC as follows

\[ u^\delta_{\text{MAMPC}}(x) := \begin{cases} 
  u_{\text{LQR}}(x), & x \in R_{\text{LQR}}, \\
  u_{\text{NN}}(x), & \exists i = 1, \ldots, N_{\text{LQR}}, \ y[i] \in \mathcal{R}^\delta_{\text{LQR}}, \\
  u_{\text{MP}}(x), & \text{otherwise},
\end{cases} \]

where \( R^\delta_{\text{LQR}} := \text{Err}_{\delta}(R_{\text{LQR}}) \) and \( A^\delta := \text{Err}_{\delta}(X) \times U \), for some \( \delta > 0 \). The stability of the above robustified MAMPC is stated in the following theorem.

**Theorem 5.4 (Robustification of Standard MAMPC).** Suppose the model uncertainties are upper bounded by some \( \alpha > 0 \), that is,

\[ \| \Delta x_{i}[k|i] \| \leq \alpha, \quad \forall i \in \mathbb{Z}_{\geq 0}, \forall k = 0, \ldots, N_{\text{LQR}}. \]

Then the following closed-loop system

\[ x[i+1] = f(x[i], u^\alpha_{\text{MAMPC}}(x[i])), \quad i \in \mathbb{Z}_{\geq 0}, \]

with \( x[0] = x_0 \in X_0 \) is locally asymptotically stable in \( X_0 \).

**Proof.** The proof is identical to that of Theorem 5.1. Note that it possible that \( R^\delta_{\text{LQR}} = \emptyset \). In this case, the hybrid control is equivalent to a dual-mode MPC-LQR controller.

Similar to the above example, the robustification procedure can also be applied to enhance the alternating-authority MAMPC and way-point MAMPC.

**5.3. Remark on Necessity of Fail-Safe MPC.**

The hybrid control scheme can clearly be more efficient if we can skip the forward verification and remove the fail-safe MPC mode. In light of this observation, we describe a condition, where the fail-safe MPC mode will never be invoked and thus can be safely removed from the hybrid control scheme.

Define the following “fail-free” MAMPC

\[ u^+_{\text{MAMPC}}(x) := \begin{cases} 
  u_{\text{LQR}}(x), & x \in R_{\text{LQR}}, \\
  u_{\text{NN}}(x), & x \in X_0 \setminus R_{\text{LQR}}.
\end{cases} \]

Define \( L : X \to \mathbb{R}_+ \) as an augmented cumulative cost function of NN, that is,

\[ L(x) := \sum_{k=0}^{N-1} c(y[k], u_{\text{NN}}(y[k])) + c_f(y[N]), \]

with

\[ y[0] = x, \quad y[k] = f(y[k-1], u_{\text{NN}}(y[k-1])). \]

From Theorem 13.1 in [4], we are ready to state the condition in the following theorem.
Theorem 5.5 (Fail-Free MAMPC). If there exists some nonnegative function $\gamma : X_0 \rightarrow \mathbb{R}_+$ such that

\begin{align}
(x, u_{\text{NN}}(x)) &\in A, \quad \forall x \in X_0, \\
L(x) &\leq J^*(x) + \gamma(x), \quad \forall x \in X_0, \\
\gamma(x) - c(x, 0) &< 0, \quad \forall x \in X_0 \setminus \mathbf{0},
\end{align}

then the closed-loop system

$$x[i + 1] = f(x[i], u^i_{\text{MAMPC}}(x[i])), \quad i \in \mathbb{Z}_{\geq 0},$$

with $x[0] = x_0 \in X_0$ is locally asymptotically stable in $X_0$ without violating any constraints in the original MPC problem (3.2).

Proof. The proof of the theorem is a direct application of Theorem 13.1 in [4]. □

Note that inequality conditions (5.5) are typically verified through sampling and interpolation [4], which makes this approach only suitable for relatively simple, moderately sized problems. Besides, the above condition is only a sufficient condition, for which we can safely discard the fail-safe MPC mode. Other methods are possible. For example, [26, 6] have proposed methods to verify the stability of a NN controller using Lyapunov approaches.

Last but not least, we emphasize that the above analysis does not impose any condition on the approximation error of the NN. In fact, as long as the NN has the right input and output dimension, the corresponding MAMPC will be closed-loop stable, even if the weights of the NN are random. In the worst case, MAMPC reduces to a hybrid controller consisting of just the MPC and the LQR, where the NN mode will never be invoked. This reduced hybrid controller is identical to one proposed in [20].

Next, we evaluate the computational performance of the proposed method in Section 6.

6. Numerical Experiments. To evaluate the running time performance of MAMPC, we conduct numerical experiments on four models in robotics, i.e., pendulum, triple pendulum, bicopter, and quadcopter. The models are selected for their significance in robotic applications: pendulum models are the building blocks for robot arm and leg control; copter models are an important model class in unmanned aerial vehicle control. We highlight key parameters and results of the four experiments in this section. For implementation details, please refer to the source code [29].

