Composing MPC With LQR and Neural Network for Amortized Efficiency and Stable Control

Fangyu Wu, Guanhua Wang, Siyuan Zhuang, Kehan Wang, Alexander Keimer, Ion Stoica, and Alexandre Bayen, *Fellow, IEEE*

***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, remains a computational challenge. To address this, common solutions include explicit MPC and function approximation. Both methods, whenever applicable, may improve the computational efficiency of the implicit MPC by several orders of magnitude. Nevertheless, explicit MPC often requires expensive pre-computation and does not easily apply to higher-dimensional problems. Meanwhile, function approximation, although scales better with dimension, still requires pre-training on a large dataset and generally cannot guarantee to find an accurate surrogate policy, the failure of which often leads to closed-loop instability. To address these issues, we propose a triple-mode hybrid control scheme, named Memory-Augmented MPC, by combining a linear quadratic regulator, a neural network, and an 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. The proposed scheme does not require pre-computation and is capable of improving the *amortized* running time of the composed MPC with a well-trained neural network. In addition, the scheme maintains closed-loop stability with *any* neural networks of proper input and output dimensions, alleviating the need for certifying optimality of the neural network in safety-critical applications.**

***Note to Practitioners*—This article was motivated by the need to reduce the amortized cost of MPC in repetitive industrial robotic applications, where long-term operational cost is important and safety is critical. Examples of such applications include factory robotic arm manipulation and fixed-route quadcopter payload transport. Unlike explicit MPC or function approximation, our approach does not require any pre-computation or pre-training. Rather, it attains task proficiency over time by learning a surrogate neural network on the spot and by gradually replacing the costly MPC with the more efficient surrogate model so long as safety permits. Consequently, the proposed scheme incurs a learning cost during the initial phase of the deployment but usually becomes more adept on the task afterwards, leading to amortized efficiency.**

***Index Terms*— Model predictive control, neural networks, robotics.**

I. Introduction

MODEL predictive control (MPC) is a powerful method for controlling dynamical systems with constraints [1], [2]. It has been widely used in robotics, with applications ranging from ground [3] and aerial [4] vehicle maneuvers to humanoid [5] and quadruped [6] robot control.

When it comes 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 computes every solution of the optimization problem via parametric optimization *offline* and looks up the computed solutions *online*.

Explicit MPC often has faster running time than implicit MPC but requires expensive pre-computation. For problems of many variables and of intricate constraints, such pre-computation tends to demand a prohibitively large amount of time and memory.

Alternatively, one can approximate an implicit MPC controller with a neural network (NN) through supervised learning. Neural network models are more flexible when it comes to pre-training and generally scales better to higher-dimensional problems. Nevertheless, to our best knowledge, 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 bounded approximation errors; *2)* it is challenging to establish closed-loop stability without any knowledge of the accuracy of the approximation.

Motivated by the above shortcomings of explicit MPC and NN function approximation methods and inspired by [7], 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.

The simplicity of our method makes it possible for it to be analyzed theoretically for stability. Such stability guarantee is not a common quality among works that involve

Manuscript received 13 January 2023; accepted 9 March 2023. Date of publication 24 March 2023; date of current version 9 April 2024. This article was recommended for publication by Associate Editor M. Gaggero and Editor M. Dotoli upon evaluation of the reviewers' comments. 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. *(Corresponding author: Fangyu Wu.)*

The authors are with the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA 94709 USA (e-mail: fangyuwu@berkeley.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TASE.2023.3259428.

Digital Object Identifier 10.1109/TASE.2023.3259428

neural networks but is a necessity for safety-critical applications. Moreover, unlike conventional explicit MPC or function approximation methods, our method does not require any form of pre-computing. Instead, it is operational on day one, in spite of being inefficient, and learns to be efficient over time.

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 amortized computational efficiency of Memory-Augmented MPC via four numerical experiments.

II. Related Works

A major limitation of MPC has been a lack of computational efficiency as evident in applications such as the Atlas humanoid robot [5] and the MIT Cheetah robot [6]. To overcome this shortcoming, one typically has three approaches: *1)* developing an efficient implicit MPC policy, *2)* pre-compiling an explicit MPC policy, and *3)* approximating a slow implicit MPC policy by an efficient surrogate policy such as a NN.

Implicit MPC approach implements the MPC policy by devising an efficient numerical algorithm for solving the optimization program. This is perhaps the most common approach of implementing an MPC policy. Efficient implicit MPC controller often depends on custom optimization solvers that leverage structures specific to the problems they solve, such as [8] and [9]. Implementation of such implicit MPC is usually done in a fast low-level programming language such as C. For example, see [10], [11]. Moreover, recent advances in software and hardware enable implicit MPC controllers to be deployed in traditionally challenging problems, including [12], [13], and [14].

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 pre-computing the solution of the optimization problem and looking up the solution in real time. At its core, explicit methods achieve reduced running time at the cost of expensive pre-computation and increased memory footprint. To reduce memory footprint, numerous works have been developed [15], [16]. Nevertheless, due to poor scalability of the pre-computation step, explicit MPC has more limited use cases than implicit MPC.

When the explicit MPC method fails to apply to a problem, one may consider another alternative, namely, function approximation. Function approximation approach implements an MPC policy by approximating the costly MPC control law with an efficient surrogate model, such as a NN, and uses that surrogate model for fast online deployment. Early works in this direction include [17], [18], which has described how 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, [19] proposes a NN method with *probabilistic* optimality bounds; [20] has designed a projection operator that can modify a NN controller to a stabilizing one for linear systems; and [21] proposes a mixed-integer linear program to certify stability and feasibility of ReLU-activated NN controllers. Besides, rather than replacing the implicit MPC controller, [22] uses a NN to warm start an optimization solver, which may then speed up convergence of the implicit MPC solver. For a comprehensive study, readers may consult [23]. Applications of function approximation methods have found many successes in practice, such as [24].

It is worth noting that closed-loop stability of the function approximation methods generally requires reasonable convergence of NN model to the original implicit MPC control law. Nevertheless, training a NN is commonly done through a black-box solver, such as [25], the convergence of which is generally not guaranteed. In contrary, we propose a novel function approximation approach named Memory-Augmented MPC, which is always stable and always guarantees constraint satisfaction, without imposing *any* condition on the convergence and optimality of the NN. The accuracy of the NN only affects running time: the more accurate the NN is, the more efficient the overall control scheme is. As a result, it can be deployed without any training, with the expectation that it will attain computational efficiency over time via learning.

As a final remark, a quick comparison of MAMPC with a few key existing works is in order. We acknowledge that our method is inspired by the seminal early work in dual-mode MPC [7], which combines an MPC with a LQR. An advantage of our approach over [7] is that it could potentially provide more reduction in latency with the help of a fast intermediate NN mode. As detailed in Section VI, with parallelization, MAMPC is *at worst* equivalent to [7]. Moreover, compared to the techniques in [20] and [22], which only apply to linear plants with convex quadratic objectives, our method works for any nonlinear problems.

III. Composing MPC, LQR, and NN

We begin the section by first introducing some basic notations, terminologies, and a key stability theorem of MPC. Next, we present the standard construction in MAMPC, followed by two variations of such construction, namely, alternating-authority MAMPC and way-point MAMPC.

A. Basics of MPC

Consider the discrete-time time-invariant constrained dynamical systems of the form

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

where

$$\mathbf{x}[0] = \mathbf{x}_0 \quad \text{for some } \mathbf{x}_0 \in \mathbb{R}^n,$$
$$(\mathbf{x}[i], \mathbf{u}[i]) \in \mathbb{A}, \quad \forall i > 0,$$

where $\mathbb{A} \subset \mathbb{R}^n \times \mathbb{R}^m$ is a closed set defining the system constraints, $\mathbf{x}[i]$ is the state at time index i, $\mathbf{u}[i]$ 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 $(\mathbf{0}_n, \mathbf{0}_m)$,

i.e., $f(\mathbf{0}_n, \mathbf{0}_m) = \mathbf{0}_n$, and the linearized time-invariant system of $f(\cdot,\cdot)$ around the equilibrium is stabilizable.

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

Consider a finite planning and control horizon N. Let $\mathcal{X}^{1:N} := \mathbf{x}[1], \ldots, \mathbf{x}[N])$ be a trajectory of the system and $\mathcal{U}^{0:N-1} := \mathbf{u}[0], \ldots, \mathbf{u}[N-1])$ be the corresponding ordered control sequence.