6.1. Pendulum. The first model is a single arm pendulum as shown in Figure 3. The goal is to maintain the pendulum at the position of the highest potential energy as marked by the dashed line.

The state is defined as $x := [\theta \quad \dot{\theta}]^\top \in [-\pi, \pi) \times \mathbb{R}$, where $\theta$ is the angular displacement from the inverted position. The input is a scalar $u \in \mathbb{R}$, which is the applied torque at the joint. The equation of motion is described by the following nonlinear dynamics

\begin{align}
\ddot{\theta} = \frac{3g}{2l} \sin \theta + \frac{3}{ml^2} u,
\end{align}

where definitions and values of parameters are specified in Table 1. By convention, we assign $\theta$ to be the clockwise angular displacement between the rod and the dashed line.
The control objective is to steer the pendulum to the origin by applying a limited torque \( u \in [-0.05, 0.05] \) in \( \text{N} \cdot \text{m} \) at the joint. The MPC design highlights a sampling interval of 0.1 s, a planning horizon \( N = 5 \), and three optimization variables per step, resulting in a linearly-constrained quadratic programming (LCQP) of 0.5 s prediction horizon and 15 variables. Plant nonlinearity is handled through linearizing around the equilibrium, leading to a linear time-invariant system. We apply a standard MAMPC to control the pendulum with

\[
\mathcal{R}_{\text{LQR}} = \{ x \in \mathbb{R}^2 \mid \| x \|_2 \leq 0.5 \}, \quad N_{\text{LQR}} = 5.
\]

The NN is a two-layer, 20-neuron multilayer perceptron (MLP) trained through supervised learning with data randomly sampled from \( \theta \in [\pi, \pi], \dot{\theta} \in [-1, 1] \).

To evaluate the computational efficiency of MAMPC, we compare its per-step running time and total running time with implicit and explicit MPC. The system is simulated with the same initial condition at \([\pi/2, 1/2]\) with the three different controllers. The per-step running times of the three controllers are shown in Figure 4 with the corresponding statistics tabulated in the first column of Table 5. The total running times of the three controllers are summarized in the first column of Table 6. In addition, An up time analysis of the three control modes within the MAMPC is tabulated in the first column of Table 7.

### 6.2. Triple Pendulum

The second model is a triple pendulum as shown in Figure 5, which extends the above pendulum model to a more complex use case. The goal is to maintain the triple pendulum at the position of the highest potential energy.

The state is defined as \( x := [\theta_1 \quad \dot{\theta}_1 \quad \theta_2 \quad \dot{\theta}_2 \quad \theta_3 \quad \dot{\theta}_3]^T \in ([-\pi, \pi] \times \mathbb{R})^3 \), where \( \theta_1, \theta_2, \theta_3 \) are the angular displacements of the three links respectively. The input is defined as \( u := [u_1 \quad u_2 \quad u_3] \in \mathbb{R}^3 \), which is the applied torques at the three joints respectively. The equation of motion of the triple pendulum is obtained through...
symbolic computation using the principle of Lagrangian mechanics:

\[
\frac{d}{dt} \frac{\partial L}{\partial \dot{\theta}_i} - \frac{\partial L}{\partial \theta_i} = u_i, \quad \forall i = 1, 2, 3,
\]

where \( L := T - V \) is the Lagrangian of the system, \( T \) is the kinematic energy, and \( V \) is the potential energy.

The kinematic energy and potential energy of the triple pendulum are

\[
T = 0.5m_1 (\dot{x}_1^2 + \dot{y}_1^2) + 0.5m_2 (\dot{x}_2^2 + \dot{y}_2^2) + 0.5m_3 (\dot{x}_3^2 + \dot{y}_3^2),
\]
\[
V = m_1gy_1 + m_2gy_2 + m_3gy_3,
\]

where the coordinates of the three joints are calculated as follows

\[
\begin{align*}
x_1 &= l_1 \cos (\pi/2 - \theta_1), & y_1 &= l_1 \sin (\pi/2 - \theta_1), \\
x_2 &= x_1 + l_2 \cos (\pi/2 - \theta_1 + \theta_2), & y_2 &= y_1 + l_2 \sin (\pi/2 - \theta_1 + \theta_2), \\
x_3 &= x_2 + l_3 \cos (\pi/2 - \theta_1 + \theta_2 + \theta_3), & y_3 &= y_2 + l_3 \sin (\pi/2 - \theta_1 + \theta_2 + \theta_3),
\end{align*}
\]

where definitions and values of parameters are specified in Table 2. For simplicity, we assign \( \theta_1, \theta_2, \theta_3 \) to be the counterclockwise angular displacements between the respective links and the perpendicular directions of the corresponding dashed lines. Explicit form of the system dynamics takes pages to write and is therefore omitted here.