Provided with the dynamical system (1), an MPC is an optimal control law implicitly defined through the following set of optimization problems

$$\begin{aligned} \min_{\mathcal{X}_i^{1:N}, \mathcal{U}_i^{0:N-1}} \quad & \sum_{k=0}^{N-1} c(\mathbf{x}[k|i], \mathbf{u}[k|i]) + c_f(\mathbf{x}[N|i]) \\ \text{s.t.:} \quad & \mathbf{x}[0|i] = \mathbf{x}[i], \\ & \mathbf{x}[k+1|i] = f(\mathbf{x}[k|i], \mathbf{u}[k|i]), \\ & (\mathbf{x}[k|i], \mathbf{u}[k|i]) \in \mathbb{A}, \\ & \mathbf{x}[N|i] \in \mathbb{X}_f, \\ & k = 0, \ldots, N-1, \end{aligned} \tag{2}$$

where N is planning and control horizon, $c : \mathbb{A} \to \mathbb{R}_+$ is a continuous stage cost function, $c_f : \mathbb{X}_f \to \mathbb{R}_+$ is a continuous terminal cost function, $\mathbf{x}[k|i]$ is the predicted state k steps ahead of *present* state $\mathbf{x}[i]$, $\mathbf{u}[k|i]$ is the anticipated input that generates $\mathbf{x}[k+1|i]$ from $\mathbf{x}[k|i]$, $\mathcal{X}_i^{1:N} := \mathbf{x}[1|i], \ldots, \mathbf{x}[N|i])$ and $\mathcal{U}_i^{0:N-1} := \mathbf{u}[0|i], \ldots, \mathbf{u}[N-1|i])$ are two sets of optimization variables corresponding to predicted states and anticipated inputs, $\mathbb{X}_f$ is a terminal constraint set containing the origin usually designed to guarantee asymptotic stability. If optimization problem (2) is feasible and admits an optimal solution $(\mathcal{X}_i^{1:N*}, \mathcal{U}_i^{0:N-1*})$, then the MPC control law selects the first element of $\mathcal{U}_i^{0:N-1*}$ as the input at time index i, that is,

$$u_{\text{MPC}}(\mathbf{x}[i]) := u^*[0|i] = \mathcal{U}_i^{0:N-1*}[0]. \tag{3}$$

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

$$\mathbf{x}[i+1] = f(\mathbf{x}[i], u_{\text{MPC}}(\mathbf{x}[i])), \quad i \in \mathbb{Z}_{\geq 0}, \tag{4}$$

with initial condition $\mathbf{x}[0] = \mathbf{x}_0 \in \mathbb{X}_0 := \{\mathbf{x}_0 \in \mathbb{R}^n \mid$ problem (2) is feasible$\}$ is the set of all admissible initial states. For well-posedness of MPC, we require $\mathbb{X} \subseteq \mathbb{X}_0$.

Let $J(\mathbf{x}[i]) := \sum_{k=0}^{N-1} c(\mathbf{x}[k|i], \mathbf{u}[k|i]) + c_f(\mathbf{x}[N|i])$ be the cumulative cost and $J^*(\mathbf{x}[i]) := \sum_{k=0}^{N-1} c(\mathbf{x}^*[k|i], \mathbf{u}^*[k|i]) + c_f(\mathbf{x}^*[N|i])$ 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 [2].

Theorem 1 (Local Asymptotic Stability of MPC): Assume that

- *The stage cost and terminal cost are continuous and positive definite.*
- $\mathbb{X}_f$ *is control invariant.*
- *For all* $\mathbf{x} \in \mathbb{X}_f$,

$$\min_{\mathbf{u}:\mathbf{u}\in\mathbb{U}, f(\mathbf{x},\mathbf{u})\in\mathbb{X}_f} \quad c(\mathbf{x},\mathbf{u}) - c_f(\mathbf{x}) + c_f(f(\mathbf{x},\mathbf{u})) \leq 0.$$

then the MPC problem (2) is persistently feasible and the closed-loop system (4) is locally asymptotically stable with respect to the origin. In addition, $J^*(\cdot)$ is a Lyapunov function and $\mathbb{X}_0$ 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.

B. Composition Rules

Provided with a computationally costly implicit MPC u_{MPC} that satisfies Theorem (1), we propose a hybrid control scheme, namely, *Memory-Augmented Model Predictive Control* (MAMPC), $u_{\text{MAMPC}} : \mathbb{X} \to \mathbb{U}$, by augmenting u_{MPC} with a LQR controller u_{LQR} and a NN controller u_{NN}, where the LQR controller and the NN controller are independently developed from the implicit MPC. We first describe how the LQR and NN controllers are derived from the implicit MPC and then present ways to combine these controllers.

1) LQR From MPC: Provided with an 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 ∞, 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

$$\begin{aligned} \min_{\mathcal{X}_i^{1:\infty}, \mathcal{U}_i^{0:\infty}} \quad & \sum_{k=0}^{\infty} \mathbf{x}^\top[k|i]\boldsymbol{Q}\mathbf{x}[k|i] + \mathbf{u}^\top[k|i]\boldsymbol{R}\mathbf{u}[k|i] \\ \text{s.t.:} \quad & \mathbf{x}[0|i] = \mathbf{x}[i], \\ & \mathbf{x}[k+1|i] = \boldsymbol{A}\mathbf{x}[k|i] + \boldsymbol{B}\mathbf{u}[k|i], \\ & k \in \mathbb{Z}_+, \end{aligned} \tag{5}$$

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

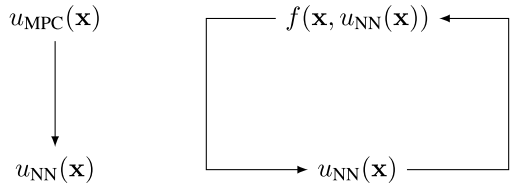
It can be shown that the optimal LQR control law for an initial state $\mathbf{x}$ is a linear map

$$u_{\text{LQR}}(\mathbf{x}) = -\boldsymbol{K}\mathbf{x} \tag{6}$$

for some $\boldsymbol{K} \in \mathbb{R}^{m\times n}$ [26]. Furthermore, if the linearized system $(\boldsymbol{A}, \boldsymbol{B})$ is *stabilizable* and if $\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K}$ is Hurwitz, then the closed-loop system

$$\mathbf{x}[i+1] = f(\mathbf{x}[i], u_{\text{LQR}}(\mathbf{x}[i])), \tag{7}$$

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 [27]. Specifically, $\mathcal{R}_{\text{LQR}}$ is a set that if $\mathbf{x}_0 \in \mathcal{R}_{\text{LQR}}$ and $\mathbf{x}[0] = \mathbf{x}_0$, then $\mathbf{x}[i] \in \mathcal{R}_{\text{LQR}}, \forall i \in \mathbb{Z}_{>0}$ and $\lim_{i\to\infty} \mathbf{x}[i] = \mathbf{0}$.



(a) Imitation phase (b) Adaptation phase

Fig. 1. Schematics of the learning processes. (a) Imitation phase: $(\mathbf{x}, u_{\text{MPC}}(\mathbf{x}))$ provides data to train the NN controller u_{NN} in a supervised learning setting. (b) Adaptation phase (optional): u_{NN} interacts with the environment $f(\cdot,\cdot)$ to enhance control effectiveness in a reinforcement learning setting.

As implied by local asymptotic stability, the largest possible $\mathcal{R}_{\text{LQR}}$ is nonempty in theory. In practice, however, estimating $\mathcal{R}_{\text{LQR}}$ is a challenging problem. To this end, several methods have been proposed, such as [28], [29], and [30].

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, $\mathbf{x}[i] \in \mathcal{R}_{\text{LQR}}$.

2) NN From MPC: Provided with an MPC, we derive a NN controller *1)* by imitating the MPC policy $u_{\text{MPC}}(\mathbf{x})$ 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

$$\min_{\mathcal{W}} \quad \frac{1}{M}\sum_{\mathbf{x}\in\mathcal{D}} \|u_{\text{MPC}}(\mathbf{x}) - \phi(\mathbf{x} \mid \mathcal{W})\|_2^2, \tag{8}$$

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

Therefore, the resulting NN control law is

$$u_{\text{NN}}(\mathbf{x}) := (\mathbf{x} \mid \overline{\mathcal{W}}), \tag{9}$$

where $\overline{\mathcal{W}}$ is a suboptimal solution to problem (8), generally estimated via a gradient method such as Adam [25].

In the *adaptation* phase, it is possible to fine tune the NN further through a reinforcement learning (RL) algorithm. A reasonable choice for environment may be the dynamic model used by the MPC. A good choice of the reward function may be the cost function of the MPC. The training setup varies based on the specific choice of the RL algorithm. For a comprehensive review of reinforcement learning methods, please refer to [31]. We note that the adaptation phase may be skipped if the imitation phase is sufficiently effective or if the adaptation phase does not bring any improvement.

In practice, people find that implicit MPC controllers can be well approximated by a more efficient NN controller [17]. 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.

$$\mathbf{x}[i+1] = f(\mathbf{x}[i], u_{\text{NN}}(\mathbf{x}[i])). \tag{10}$$

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, even if the NN is *random*. Next, we present the basic form of our method, i.e., the standard MAMPC.