| Parameter | Description          | Value  |
|-----------|---------------------|--------|
| \( m_1 \) | Mass of link 1      | 0.1 kg |
| \( l_1 \) | Length of link 1    | 0.1 m  |
| \( m_2 \) | Mass of link 2      | 0.1 kg |
| \( l_2 \) | Length of link 2    | 0.1 m  |
| \( m_3 \) | Mass of link 3      | 0.1 kg |
| \( l_3 \) | Length of link 3    | 0.1 m  |
| \( g \)  | Gravitational acceleration | 9.8 m \cdot s^{-2} |

*Table 2: Specifications of the triple pendulum model.*
Fig. 5. Diagram of a triple pendulum model. Note that the centers of mass of the first and second torques coincide with the second and third joints. A torque between \([-1, 1]\) N·m is being applied to each joint to keep the triple pendulum at its inverted position.

The control objective is to steer the triple pendulum to the origin by applying three limited torque inputs \(u_1, u_2, u_3 \in [-1, 1]\) in N·m at the three joints, respectively. The MPC design highlights a sampling interval of 0.1 s, a planning horizon \(N = 5\), and nine optimization variables per step, resulting in a LCQP of 0.5 s prediction horizon and 45 variables. Linearization is applied as before to handle nonlinearity. Because triple pendulum is a chaotic system, we apply an alternating-authority MAMPC instead of standard MAMPC with

\[
\mathcal{R}_{LQR} = \{x \in \mathbb{R}^6 \mid \|x\|_2 \leq 0.4\}, \quad N_{LQR} = 5, \quad i_d = 2.
\]

The NN is a three-layer, 50-neuron MLP trained through supervised learning with data randomly sampled from \(\theta_1, \theta_2, \theta_3 \in [-\pi/6, \pi/6], \quad \dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3 \in [-1, 1]\).

Evaluation of computational efficiency is also identical to the single pendulum experiment. The system is simulated three times with alternating-authority MAMPC, implicit MPC, and explicit MPC at \([\pi/6, 1 \pi/6, 1 \pi/6, 1]\). The per-step running times of the three controllers are shown in Figure 6 with the corresponding statistics tabulated in the second column of Table 5. The total running times of the three controllers are summarized in the second column of Table 6. An up time analysis of the three control modes within the MAMPC is tabulated in the second column of Table 7.

6.3. Bicopter. As a parallel robotic application to the pendulum models, the third model is a bicopter as shown in Figure 7. The goal is to hover the bicopter in air.

The state is defined as \(x := [x \dot{x} y \dot{y} \theta \dot{\theta}]^\top \in \mathbb{R}^4 \times (-\pi, \pi) \times \mathbb{R}\), where \(x, y\) are horizontal and vertical translations and \(\theta\) is the angle of tilting. The input is defined as \(u := [u_1 \ u_2]^\top \in \mathbb{R}^2\), where \(u_1, u_2\) are the thrust exerted by the left and right propellers, respectively. The equation of motion is described in the following:

\[
\begin{bmatrix}
\dot{x} \\
\dot{y} \\
\dot{\theta}
\end{bmatrix} = \begin{bmatrix}
\frac{1}{m}(u_1 + u_2) \sin \theta \\
\frac{1}{m}(u_1 + u_2) \cos \theta - g \\
\frac{1}{I}(u_1 - u_2)
\end{bmatrix},
\]
Fig. 6. Triple pendulum per-step running time comparison. The triple pendulum evolves from \( x = [\pi/6 \ 1 \ \pi/6 \ 1 \ \pi/6 \ 1] \) to the origin. Simulation stops when \( \|x\|_2 \leq 0.01 \). The dashed line represents the running time of NN mode without forward simulation check.

where definitions and values of parameters are specified in Table 3.

![Diagram of a bicopter model.](image)

Fig. 7. Diagram of a bicopter model. Each of the two propellers is capable of generating a thrust between 0.1 N and 9.1572 N to keep the copter hover in air.

| Parameter   | Description                        | Value          |
|-------------|------------------------------------|----------------|
| \( m \)     | Mass of the bicopter               | 1.1 kg         |
| \( l \)     | Arm length of the bicopter         | 0.21 m         |
| \( I \)     | Moment of inertia of the bicopter  | 0.0196 kg \cdot m^2 |
| \( g \)     | Gravitational acceleration         | 9.8 m \cdot s^{-2} |

**Table 3**

Specifications of the bicopter model.