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 (10) for up to N_{LQR} steps: if there *exists* a step $j \le N_{\text{LQR}}$ 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

$$u_{\text{MAMPC}}(\mathbf{x}) := \begin{cases} u_{\text{LQR}}(\mathbf{x}), & \text{if } \mathbf{x} \in \mathcal{R}_{\text{LQR}}, \\ u_{\text{NN}}(\mathbf{x}), & \begin{array}{l}\text{if } \mathbf{x} \in \mathbb{X}_0 \setminus \mathcal{R}_{\text{LQR}} \quad \text{and} \\ \quad \exists i = 1, \ldots, N_{\text{LQR}}, \quad \mathbf{y}[i] \in \mathcal{R}_{\text{LQR}} \quad \text{and} \\ \quad \forall j = 0, \ldots, i, \quad (\mathbf{y}[j], u_{\text{NN}}(\mathbf{y}[j])) \in \mathbb{A}, \end{array} \\ u_{\text{MPC}}(\mathbf{x}), & \text{otherwise}, \end{cases}$$

where $\mathbf{y}$ is the *simulated* state by numerically stepping: $\mathbf{y}[i] = f(\mathbf{y}[i-1], u_{\text{NN}}(\mathbf{y}[i-1]))$ with $\mathbf{y}[0] = \mathbf{x}$, $\mathcal{R}_{\text{LQR}}$ is designed to be a subset of $\mathbb{X}_0$, and $N_{\text{LQR}} \in \mathbb{Z}_{>0}$ is the verification horizon of $\mathcal{R}_{\text{LQR}}$. Note that compliance with state and input constraints are satisfied by construction: we explicit check constraint satisfaction via forward simulation.

As we show in Section IV, the stability of NN relies on verifying whether the system reaches within $\mathcal{R}_{\text{LQR}}$ 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) 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.

$$u^{\mathrm{AA}}_{\mathrm{MAMPC}}(\mathbf{x}) := \begin{cases} u_{\mathrm{LQR}}(\mathbf{x}), & \text{if } \mathbf{x} \in \mathcal{R}_{\mathrm{LQR}}, \\ \hline u_{\mathrm{NN}}(\mathbf{x}), & \text{if } \mathbf{x} \in \mathbb{X}_0 \setminus \mathcal{R}_{\mathrm{LQR}} \quad \text{and} \\ & u_{\mathrm{NN}}(\mathbf{x}) \in \mathbb{U} \wedge \mathbf{y}[1] \in \mathbb{X} \quad \text{and} \\ & i \pmod{i_d} \neq 0, \\ \hline u_{\mathrm{MPC}}(\mathbf{x}), & \text{otherwise}, \end{cases}$$

where $i_d \in \mathbb{Z}_{\geq 1}$ is the period of MPC defaulting and $\mathcal{R}_{\mathrm{LQR}} \subset \mathbb{X}_0$.

The main variation from the standard MAMPC is that we do not verify if the system falls within $\mathcal{R}_{\mathrm{LQR}}$ in N_{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 attaction compared to standard MAMPC, since we do not verify whether the system enters $\mathcal{R}_{\mathrm{LQR}}$ through forward simulation.

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_{LQR}: a small N_{LQR} may be too short to predict convergence of the NN controller; a large N_{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, $\mathcal{D}_{\mathrm{WP}}$, 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.

$$u^{\mathrm{WP}}_{\mathrm{MAMPC}}(\mathbf{x}) := \begin{cases} u_{\mathrm{LQR}}(\mathbf{x}), & \text{if } \mathbf{x} \in \mathcal{R}_{\mathrm{LQR}}, \\ \hline u_{\mathrm{NN}}(\mathbf{x}), & \text{if } \mathbf{x} \in \mathcal{D}_{\mathrm{WP}} \setminus \mathcal{R}_{\mathrm{LQR}} \text{ and} \\ & \exists i = 1, \dots N_{\mathrm{LQR}}, \ \mathbf{y}[i] \in \mathcal{R}_{\mathrm{LQR}} \text{ and} \\ & \forall j = 0, \dots i, \quad (\mathbf{y}[j], u_{\mathrm{NN}}(\mathbf{y}[j])) \in \mathbb{A}_{\mathrm{WP}}, \\ \hline u_{\mathrm{NN}}(\mathbf{x}), & \text{if } \mathbf{x} \in \mathbb{X}_0 \setminus \mathcal{D}_{\mathrm{WP}} \text{ and} \\ & \exists i \in 1, \dots, N_{\mathrm{WP}}, \ \mathbf{y}[i] \in \mathcal{D}_{\mathrm{WP}} \text{ and} \\ & \forall j \in 0, \dots, i, \quad (\mathbf{y}[j], u_{\mathrm{NN}}(\mathbf{y}[j])) \in \mathbb{A}, \\ \hline u_{\mathrm{MPC}}(\mathbf{x}), & \text{otherwise}, \end{cases}$$

where $\mathcal{D}_{\mathrm{WP}}$ is a compact set that contains $\mathcal{R}_{\mathrm{LQR}}$, $\mathbb{A}_{\mathrm{WP}} := D_{\mathrm{WP}} \times \mathbb{U}$, and $N_{\mathrm{WP}} \in \mathbb{Z}_{>0}$ is the verification horizon of $\mathcal{D}_{\mathrm{WP}}$.

At minimum, we require that the way-point set contains the region of attraction of the LQR, that is, $\mathcal{R}_{\mathrm{LQR}} \subset \mathcal{D}_{\mathrm{WP}}$. Additional improvement of the way-point set design can be performed through search. For example, one can parameterize $\mathcal{D}_{\mathrm{WP}}$ 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.

IV. Theoretical Analysis

In this section, we prove the stability of the three MAMPC methods proposed above. We also remark on the robustness of the methods and the necessity of the fail-safe MPC mode.

A. 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 $\mathbb{X}_0$, as stated in the theorem below.

Theorem 2 (Stability of Standard MAMPC): For every MPC problem of the form (2), there always exists a u_{MAMPC} of the form (11). Furthermore, the closed-loop system

$$\mathbf{x}[i+1] = f(\mathbf{x}[i], u_{\mathrm{MAMPC}}(\mathbf{x}[i])), \quad i \in \mathbb{Z}_{\geq 0}, \tag{11}$$

with $\mathbf{x}[0] = \mathbf{x}_0 \in \mathbb{X}_0$ is locally asymptotically stable in $\mathbb{X}_0$.

Proof: See Appendix A. □

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 $\mathcal{R}_{\mathrm{MPC}}$ is first steered by u_{MPC} into $\mathcal{R}_{\mathrm{NN}}$, which is then steered by u_{NN} into $\mathcal{R}_{\mathrm{LQR}}$, 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 $\mathbb{X}_0$, because it does not check whether NN shoots inside $\mathcal{R}_{\mathrm{LQR}}$. Rather, we can only show that the system is locally asymptotically stable in $\mathcal{R}_{\mathrm{LQR}}$ and will never escape $\mathbb{X}_0$, as stated the theorem below.

Theorem 3 (Stability of Alternating-Authority MAMPC): For every MPC problem of the form (2), there always exists a $u^{\mathrm{AA}}_{\mathrm{MAMPC}}$ of the form (11). Furthermore, the closed-loop system

$$\mathbf{x}[i+1] = f\big(\mathbf{x}[i], u^{\mathrm{AA}}_{\mathrm{MAMPC}}(\mathbf{x}[i])\big), \quad i \in \mathbb{Z}_{\geq 0}, \tag{12}$$

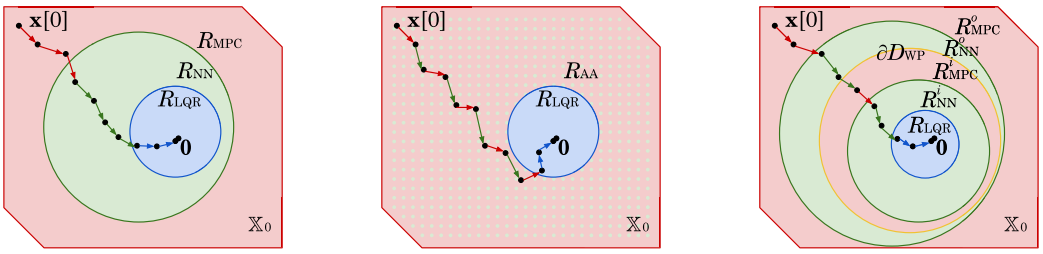
with $\mathbf{x}[0] = \mathbf{x}_0 \in \mathbb{X}_0$ is locally asymptotically stable in $\mathcal{R}_{\mathrm{LQR}}$. Besides, for every initial condition $\mathbf{x}[0] \in \mathbb{X}_0$, $\mathbf{x}[i] \in \mathbb{X}_0$ for all $i \in \mathbb{Z}_+$.

Proof: See Appendix B. □

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 in the *entire* $\mathbb{X}_0$ is absolutely required, one can replace the admissible control set $\mathbb{X}_0$ with a time-varying set $\mathbb{S}[i]$, where

$$\mathcal{R}_{\mathrm{LQR}} \subset \mathbb{S}[i+1] \subset \mathbb{S}[i] \subset \mathbb{X}_0, \quad \forall i = 1, \dots, n-1, \tag{13}$$



(a) Standard MAMPC (b) Alternating-Authority MAMPC (c) Way-Point MAMPC

Fig. 2. An illustrative example of how different variants of MAMPC *could* steer a system toward equilibrium. (a) Standard MAMPC: a triple-mode hybrid control composed of an MPC, a NN, and a LQR. (b) Alternating-authority MAMPC: a variant of the standard MAMPC, modified for chaotic systems. (c) Way-point MAMPC: a variant of the standard MAMPC, modified for slow systems. Solid red regions represent $\mathbb{X}_0$; solid green regions represent $\mathcal{R}_{\text{NN}}$; *green dots* indicate periodic controller switching region in alternating-authority MAMPC; and solid blue regions represent $\mathcal{R}_{\text{LQR}}$. A black dot represents a state and an arrow represents a controlled state transition. The color of arrows indicates which sub-controller is invoked during that control step: red is MPC; green is NN; and blue is LQR. The yellow line represents $\partial\mathcal{D}_{\text{WP}}$. Note that the above examples by no means exhaustive. For instance, it is possible that $\mathcal{R}_{\text{NN}} = \emptyset$.

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

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

Theorem 4 (Stability of Way-Point MAMPC): For every MPC problem of the form (2), there always exists a $u^{\text{WP}}_{\text{MAMPC}}$ *of the form (11). Furthermore, the closed-loop system*

$$\mathbf{x}[i+1] = f\big(\mathbf{x}[i], u^{\text{WP}}_{\text{MAMPC}}(\mathbf{x}[i])\big), \quad i \in \mathbb{Z}_{\geq 0}, \tag{14}$$

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

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

where $V : \mathbb{R}^n \to \mathbb{R}_+$ *is a Lyapunov function of the closed-loop system (4) and* $\partial\mathcal{D}_{\text{WP}}$ *is the boundary of the compact set* $\mathcal{D}_{\text{WP}}$ *that by construction contains* $\mathcal{R}_{\text{LQR}}$.

Proof: See Appendix C. □

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 $\mathcal{D}_{\text{WP}}$ until $\mathcal{D}_{\text{WP}} = \mathcal{R}_{\text{LQR}}$ in the same way as presented in equation (13).

B. Remark on Robustness

MAMPC may be *robustified* to account for model uncertainties using the idea of bounding error margin. We demonstrate such robustification procedure by applying it to the standard MAMPC.

Let the *true* system dynamics be f. We distinguish $\bar{f}$ as the *model* of the dynamical system. Define the model uncertainties of a controller u with initial condition $\mathbf{x}[i]$ at time step $i+k$ as

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

where

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

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

$$\text{Ero}_\delta(\mathcal{X}) := \mathbf{x} \in \mathcal{X} \mid \|\mathbf{x} - \mathbf{q}\| \geq \delta, \forall \mathbf{q} \in \partial\mathcal{X}\} \text{ for some } \delta > 0,$$

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

We can apply the above procedure to robustify the standard MAMPC as follows

$$u^\delta_{\text{MAMPC}}(\mathbf{x}) := \begin{cases} u_{\text{LQR}}(\mathbf{x}), & \mathbf{x} \in \mathcal{R}_{\text{LQR}}, \\ u_{\text{NN}}(\mathbf{x}), & \mathbf{x} \in \mathbb{X}_0 \setminus \mathcal{R}_{\text{LQR}}, \\ & \exists i = 1, \ldots, N_{\text{LQR}}, \quad \bar{\mathbf{y}}[i] \in \mathcal{R}^\delta_{\text{LQR}}, \\ & \forall j = 0, \ldots, j, \quad (\bar{\mathbf{y}}[j], u_{\text{NN}}(\bar{\mathbf{y}}[j])) \in \mathbb{A}^\delta \\ u_{\text{MPC}}(\mathbf{x}), & \text{otherwise}, \end{cases}$$

where $\mathcal{R}^\delta_{\text{LQR}} :=_\delta (\mathcal{R}_{\text{LQR}})$ and $\mathbb{A}^\delta :=_\delta (\mathbb{X}) \times \mathbb{U}$, for some $\delta > 0$. Here, δ can be viewed geometrically as a margin of robustness.

The stability of the above robustified MAMPC is stated in the following theorem.

Theorem 5 (Robustification of Standard MAMPC): Suppose the model uncertainties are upper bounded by some $\alpha > 0$, *that is,*

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

Then the following closed-loop system

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

with $\mathbf{x}[0] = \mathbf{x}_0 \in \mathbb{X}_0$ *is locally asymptotically stable in* $\mathbb{X}_0$.

Proof: The proof is identical to that of Theorem 2. Note that it possible that $\mathcal{R}^{\delta}_{\mathrm{LQR}} = \emptyset$. In this case, the hybrid control is virtually 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.

C. Remark on Necessity of Fail-Safe MPC

The hybrid control scheme can be clearly 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

$$u^{+}_{\mathrm{MAMPC}}(\mathbf{x}) := \begin{cases} u_{\mathrm{LQR}}(\mathbf{x}), & \mathbf{x} \in \mathcal{R}_{\mathrm{LQR}}, \\ u_{\mathrm{NN}}(\mathbf{x}) & \mathbf{x} \in \mathbb{X}_0 \setminus \mathcal{R}_{\mathrm{LQR}}. \end{cases}$$

Let $L : \mathbb{X} \to \mathbb{R}_+$ be an augmented cumulative cost function of NN, that is,

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

with

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

From Theorem 13.1 in [2], we are ready to state the condition in the following theorem.

Theorem 6 (Fail-Free MAMPC): If there exists some non-negative function $\gamma : \mathbb{X}_0 \to \mathbb{R}_+$ such that

$$\begin{aligned} &(\mathbf{x}, u_{\mathrm{NN}}(\mathbf{x})) \in \mathbb{A}, \quad \forall \mathbf{x} \in \mathbb{X}_0, \\ &L(\mathbf{x}) \le J^*(\mathbf{x}) + \gamma(\mathbf{x}), \quad \forall \mathbf{x} \in \mathbb{X}_0, \\ &\gamma(\mathbf{x}) - c(\mathbf{x}, \mathbf{0}) < 0, \quad \forall \mathbf{x} \in \mathbb{X}_0 \setminus \mathbf{0}, \end{aligned} \tag{15}$$

then the closed-loop system

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

with $\mathbf{x}[0] = \mathbf{x}_0 \in \mathbb{X}_0$ is locally asymptotically stable in $\mathbb{X}_0$ without violating any constraints in the original MPC problem (2).

Proof: The proof of the theorem is a direct application of Theorem 13.1 in [2]. □

Note that the inequality conditions in (15) are typically verified through sampling and interpolation [2], which makes this approach only suitable for relatively simple, moderately sized problems. Besides, the above theorem is only a sufficient condition and other sufficient conditions are possible too. For example, see [32], [33].

Last but not least, we emphasize that *all* the analysis above does not impose any condition on the approximation error of the NN. So long as the NN has the right input and output dimensions, 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 virtually identical to the one proposed in [7], despite of having slightly different running time performance.

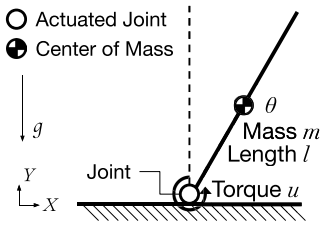


Fig. 3. Diagram of a pendulum model. A torque between $[-0.05, 0.05]$ N·m is being applied to the hinge to keep the pendulum at its inverted position.

V. Numerical Experiments

To evaluate the running time performance of MAMPC, we conduct four numerical experiments on controlling a pendulum, a triple pendulum, a bicopter, and a quadcopter. The models are selected for their relevance in industrial robotic applications: Pendulum models are the building blocks for robot arm manipulation, while copter models are important model classes in unmanned aerial vehicle control. Performance and efficiency of MAMPC are evaluated through comparison with the corresponding baseline implicit MPC. In addition to the four experiments, a short numerical simulation is appended to the end of the section to demonstrate the notion of robustness margin.

All computations were conducted on an Intel Core i7-1065G7 CPU machine. Implementation is done via MATLAB. The following paragraphs highlight the key parameters and results of the four experiments. For implementation details, please refer to the source code at [34].

A. 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 $\mathbf{x} := \begin{bmatrix} \theta & \dot{\theta} \end{bmatrix}^\top \in [-\pi, \pi) \times \mathbb{R}$, where θ is the angular displacement from the inverted position. The input is a scalar $u \in \mathbb{R}$, which is a bounded torque applied at the joint. The control objective is to steer the pendulum to the origin through the limited torque actuator $u \in [-0.05, 0.05]$ N · 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

$$\mathcal{R}_{\mathrm{LQR}} = \{\mathbf{x} \in \mathbb{R}^2 \mid \|\mathbf{x}\|_2 \le 0.5\}, \quad N_{\mathrm{LQR}} = 5.$$

The NN is a two-layer, 20-neuron multilayer perceptron (MLP) trained through supervised learning with data uniformly randomly sampled from $\theta \in [\pi, \pi], \dot{\theta} \in [-1, 1]$. Details of the NN architecture can be found in Table III in the Appendix.