The control objective is to steer the bicopter to the origin by applying limited thrusts \( u_1, u_2 \in [0.1, 9.1572] \) in N at the two propellers. The thrust limits are chosen such that hovering thrusts are approximately in the center of the thrust limit. The MPC design highlights a sampling interval of 0.1 s, a planning horizon \( N = 20 \), and eight optimization variables per step, resulting in a LCQP of 2.0 s prediction horizon and 160 variables. To handle plant nonlinearity, we linearize the system around the equilibrium. We apply a standard MAMPC to control the bicopter with

\[
\mathcal{R}_{\text{LQR}} = \{ x \in \mathbb{R}^6 \mid \|x\|_2 \leq 0.5 \}, \quad N_{\text{LQR}} = 10.
\]

The NN is a three-layer, 50-neuron MLP trained through supervised learning with data randomly sampled from

\[
x, y, \theta \in [-\pi/2, \pi/2], \quad \dot{x}, \dot{y}, \dot{\theta} \in [-1, 1].
\]
Evaluation of computational efficiency is identical as before. The system is simulated with the same initial condition at \([\pi/2 \ 1 \ \pi/2 \ 1 \ \pi/2 \ 1]\) with standard MAMPC, implicit MPC, and explicit MPC. The per-step running times of the three controllers are shown in Figure 8 with the corresponding statistics tabulated in the third column of Table 5. The total running times of the three controllers are summarized in the third column of Table 6. An up time analysis of the three control modes within the MAMPC is tabulated in the third column of Table 7.

![Fig. 8. Bicopter per-step running time comparison. The triple pendulum evolves from \(x = [\pi/2 \ 1 \ \pi/2 \ 1 \ \pi/2 \ 1]\) to the origin. Simulation stops when \(\|x\|_2 \leq 0.01\). The dashed line represents the running time of NN mode without forward simulation check.](image)

### 6.4. Quadcopter

As an extension to the ideal bicopter model, the last model is a quadcopter as shown in Figure 9 [1]. The goal is also to hover the copter in air but with more realistic system dynamics.

The state is defined as \(\mathbf{x} := [\dot{x} \ y \ \dot{y} \ z \ \phi \ \dot{\phi} \ \theta \ \dot{\theta} \ \psi \ \psi] \in \mathbb{R}^6 \times ((-\pi, \pi) \times \mathbb{R})^3\), where \(x, y, z\) represent translations and \(\phi, \theta, \psi\) represent rotations corresponding to the three ZYX Euler angles roll, pitch, and yaw. The input is \(\mathbf{u} = [u_1 \ u_2 \ u_3 \ u_4]^T \in \mathbb{R}^4\), where \(u_i\) is the rotational speeds of the \(i^{th}\) propeller for \(i = 1, 2, 3, 4\). The equation of motion of the quadcopter is developed by [1], as described below:

\[
\begin{bmatrix}
\dot{x} \\
\dot{y} \\
\dot{z} \\
\dot{\phi} \\
\dot{\theta} \\
\dot{\psi}
\end{bmatrix}
= \begin{bmatrix}
(cos \phi \sin \theta \cos \psi + \sin \phi \\cos \psi) /m \\
(cos \phi \sin \theta \sin \psi - \sin \phi \\cos \psi) /m \\
-g + \cos \phi \cos \theta U_1 /m \\
\frac{\dot{\phi}}{I_{\phi} - \frac{1}{I_{\phi}}} U_2 - \frac{\dot{\theta}}{I_{\phi} - \frac{1}{I_{\phi}}} U_3 + \frac{\dot{\psi}}{I_{\phi} - \frac{1}{I_{\phi}}} U_4 \\
\frac{\dot{\theta}}{I_{\phi} - \frac{1}{I_{\phi}}} U_2 - \frac{\dot{\phi}}{I_{\phi} - \frac{1}{I_{\phi}}} U_3 + \frac{\dot{\psi}}{I_{\phi} - \frac{1}{I_{\phi}}} U_4 \\
\frac{\dot{\psi}}{I_{\phi} - \frac{1}{I_{\phi}}} U_2 - \frac{\dot{\phi}}{I_{\phi} - \frac{1}{I_{\phi}}} U_3 + \frac{\dot{\psi}}{I_{\phi} - \frac{1}{I_{\phi}}} U_4
\end{bmatrix},
\]

\[
\begin{bmatrix}
U_1 \\
U_2 \\
U_3 \\
U_4 \\
U_5
\end{bmatrix}
= \begin{bmatrix}
b(u_1^2 + u_2^2 + u_3^2 + u_4^2) \\
b(u_2^2 - u_3^2) \\
b(-u_1^2 + u_2^2 + u_3^2 + u_4^2) \\
d(-u_1^2 + u_2^2 - u_3^2 + u_4^2) \\
u_1 - u_2 - u_3 + u_4
\end{bmatrix},
\]

where definitions and values of parameters are specified in Table 4.