Performance and efficiency of MAMPC, at various stages of training, is illustrated in Figure 4. The total and per-step

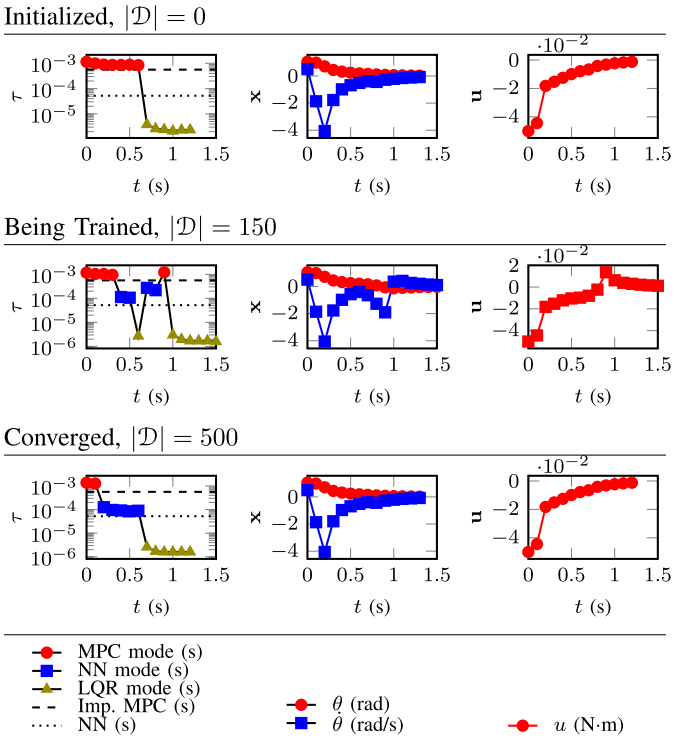


Fig. 4. Application of standard MAMPC to the pendulum problem. τ is running time. $\mathbf{x}$ is selected states and $\mathbf{u}$ is selected inputs. t is simulation time. At initialization, the NN mode fails forward verification and thus never get invoked. As the NN is being trained, its likelihood of passing forward verification increases. At last, a well trained NN takes over most of MPC steps, resulting in a more efficient operation.

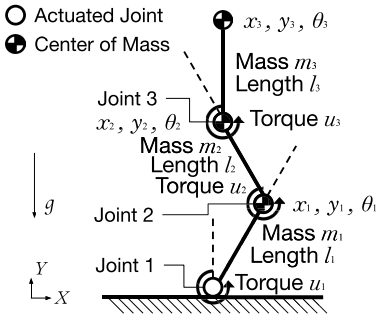


Fig. 5. Diagram of a triple pendulum model. Note that the centers of mass of the first and second torques coincide with the second and third joints. A torque between $[-1, 1]$ N · m is being applied to each joint to keep the triple pendulum at its inverted position.

running times of MAMPC are summarized in the first columns of Table I and Table II, respectively.

Please note that if even though the running times of implicit MPC and NN appear constant in Figure 4, they are in fact time-varying if we zoom in. This applies to the other three systems below.

B. 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 $\mathbf{x} := \theta_1 \quad \dot{\theta}_1 \quad \theta_2 \quad \dot{\theta}_2 \quad \theta_3 \quad \dot{\theta}_3]^\top \in ([-\pi, \pi) \times \mathbb{R})^3$, where $\theta_1, \theta_2, \theta_3$ are the angular displacements

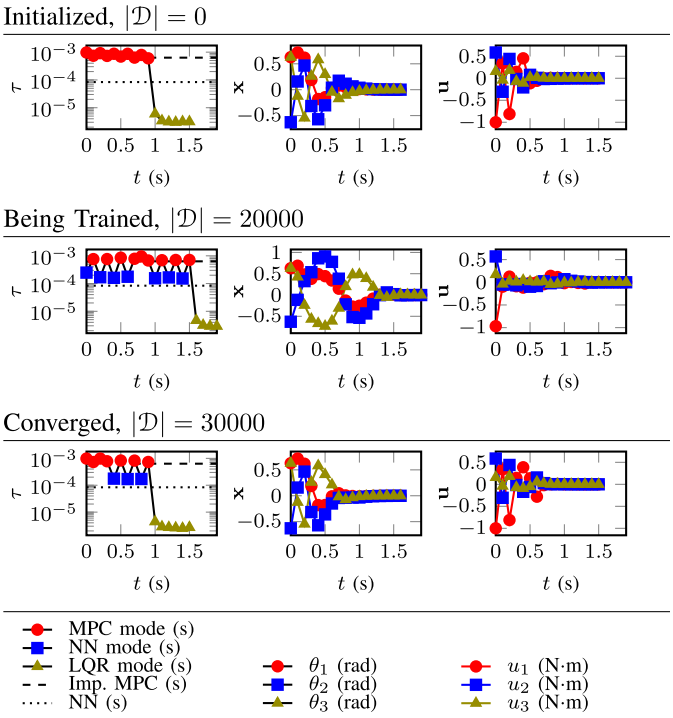


Fig. 6. Application of alternating-authority MAMPC to the triple pendulum problem. τ is running time. $\mathbf{x}$ is selected states and $\mathbf{u}$ is selected inputs. t is simulation time. At initialization, the NN mode fails forward verification and thus never get invoked. As the NN is being trained, its likelihood of passing forward verification increases. At last, a well trained NN takes over most of MPC steps, resulting in a more efficient operation.

of the three links respectively. The input is defined as $\mathbf{u} := u_1 \quad u_2 \quad u_3] \in \mathbb{R}^3$, which is the applied torques at the three joints respectively. The control objective is to steer the triple pendulum to the origin by applying three limited torque inputs $u_1, u_2, u_3 \in [-1, 1]$ 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

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

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

$$\theta_1, \theta_2, \theta_3 \in [-\pi/6, \pi/6], \quad \dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3 \in [-1, 1].$$

Details of the NN architecture can be found in Table III in the Appendix.

Performance and efficiency of MAMPC, at various stages of training, is illustrated in Figure 6. The total and per-step running times of MAMPC are summarized in the second columns of Table I and Table II, respectively.

Lastly, note that running times of implicit MPC is represented as constant lines when in reality it is decreasing with time. For readability, we choose to use the median of the

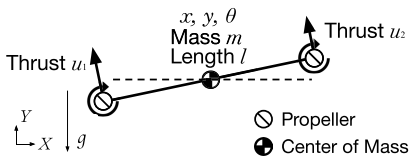


Fig. 7. Diagram of a bicopter model. Each of the two propellers is capable of generating a thrust between 0.1 N and 9.1572 N to keep the copter hover in air. The thrust limits are chosen such that hovering thrusts are approximately in the center of the thrust limit.

varying running times to represent it as they are orders of magnitude larger than NN and MAMPC and look like constant in log scale.

C. Bicopter

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 $\mathbf{x} := x \quad \dot{x} \quad y \quad \dot{y} \quad \theta \quad \dot{\theta}]^\top \in \mathbb{R}^4 \times ([-\pi, \pi) \times \mathbb{R})$, where x, y are horizontal and vertical translations and θ is the angle of tilting. The input is defined as $\mathbf{u} := u_1 \quad u_2]^\top \in \mathbb{R}^2$, where u_1, u_2 are the thrust exerted by the left and right propellers, respectively. The control objective is to steer the bicopter to the origin by applying limited thrusts $u_1, u_2 \in [0.1, 9.1572]$ N at the two propellers.

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

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

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

$$x, y, \theta \in [-\pi/2, \pi/2], \quad \dot{x}, \dot{y}, \dot{\theta} \in [-1, 1].$$

Details of the NN architecture can be found in Table III in the Appendix.

Performance and efficiency of MAMPC, at various stages of training, is illustrated in Figure 8. The total and per-step running times of MAMPC are summarized in the third columns of Table I and Table II, respectively.

D. Quadcopter

As an extension to the ideal bicopter model, the last model is a quadcopter as shown in Figure 9 [4]. The goal is also to hover the copter in air but with more realistic system dynamics.

The state is defined as $\mathbf{x} := x \; \dot{x} \; y \; \dot{y} \; z \; \dot{z} \; \phi \; \dot{\phi} \; \theta \; \dot{\theta} \; \psi \; \dot{\psi}]^\top \in \mathbb{R}^6 \times ([-\pi, \pi) \times \mathbb{R})^3$, where x, y, z represent translations and ϕ, θ, ψ represent rotations corresponding to the three ZYX Euler angles roll, pitch, and yaw. The input is $\mathbf{u} = [u_1 \quad u_2 \quad u_3 \quad u_4]^\top \in \mathbb{R}^4$, where u_i is the rotational speeds of the i^{th} propeller for $i = 1, 2, 3, 4$. 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 $\text{rad} \cdot \text{s}^{-1}$ at the four propellers.