The control objective is to steer the quadcopter to the origin by applying bounded rotations \(u_1, u_2, u_3, u_4 \in [0, 313.96]\) in rad · s\(^{-1}\) at the four propellers. The rotor limits are chosen such that hovering rotor rotations are approximately in the center of the rotor limit. The MPC design highlights a sampling interval of 0.1 s, a planning horizon \(N = 20\), and 16 optimization variables per step, resulting in a LCQP of 2.0 s prediction horizon and 320 variables. Nonlinearity is handled by linearization around the equilibrium. Because the closed-loop quadcopter dynamics with MPC takes many
steps to converge, we choose a way-point MAMPC with

$$\mathcal{D}_{WP} = \{ \mathbf{x} \in \mathbb{R}^{12} \mid \| \mathbf{x} \|_2 \leq 2 \}, \quad N_{WP} = 10,$$

$$\mathcal{R}_{LQR} = \{ \mathbf{x} \in \mathbb{R}^{12} \mid \| \mathbf{x} \|_2 \leq 0.5 \}, \quad N_{LQR} = 10.$$  

The NN is a four-layer, 60-neuron MLP trained through supervised learning with data randomly sampled from

$$x, y, z \in [-0.5, 0.5], \quad \dot{x}, \dot{y}, \dot{z} \in [-0.1, 0.1],$$

$$\phi, \theta \in [-\pi/6, \pi/6], \quad \psi \in [-\pi/4, \pi/4], \quad \dot{\phi}, \dot{\theta}, \dot{\psi} \in [-0.1, 0.1].$$

Evaluation of computational efficiency is identical as before. The system is simulated with the same initial condition at $[0.5 \ 0.1 \ 0.5 \ 0.1 \ 0.5 \ 0.1 \ \pi/6 \ 0.1 \ \pi/6 \ 0.1 \ \pi/4 \ 0.1]$ with way-point MAMPC, implicit MPC, and explicit MPC. The per-step running times of the three controllers are shown in Figure 10 with the corresponding statistics tabulated in the last column of Table 5. The total running times of the three controllers are summarized in the last column of Table 6. An up time analysis of the three control modes within the MAMPC is tabulated in the last column of Table 7.

Note that this experiment is already at the limit of explicit MPC method. The explicit MPC offline caching takes over 10 hours to complete with the resulting explicit control law occupies around 100 MB of hard disk storage. In comparison, NN supervised learning takes less than one hour and only requires less than 20 KB of storage. As explicit MPC method scales exponentially with state dimension, it will not likely work well for problems of higher dimensions.

Table 4
Specifications of the quadcopter model.

| Parameter | Description | Value |
|-----------|-------------|-------|
| $m$       | Mass of the quadcopter | 1.1 kg |
| $l$       | Arm length of the quadcopter | 0.21 m |
| $I_{xx}$  | Moment of inertia about $x$ axis | 0.0196 kg m$^2$ |
| $I_{yy}$  | Moment of inertia about $y$ axis | 0.0196 kg m$^2$ |
| $I_{zz}$  | Moment of inertia about $z$ axis | 0.0264 kg m$^2$ |
| $I_p$     | Moment of inertia of a propeller | $8.5 \times 10^{-4}$ kg m$^2$ |
| $b$       | Buoyancy coefficient | $9.29 \times 10^{-5}$ N s$^2$ |
| $d$       | Drag coefficient | $1.1 \times 10^{-6}$ N m s$^2$ |
| $g$       | Gravitational acceleration | 9.8 m s$^{-2}$ |
Fig. 10. Quadcopter per-step running time comparison. The triple pendulum evolves from $x = [0.5 \ 0.1 \ 0.5 \ 0.1 \ \pi/6 \ 0.1 \ \pi/6 \ 0.1 \ \pi/4 \ 0.1]$ to the origin. Simulation stops when $\|x\|_2 \leq 0.01$. The dashed line represents the running time of NN mode without forward simulation check.

### Table 5

|               | Pendulum (µs) | Triple Pend. (µs) | Bicopter (µs) | Quadcopter (µs) |
|---------------|--------------|------------------|---------------|-----------------|
| Imp. MPC      | 438 ± 252    | 555 ± 366        | 436 ± 221     | 474 ± 184       |
| Exp. MPC      | 257 ± 5.38   | 263 ± 3.07       | 245 ± 3.25    | 246 ± 2.01      |
| MAMPC-MPC     | 1110 ± 397   | 597 ± 406        | 1300 ± 615    | 1070 ± 404      |
| MAMPC-NN      | 82.3 ± 66.8  | 158 ± 95.4       | 252 ± 136     | 247 ± 126       |
| MAMPC-LQR     | 1.60 ± 0.259 | 2.31 ± 0.480     | 1.79 ± 0.198  | 1.68 ± 0.102    |
| NN            | 32.2 ± 0.445 | 35.8 ± 0.500     | 41.7 ± 11.7   | 37.3 ± 0.771    |

Summary of per-step running time in µs. MAMPC-MPC, MAMPC-NN, MAMPC-LQR refer to the MPC mode, NN mode, and LQR mode of the MAMPC respectively. NN is the NN mode of MAMPC without forward simulation. The numerical results indicate that MAMPC has better amortized running time if the system is mostly running in the NN and LQR mode. However, the trade-off is that MAMPC has worse worst-case latency.

### 6.5. Timing Method

The numerical experiments are conducted on a macOS with a 3.2 GHz 6-Core Intel Core i7 processor. All codes are implemented and timed using MATLAB. The running time data in Figure 4 to Figure 10 are measured by repeating and timing the same experiment 50 times and then taking the median of the 50 measurements. Due to initial loading and logging performed by MATLAB MPC package, the first control step of each experiment is very slow and thus omitted from data. The per-step running times are obtained by taking the means and one standard deviations of relevant data along a trajectory. The total running times are obtained by summing the running time of every data point along a trajectory. Up time division is obtained by dividing the running time and the number of control steps each mode of MAMPC takes by the total amount.

We remark that the explicit MPC solver in MATLAB is likely not very optimized. There should be room for improvement at least for the simpler systems like pendulum and bicopter. Nevertheless, we decide to benchmark in MATLAB as it is one of the standard software that the community use for robotic control. We believe that although the exact values of running time in the Section may vary with a different setup of hardware and software, the comparative performance of each control methods should approximately remain the same.

With the numerical experiments presented above, we continue to provide a few key interpretations of the experimental results and controller design in Section 7.
COMPOSING MPC WITH LQR AND NEURAL NETWORKS

|                     | Pendulum (ms) | Triple Pend. (ms) | Bicoter (ms) | Quadcopter (ms) |
|---------------------|---------------|-------------------|--------------|-----------------|
| Imp. MPC            | 12.7          | 5.55              | 17.0         | 52.6            |
| Exp. MPC            | 7.46          | 2.63              | 9.57         | 27.3            |
| MAMPC               | 7.18          | 3.78              | 11.4         | 42.1            |

Table 6
Summary of total running time in ms. The implicit MPC method is slower than the explicit MPC and MAMPC method. MAMPC method is mostly on par with the explicit MPC, except in the case of quadcopter.

|                     | Pendulum (%) | Triple Pend. (%) | Bicoter (%) | Quadcopter (%) |
|---------------------|--------------|------------------|-------------|----------------|
| MPC Time Div.       | 91.831       | 78.12            | 79.762      | 89.270         |
| MPC Step Div.       | 24.000       | 33.333           | 17.949      | 37.634         |
| NN Time Div.        | 6.879        | 20.883           | 19.877      | 10.570         |
| NN Step Div.        | 24.000       | 33.333           | 23.077      | 19.355         |
| LQR Time Div.       | 0.290        | 0.305            | 0.361       | 0.160          |
| LQR Step Div.       | 52.000       | 33.333           | 58.974      | 43.011         |

Table 7
Summary of division of up time in percentage. Even though most of the compute steps the controller is running in the NN or LQR mode, it spends most of it computational time in the MPC mode.

7. Discussions. The numerical experiments above indicates that MAMPC may have superior amortized running time but inferior worst-case running time in common robotic applications. As data shows, the LQR mode and NN mode of MAMPC is faster than implicit and explicit MPC methods. Moreover, we will be able to further reduce the running time of the NN mode by approximately an order of magnitude, if we can safely skip the forward verification and apply the NN directly. However, the MPC mode of MAMPC is slower than the implicit and explicit methods due to overhead from checking switch conditions, leading to increased latency in the worst-case scenario. Despite of the drawback, experiments show that the MPC mode is seldom invoked, which enables MAMPC to have smaller average latency than the implicit and explicit methods.

Beyond the direct implications of the numerical experiments, we make following remarks on set design, forward verification, additional running time optimizations, and limitations.

7.1. Set Design. Identification and design of feasibility set $X_0$ and attraction set $R_{LQR}$ is very challenging. One one hand, it is easier to just approximate them by conservative set estimates for robustness measures. For example, one can replace $X_0$ and $R_{LQR}$ with $X_0^\prime$ and $R_{LQR}^\prime$, where $X_0^\prime \subset X_0$ and $R_{LQR}^\prime \subset R_{LQR}$. On the other hand, however, to guarantee computational efficiency, one should make sets $R_{LQR}, R_{NN}$ as large as possible. This way, faster modes of MAMPC get invoked more frequently than the slow, default mode of MAMPC, leading to a more efficient amortized running time performance. Conservative set design makes the system more robust to model uncertainties but also slower in running time. Balancing the trade-off between latency and robustness is therefore a design process.
7.2. **Forward verification.** The horizon used in forward verification immediately affects the maximum possible running time of the MAMPC. A valid verification horizon should not make the running time of the MPC mode longer than the maximum allowable computational latency. Moreover, an additional rule of thumb is to choose a verification horizon such that the maximum possible running time of NN mode is on par with the running time of the implicit MPC.