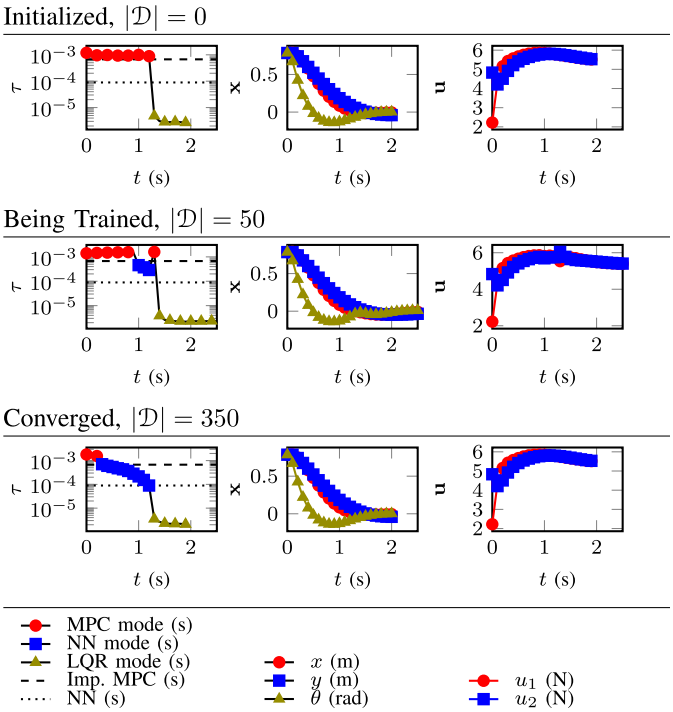


Fig. 8. Application of standard MAMPC to the bicopter problem. τ is running time. $\mathbf{x}$ is selected states and $\mathbf{u}$ is selected inputs. t is simulation time. At initialization, the NN mode fails forward verification and thus never get invoked. As the NN is being trained, its likelihood of passing forward verification increases. At last, a well trained NN takes over most of MPC steps, resulting in a more efficient operation.

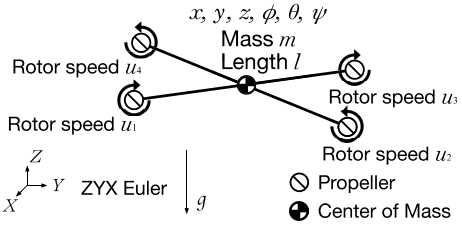


Fig. 9. Diagram of a quadcopter model. Each of the four propellers is capable of rotating at speed between $0\,\text{rad} \cdot \text{s}^{-1}$ and $313.96\,\text{rad} \cdot \text{s}^{-1}$ to keep the copter hover in air. The rotor limits are chosen such that hovering rotor rotations are approximately in the center of the rotation range.

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

$$\mathcal{D}_{\text{WP}} = \{\mathbf{x} \in \mathbb{R}^{12} \mid \|\mathbf{x}\|_2 \leq 2\}, \quad N_{\text{WP}} = 10,$$
$$\mathcal{R}_{\text{LQR}} = \{\mathbf{x} \in \mathbb{R}^{12} \mid \|\mathbf{x}\|_2 \leq 0.5\}, \quad N_{\text{LQR}} = 10.$$

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

$$x, y, z \in [-0.5, 0.5], \quad \dot{x}, \dot{y}, \dot{z} \in [-0.1, 0.1],$$
$$\phi, \theta \in [-\pi/6, \pi/6], \psi \in [-\pi/4, \pi/4], \dot{\phi}, \dot{\theta}, \dot{\psi} \in [-0.1, 0.1].$$

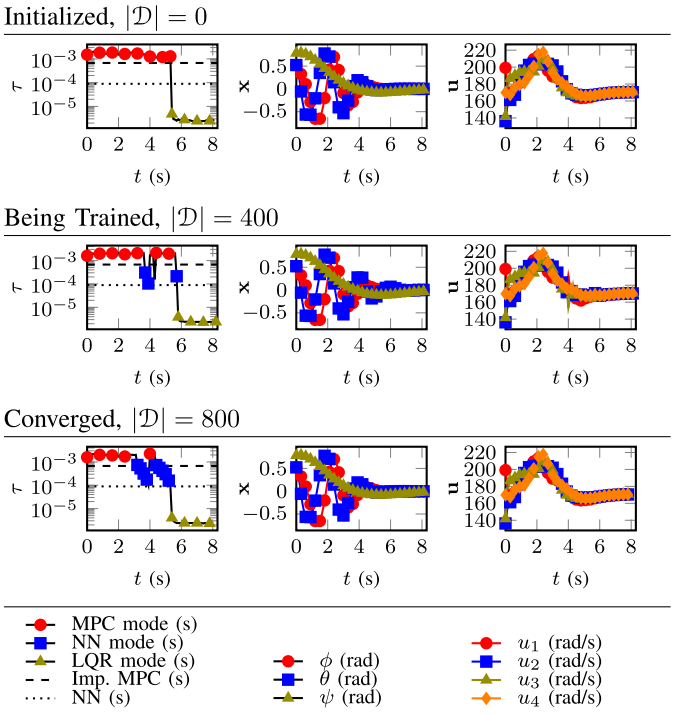


Fig. 10. Application of way-point MAMPC to the quadcopter problem. τ is running time. $\mathbf{x}$ is selected states and $\mathbf{u}$ is selected inputs. t is simulation time. At initialization, the NN mode fails forward verification and thus never get invoked. As the NN is being trained, its likelihood of passing forward verification increases. At last, a well trained NN takes over most of MPC steps, resulting in a more efficient operation.

TABLE I
SUMMARY OF TOTAL RUNNING TIME IN MILLISECOND

	Pendulum	Tri. Pend.	Bicopter	Quadcopter
Imp. MPC	10.17 ± 3.41	13.03 ± 3.44	15.58 ± 3.12	92.66 ± 11.15
MAMPC (initialized)	7.37 ± 2.40	8.54 ± 1.36	14.69 ± 3.97	93.55 ± 11.67
MAMPC (being train.)	7.21 ± 2.11	9.00 ± 1.69	20.04 ± 4.74	117.64 ± 13.74
MAMPC (converged)	3.63 ± 0.94	7.11 ± 1.63	9.66 ± 1.85	87.84 ± 16.69

TABLE II
SUMMARY OF PER-STEP RUNNING TIME IN MIRCOSECOND

	Pendulum	Tri. Pend.	Bicopter	Quadcopter
Imp. MPC	686.39 ± 248.03	695.79 ± 178.03	743.84 ± 152.67	1032.76 ± 89.52
MAMPC-MPC*	1529.29 ± 411.57	897.02 ± 226.43	1777.19 ± 404.51	2100.90 ± 401.96
MAMPC-NN*	105.23 ± 33.60	187.53 ± 51.62	418.31 ± 90.21	420.92 ± 95.02
MAMPC-LQR*	1.86 ± 0.46	3.11 ± 0.92	2.42 ± 0.55	2.65 ± 0.67
NN†	59.85 ± 21.86	79.86 ± 24.20	98.60 ± 24.88	56.25 ± 23.89

* MAMPC-MPC, MAMPC-NN, and MAMPC-LQR are the MPC, NN, and LQR modes of the MAMPC, respectively.
† NN is the NN mode of MAMPC *without* forward verification.

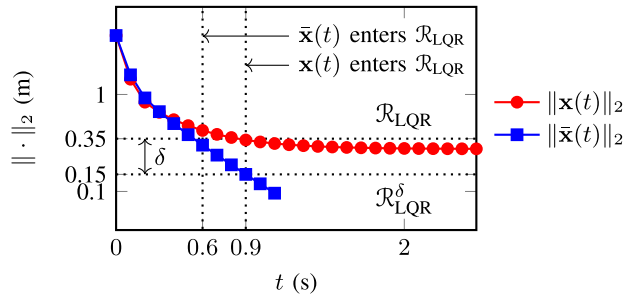


Fig. 11. Illustration of robustness margin. The *true* system $f(\cdot)$ enters $\mathcal{R}_{\text{LQR}}$ when the *modeled* system $\bar{f}(\cdot)$ enters $\mathcal{R}^{\delta}_{\text{LQR}}$. Here, δ maybe viewed as a notion of robustness margin.

Details of the NN architecture can be found in Table III in the Appendix.

Performance and efficiency of MAMPC, at various stages of training, is illustrated in Figure 10. The total and per-step running times of MAMPC are summarized in the last columns of Table I and Table II, respectively.

E. Robustness Margin

Lastly, we present how $\mathcal{R}_{\text{LQR}}$ can be shrunk to enhance the controller robustness with respect to model uncertainty.

Consider the pendulum model presented above. Denote the pendulum system model as $\dot{\mathbf{x}} = \bar{f}(\mathbf{x})$. Suppose there is an external disturbance which alters the system into $\dot{\mathbf{x}} = f(\mathbf{x}) := f(\mathbf{x}) + \mathbf{1} \cdot b$ with $b > 0$ being a constant bias term. Let the true system trajectory be $\mathbf{x}(t)$ and the trajectory predicted by $\bar{f}(\cdot)$ be $\bar{\mathbf{x}}(t)$.

Define $\mathcal{R}_{\text{LQR}} = \{\mathbf{x} \in \mathbb{R}^2 \mid \|\mathbf{x}\|_2 \leq 0.35\}$. Simulating both systems from the same initial condition $\mathbf{x}_0 = [0.65 - 4]^\top$ and $b = 0.2$, we obtain the result shown in Figure 11.

From the simulation result, we learn that the true system does not enter $\mathcal{R}_{\text{LQR}}$ until the modeled system enters $\mathcal{R}^{\delta}_{\text{LQR}}$ with $\delta = 0.2$. Hence, δ can be considered as a margin of robustness. That is, if we verify $\bar{\mathbf{x}} \in \mathcal{R}^{\delta}_{\text{LQR}}$, we can ensure that $\mathbf{x} \in \mathcal{R}_{\text{LQR}}$ despite the lack of the bias term $\mathbf{1} \cdot b$ in $\bar{f}(\cdot)$.

VI. DISCUSSIONS

As shown in Figure 4, Figure 6, Figure 8, and Figure 10 and in Table I, the MAMPC gains efficiency over time through learning on the trajectory data generated by the implicit MPC policy. Upon initialization, the MAMPC policy, despite of not being the most efficient, is immediately functional. After learning on a few samples, the NN mode behaves closer to the MPC mode but still lacks in accuracy. This period of learning sometimes results an temporary degradation in control efficacy and an temporary increase in running time. In particular, see Figure 4 and Figure 6. After training on sufficient number of samples, the NN mode eventually converges to and takes over the implicit MPC. Consequently, the overall control efficacy is recovered and computational efficiency improved.

After the NN mode converges, the resulting MAMPC is shown to have superior amortized running time but an inferior worst-case running time. As the shown in the first four rows of Table II, the LQR mode and NN mode of MAMPC is faster than implicit MPC, while the MPC mode of MAMPC is slower than the implicit method due to overhead from computing the switch conditions. Therefore, in the worst case, the MAMPC will take more time to stabilize the plant than the implicit MPC. However, if the MAMPC spends sufficient number of steps in the NN and LQR modes, it will still has faster average running time, which is the case observed in the four numerical experiments.

If the target system is capable of parallel computation, it is possible the reduce the worst-time running time down to the baseline running time of the implicit MPC. 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 and repeat for next control step. Ignoring communication overhead, one can show that the worst-case running time is roughly identical to that of the implicit MPC. Note that under this setup the MAMPC scheme is *at worst* equivalent to the dual-mode MPC in [7].

Of course, it should be noted that there is no guarantee that the resulting MAMPC will be more efficient than the implicit MPC, because there is no guarantee that the NN will converge to an accurate surrogate policy. Comparing to methods like [19], where the implicit MPC controller is completely replaced by a NN controller, we trade a lack of deterministic guarantee on stability for a lack of deterministic guarantee on running time improvement. We believe that such a trade-off is often desirable in practice: in industrial applications, safety almost always precedes efficiency.

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

1) Set Design: Identification and design of feasibility set $\mathbb{X}_0$ and attraction set $\mathcal{R}_{\text{LQR}}$ is very challenging. One hand, it is easier to just approximate them by conservative set estimates for robustness measures. For example, one can replace $\mathbb{X}_0$ and $\mathcal{R}_{\text{LQR}}$ with $\mathbb{X}_0^-$ and $\mathcal{R}_{\text{LQR}}^-$, where $\mathbb{X}_0^- \subset \mathbb{X}_0$ and $\mathcal{R}_{\text{LQR}}^- \subset \mathcal{R}_{\text{LQR}}$. On the other hand, however, to guarantee computational efficiency, one should make sets $\mathcal{R}_{\text{LQR}}, \mathcal{R}_{\text{NN}}$ 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.

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.

3) Additional Optimizations: For small to moderately sized problems, we may consider developing a fail-free MAMPC as specified in Theorem 6, 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.

Another possibility for reducing running time is to pre-compute the results of forward verification for the entire state space *offline* and look up the results *online*. This look-up table approach is in spirit similar to explicit MPC. The complexity of this approach also grows exponentially with state dimension, so it suffers the similar scalability issue like explicit MPC does.

Of course, a third possibility for reducing running time is to speed up computation of the trained neural network through techniques such as quantization, pruning, or model distillation.

It is sometimes useful to further improve the performance of the NN mode via unsupervised reinforcement learning. This is usually the case when the default implicit MPC solver is sub-optimal, typically due to lack of convexity in the objective or constraints or due to large problem dimension induced by a long prediction horizon.

Finally, one could leverage warm-started MPC with our method for enhanced performance. Warm-started MPC methods like [22] aim to use a NN to speed up the *solving* of an implicit MPC, while MAMPC seeks to *replace* the implicit MPC with a NN or a LQR. Consequently, one could directly combine the two techniques together by substituting the standard implicit MPC in a MAMPC with a warm-started implicit MPC.

VII. Conclusion

To improve *amortized* 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. Numerical experiments indicate that MAMPC often has a better *amortized* running time but a slightly prolonged *worst-case* per-step running time compared to implicit MPC method.

Appendix

A. Proof of Theorem 2

Proof: Existence of MAMPC depends on existence of LQR and NN. A LQR controller of the form (5) can always be derived from (2) because *1)* f is continuously differentiable, so it can always be linearized to produce the $\boldsymbol{A}, \boldsymbol{B}$ matrices in (5); *2)* the linearized system $(\boldsymbol{A}, \boldsymbol{B})$ is by assumption *stabilizable*, so region of attraction of LQR is nonempty, i.e., $\mathcal{R}_{\text{LQR}} \neq \emptyset$; *3)* as for $\boldsymbol{Q}, \boldsymbol{R}$ we only require $\boldsymbol{Q} \succeq 0, \boldsymbol{R} \succ 0$, which is always possible; *4)* lastly, optimization problem (5) is just a relaxation of problem (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 $\mathbb{X}_0$, we partition $\mathbb{X}_0$ into three regions as follows

$$\mathbb{X}_0 = \mathcal{R}_{\text{LQR}} \cup \mathcal{R}_{\text{NN}} \cup \mathcal{R}_{\text{MPC}},$$

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 $\mathbb{X}_0$, i.e.,

$\mathcal{R}_{\text{MPC}} := X_0 \setminus (\mathcal{R}_{\text{LQR}} \cup \mathcal{R}_{\text{NN}})$. By construction, $\mathcal{R}_{\text{LQR}}$, $\mathcal{R}_{\text{NN}}$, and $\mathcal{R}_{\text{MPC}}$ are mutually disjoint.

For every state $\mathbf{x}[0] \in \mathcal{R}_{\text{LQR}}$, u_{LQR} is invoked and the closed-loop system (11) will be equivalent to (7), which is locally asymptotically stable.

For every state $\mathbf{x}[0] \in \mathcal{R}_{\text{NN}}$, u_{NN} is invoked and the closed-loop system (11) will be equivalent to (10) until the state enters $\mathcal{R}_{\text{LQR}}$ in no more than N_{LQR} steps. The system will stay in $\mathcal{R}_{\text{LQR}}$ and will be taken to the origin by u_{LQR} as $i \to \infty$. Consequently, the closed-loop system (11) is locally asymptotically stable in $\mathcal{R}_{\text{MPC}} \cup \mathcal{R}_{\text{NN}}$.

For every state $\mathbf{x}[0] \in \mathcal{R}_{\text{MPC}}$, u_{MPC} is invoked and the closed-loop system (11) will be equivalent to (4) for at least one time step. In the next time step, if the state enters $\mathcal{R}_{\text{LQR}} \cup \mathcal{R}_{\text{NN}}$, then the closed-loop system will converge to the origin as it is locally asymptotically stable in $\mathcal{R}_{\text{LQR}} \cup \mathcal{R}_{\text{NN}}$; otherwise, $\mathcal{R}_{\text{MPC}}$ will be invoked again. Because the closed-loop system (4) is locally asymptotically stable, even if MPC never brings the state into $\mathcal{R}_{\text{NN}}$, it will eventually steer the system into $\mathcal{R}_{\text{LQR}}$ in finite time, which will then be taken to the origin by u_{LQR}. Consequently, the closed-loop system (11) is locally asymptotically stable in $\mathbb{X}_0$. □

B. Proof of Theorem 3

Proof: The first part of the proof on existence is identical to the one in Theorem 2.

To prove stability of alternating-authority MAMPC, we partition $\mathbb{X}_0$ into two disjoint regions $\mathbb{X}_0 = \mathcal{R}_{\text{LQR}} \cup \mathcal{R}_{\text{AA}}$, where $\mathcal{R}_{\text{AA}} := X_0 \setminus \mathcal{R}_{\text{LQR}}$.

For every state $\mathbf{x}[0] \in \mathcal{R}_{\text{LQR}}$, the closed-loop system (12) is identical to closed-loop system (11), which is locally asymptotically stable, as shown in Theorem (2).