The simulation method used for forward verification also directly impacts the worst-case latency of the MAMPC. For continuous plants, we recommend to use forward Euler method with a sampling time that is as large as possible and to use a conservative set design to absorb integration errors. Like designing sets, choosing the right sampling time is a balancing act between latency and robustness.

7.3. **Additional Optimizations.** For small to moderately sized problems, we may consider developing a fail-free MAMPC as specified in Theorem 5.5, since it will further speed up computation. Fail-free MAMPC prevents controller from ever having to default to fail-safe mode and therefore skips the routine computation of forward verification in NN mode. It significantly improves the amortized and worst-case running time.

If the computational device is capable of executing independent threads in parallel, it may be advantageous to distribute computation across multiple threads to further reduce latency. For example, a possible parallelization strategy for standard MAMPC is as follows: 1) compute the three control modes of MAMPC in three parallel threads, 2) among those that pass their corresponding switch conditions, execute whichever that finishes first, and 3) reset threads for next control step and repeat. This way, the worst-case running time will be roughly identical to that of the original MPC. As a result, the parallelized MAMPC scheme is at worst equivalent to the hybrid strategy in [20].

7.4. **Limitations.** The foremost limitation of the MAMPC method is that without removing the fail-safe controller it can never reduce the worst-case running time to be below that of the original MPC. This shortcoming limits the applicability of the hybrid control scheme. For instance, if the original MPC breaks closed-loop stability due to exceedingly high latency, the MAMPC counterpart will also fail to make the closed-loop system stable. In the worst-case scenario, the MAMPC controller is at best equivalent to the original MPC controller. Therefore, MAMPC is only suited for applications where the existing MPC controller works but users demand additional improvement in amortized running time. Examples of such applications include routine repetitive manipulator tasks, where one can acquire proficiency through practice, such as pick-and-place and item assembly.

Another limitation of MAMPC is that the MPC designs presented in the four numerical experiments are neither optimal nor certified. For example, instead of linearization around the equilibrium, nonlinearity may be handled directly by a nonlinear solver. Also, no verification has been done to check that the MPC controllers obtained through plant linearization are persistent feasible and closed-loop stable with respect to the original nonlinear plants. Hence, the controllers in the numerical experiments are not solutions for deployment. Instead, they are academic prototypes designed specifically to investigate the computational aspects of the proposed method.

As the final remark, we conclude our work in Section 8.

8. **Conclusion.** To improve average computational efficiency of MPC, we have developed a triple-mode hybrid control named MAMPC. Stability of MAMPC is
guaranteed via forward simulation, while efficiency is achieved by replacing MPC with a more efficient NN or LQR, whenever stability permits. Results indicate that MAMPC often has a better amortized running time but a slightly prolonged worst-case running time.

As future work, one may generalize the hybrid control pattern in MAMPC to other controllers beyond MPC. The hybrid control pattern consists of three basic ingredients: an efficient equilibrium policy, a frugal heuristic policy, and an expensive default policy. When the state is close to its equilibrium, we may effectively approximate the system dynamics with a linear system and control it with a linear controller. Developing such linear controller can be achieved through LQR or other linear control methods in either frequency or time domain. Away from the equilibrium, first-order approximation will no longer be effective, so a new controller is required to maintain system stability. In this case, one can attempt to leverage machine learning methods, such as supervised learning or reinforcement learning, to train a heuristic controller that is empirically effective and efficient. Lastly, when the heuristic policy is expected to fail, we should invoke a default policy to maintain the system in control, or at least to prevent the system from catastrophic failures. In this work, the default policy is MPC but this can be other nonlinear controllers too.

Acknowledgments. This material is based upon work supported by the National Science Foundation under Grant Numbers CNS-1837244. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. This material is based upon work supported by the U.S. Department of Energy’s Office of Energy Efficiency and Renewable Energy (EERE) award number CID DE-EE0008872. The views expressed herein do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

REFERENCES

[1] K. Alexis, G. Nikolakopoulos, and A. Tzes, Model predictive quadrotor control: attitude, altitude and position experimental studies, IET Control Theory and Applications, 6 (2012), pp. 1812–1827.
[2] B. D. O. Anderson and J. B. Moore, Optimal Control: Linear Quadratic Methods, Dover Publications, 2018, pp. 53–54.
[3] G. Bledt, M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing, and S. Kim, MIT Cheetah 3: Design and Control of a Robust, Dynamic Quadruped Robot, in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2018, pp. 2245–2252.
[4] F. Borrelli, A. Bemporad, and M. Morari, Predictive Control for Linear and Hybrid Systems, Cambridge University Press, 2017.
[5] F. Borrelli, P. Falcone, T. Keviczky, J. Asgari, and D. Hrovat, MPC-based approach to active steering for autonomous vehicle systems, International Journal of Vehicle Autonomous Systems, 3 (2005), pp. 265–291.
[6] S. Chen, M. Fazlyab, M. Morari, G. J. Pappas, and V. M. Preciado, Learning Lyapunov functions for hybrid systems, in Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control, 2021, pp. 1–11.
[7] S. W. Chen, T. Wang, N. Atanasov, V. Kumar, and M. Morari, Large Scale Model Predictive Control with Neural Networks and Primal Active Sets, Automatica, 135 (2022), p. 109947.
[8] M. Diehl, H. G. Bock, and J. P. Schlöder, A Real-Time Iteration Scheme for Nonlinear Optimization in Optimal Feedback Control, SIAM Journal on Control and Optimization, 43 (2005), pp. 1714–1736.
[9] T. Faulwasser, T. Weber, P. Zometa, and R. Findeisen, Implementation of Nonlinear Model Predictive Path-Following Control for an Industrial Robot, IEEE Transactions on
Control Systems Technology, 25 (2016), pp. 1505–1511.
[10] B. Houska, H. J. Ferreau, and M. Diehl, An Auto-Generated Real-Time Iteration Algorithm for Nonlinear MPC in the Microsecond Range, Automatica, 47 (2011), pp. 2279–2285.
[11] B. Karg, T. Alamo, and S. Lucia, Probabilistic performance validation of deep learning-based robust mpc controllers, International Journal of Robust and Nonlinear Control, (2021).
[12] B. Karg and S. Lucia, Stability and feasibility of neural network-based controllers via output range analysis, in IEEE Conference on Decision and Control, IEEE, 2020, pp. 4947–4954.
[13] H. K. Khalil, Nonlinear Systems, Prentice Hall, 3rd ed., 2002.
[14] D. P. Kingma and J. Ba, Adam: A Method for Stochastic Optimization, in International Conference on Learning Representations, 2015.
[15] S. Kleff, A. Meduri, R. Budhiraja, N. Mansard, and L. Righetti, High-Frequency Nonlinear Model Predictive Control of a Manipulator, in IEEE International Conference on Robotics and Automation, 2021.
[16] S. Kundersha, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koo, L. Marion, and R. Tedrake, Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot, Autonomous robots, 40 (2016), pp. 429–455.
[17] M. Kvassina, J. Hledík, I. Rauova, and M. Fikar, Complexity reduction of explicit model predictive control via separation, Automatica, 49 (2013), pp. 1776–1781.
[18] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpentier, L. Righetti, S. Vijayakumar, and N. Mansard, Crocoddyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control, in IEEE International Conference on Robotics and Automation, IEEE, 2020, pp. 2536–2542.
[19] J. Mattingley and S. Boyd, CVXGEN: a code generator for embedded convex optimization, Optimization and Engineering, 13 (2012), pp. 1–27.
[20] H. Michalska and D. Q. Mayne, Robust Receding Horizon Control of Constrained Nonlinear Systems, IEEE transactions on automatic control, 38 (1993), pp. 1623–1633.
[21] J. Nubert, J. Köhler, V. Berenz, F. Allgöwer, and S. Trimpe, Safe and Fast Tracking on a Robot Manipulator: Robust MPC and Neural Network Control, IEEE Robotics and Automation Letters, 5 (2020), pp. 3050–3057.
[22] T. Parisini, M. Sanguineti, and R. Zoppoli, Nonlinear Stabilization by Reeding-Horizon Neural Regulators, International Journal of Control, 70 (1998), pp. 341–362.
[23] T. Parisini and R. Zoppoli, A Reeding-horizon Regulator for Nonlinear Systems and a Neural Approximation, Automatica, 31 (1995), pp. 1443–1451.
[24] J. A. Paulson and A. Mesbah, Approximate Closed-Loop Robust Model Predictive Control with Guaranteed Stability and Constraint Satisfaction, IEEE Control Systems Letters, 4 (2020), pp. 719–724.
[25] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl, Model Predictive Control: Theory, Computation and Design, Nob Hill Publishing, 2017.
[26] S. M. Richards, F. Berkenkamp, and A. Krause, The Lyapunov Neural Network: Adaptive Stability Certification for Safe Learning of Dynamical Systems, in Conference on Robot Learning, PMLR, 2018, pp. 466–476.
[27] S. Summers, C. N. Jones, J. Lygeros, and M. Morari, A Multiresolution Approximation Method for Fast Explicit Model Predictive Control, IEEE Transactions on Automatic Control, 56 (2011), pp. 2530–2541.
[28] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, MIT press, 2018.
[29] F. Wu, Memory-Augmented Model Predictive Control: Numerical Experiments, https://github.com/fywu85/mampc, 2021.
[30] X. Zhang, M. Bujarbaruah, and F. Borrelli, Safe and Near-Optimal Policy Learning for Model Predictive Control using Primal-Dual Neural Networks, in 2019 American Control Conference, IEEE, 2019, pp. 354–359.
[31] R. Zoppoli, M. Sanguineti, G. Gnecco, and T. Parisini, Neural Approximations for Optimal Control and Decision, Springer, 2020.