Next, we prove by contradiction that the system will never escape $\mathbb{X}_0$. Suppose there exits an escaping control $\bar{\mathbf{u}}$ such that $\bar{\mathbf{x}} \in \mathbb{X}_0$ but $f(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \notin \mathbb{X}_0$. This escaping control $\bar{\mathbf{u}}$ must be produced by either u_{MPC}, u_{NN}, or u_{LQR}, that is, $\bar{\mathbf{u}} = u_{\text{MPC}}(\bar{\mathbf{x}})$, $\bar{\mathbf{u}} = u_{\text{NN}}(\bar{\mathbf{x}})$, or $\bar{\mathbf{u}} = u_{\text{LQR}}(\bar{\mathbf{x}})$. If $\bar{\mathbf{u}} = u_{\text{MPC}}(\bar{\mathbf{x}})$, then u_{MPC} is not persistently feasible in $\mathbb{X}_0$, which is not possible because we assume that u_{MPC} satisfies Theorem (1). If $\bar{\mathbf{u}} = u_{\text{NN}}(\bar{\mathbf{x}})$, then the following switch condition must hold: $f(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \in \mathbb{X} \subseteq \mathbb{X}_0$. However, this is not possible since $\bar{\mathbf{u}}$ is assumed to be an escaping control, that is, $f(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \notin \mathbb{X}_0$. If $\bar{\mathbf{u}} = u_{\text{LQR}}(\bar{\mathbf{x}})$, then there must be $\bar{\mathbf{x}} \in \mathcal{R}_{\text{LQR}}$. Since $\mathcal{R}_{\text{LQR}}$ is positively invariant, we must have $f(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \in \mathcal{R}_{\text{LQR}} \subset \mathbb{X}_0$, which is not possible since $f(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \notin \mathbb{X}_0$. Therefore, the system will never escape $\mathbb{X}_0$. □

C. Proof of Theorem 4

Proof: The first part of the proof on existence is identical to the one in Theorem 2.

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

$$\mathbb{X}_0 = \mathcal{R}_{\text{LQR}} \cup \mathcal{R}^i_{\text{NN}} \cup \mathcal{R}^i_{\text{MPC}} \cup \mathcal{R}^o_{\text{NN}} \cup \mathcal{R}^o_{\text{MPC}},$$

where $\mathcal{R}^i_{\text{NN}}$ is the set of states in or on the boundary of $\mathcal{D}_{\text{WP}}$ where the NN is invoked; $\mathcal{R}^o_{\text{NN}}$ is the set of states in $\mathbb{X}_0 \setminus \mathcal{D}_{\text{WP}}$ where the NN is invoked; $\mathcal{R}^i_{\text{MPC}} := D_{\text{WP}} \setminus (\mathcal{R}_{\text{LQR}} \cup \mathcal{R}^i_{\text{NN}})$; and $\mathcal{R}^o_{\text{MPC}} := \mathbb{X}_0 \setminus \mathcal{D}_{\text{WP}}) \setminus \mathcal{R}^o_{\text{NN}}$.

TABLE III
NEURAL NETWORKS ARCHITECTURE

	Layer	Dimension	Activation
Pendulum	Input	$\mathbf{x} \in \mathbb{R}^2$	
	Hidden layer 1	10 neurons + bias	ReLU
	Hidden layer 2	10 neurons + bias	ReLU
	Output	$u \in \mathbb{R}$	Linear
Tri. Pend.	Input	$\mathbf{x} \in \mathbb{R}^6$	
	Hidden layer 1	20 neurons + bias	ReLU
	Hidden layer 2	10 neurons + bias	ReLU
	Hidden layer 3	20 neurons + bias	ReLU
	Output	$\mathbf{u} \in \mathbb{R}^3$	Linear
Bicopter	Input	$\mathbf{x} \in \mathbb{R}^6$	
	Hidden layer 1	20 neurons + bias	ReLU
	Hidden layer 2	10 neurons + bias	ReLU
	Hidden layer 3	20 neurons + bias	ReLU
	Output	$\mathbf{u} \in \mathbb{R}^2$	Linear
Quadcopter	Input	$\mathbf{x} \in \mathbb{R}^{12}$	
	Hidden layer 1	20 neurons + bias	ReLU
	Hidden layer 2	10 neurons + bias	ReLU
	Hidden layer 3	10 neurons + bias	ReLU
	Hidden layer 4	20 neurons + bias	ReLU
	Output	$\mathbf{u} \in \mathbb{R}^4$	Linear

For every state $\mathbf{x} \in \mathcal{R}_{\text{LQR}}$, the closed-loop system (14) is identical to closed-loop system (11), which is locally asymptotically stable, as shown in Theorem (2).

Next, we prove that the system will eventually enter and stay within $\mathcal{D}^+_{\text{WP}}$.

For every state $\mathbf{x}[0] \in \mathbb{X}_0 \setminus \mathcal{D}_{\text{WP}}$, either NN or MPC will steer the system into $\mathcal{D}^+_{\text{WP}}$ in finite time. If NN is ever invoked, the system (14) will be taken into $\mathcal{D}_{\text{WP}}$ in no more than N_{WP} steps. Otherwise, MPC will bring the system (14) into $\mathcal{D}_{\text{WP}}$ in finite time since the MPC is locally asymptotically stable in $\mathbb{X}_0$.

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

This escaping control $\bar{\mathbf{u}}$ must be produced by either u_{MPC}, u_{NN}, or u_{LQR}, that is, $\bar{\mathbf{u}} = u_{\text{MPC}}(\mathbf{x}[i-1])$, $\bar{\mathbf{u}} = u_{\text{NN}}(\mathbf{x}[i-1])$, or $\bar{\mathbf{u}} = u_{\text{LQR}}(\mathbf{x}[i-1])$. If $\bar{\mathbf{u}} = u_{\text{MPC}}(\mathbf{x}[i-1])$, then the close-loop system (14) is equivalent to system (4). Because $\mathbf{x}[i] \notin \mathcal{D}^+_{\text{WP}}$, we have

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

but this is not possible because V is a Lyapunov function of the system (4), i.e.,

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

If $\bar{\mathbf{u}} = u_{\text{NN}}(\mathbf{x}[i-1])$, then $f(\mathbf{x}[i-1], u_{\text{NN}}(\mathbf{x}[i-1])) \notin \mathcal{D}_{\text{WP}}$, which is not possible because invocation of NN implies that $f(\mathbf{x}[i-1], u_{\text{NN}}(\mathbf{x}[i-1])) \in \mathcal{D}_{\text{WP}}$. If $\bar{\mathbf{u}} = u_{\text{LQR}}(\mathbf{x}[i-1])$, then by assumption $\lim_{i\to\infty} \mathbf{x}[i]$ does not exist. However, this is impossible because $\lim_{i\to\infty} \mathbf{x}[i] = \mathbf{0}$ by the local

asymptotic stability of the system (14) in $\mathcal{R}_{\text{LQR}}$. Consequently, the assumption must be false. □

D. Architecture of Neural Networks

Please find the details of neural network designs in Tables III. All networks are multilayer perceptrons with mean square loss and are trained with the Levenberg-Marquardt algorithm.

REFERENCES

[1] B. J. Rawlings, Q. D. Mayne, and M. M. Diehl, *Model Predictive Control: Theory, Computation and Design*. Santa Barbara, CA, USA: Nob Hill Publishing, 2017.

[2] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*. Cambridge, U.K.: Cambridge Univ. Press, 2017.

[3] F. Borrelli, P. Falcone, T. Keviczky, J. Asgari, and D. Hrovat, "MPC-based approach to active steering for autonomous vehicle systems," *Int. J. Veh. Auto. Syst.*, vol. 3, nos. 2–4, pp. 265–291, 2005.

[4] K. Alexis, G. Nikolakopoulos, and A. Tzes, "Model predictive quadrotor control: Attitude, altitude and position experimental studies," *IET, Control Theory Appl.*, vol. 6, no. 12, pp. 1812–1827, Aug. 2012.

[5] S. Kuindersma et al., "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Auton. Robots*, vol. 40, no. 3, pp. 429–455, Mar. 2016.

[6] 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 *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 2245–2252.

[7] H. Michalska and D. Q. Mayne, "Robust receding horizon control of constrained nonlinear systems," *IEEE Trans. Autom. Control*, vol. 38, no. 11, pp. 1623–1633, Nov. 1993.

[8] M. Diehl, H. G. Bock, and J. P. Schlöder, "A real-time iteration scheme for nonlinear optimization in optimal feedback control," *SIAM J. Control Optim.*, vol. 43, no. 5, pp. 1714–1736, Jul. 2005.

[9] C. Mastalli et al., "Crocoddyl: An efficient and versatile framework for multi-contact optimal control," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 2536–2542.

[10] B. Houska, H. J. Ferreau, and M. Diehl, "An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range," *Automatica*, vol. 47, no. 10, pp. 2279–2285, Oct. 2011.

[11] J. Mattingley and S. Boyd, "CVXGEN: A code generator for embedded convex optimization," *Optim. Eng.*, vol. 13, no. 1, pp. 1–27, 2012.

[12] S. Richter, C. N. Jones, and M. Morari, "Real-time input-constrained MPC using fast gradient methods," in *Proc. 48h IEEE Conf. Decis. Control (CDC) 28th Chin. Control Conf.*, Dec. 2009, pp. 7387–7393.

[13] T. Faulwasser, T. Weber, P. Zometa, and R. Findeisen, "Implementation of nonlinear model predictive path-following control for an industrial robot," *IEEE Trans. Control Syst. Tech.*, vol. 25, no. 4, pp. 1505–1511, Jul. 2017.

[14] S. Kleff, A. Meduri, R. Budhiraja, N. Mansard, and L. Righetti, "High-frequency nonlinear model predictive control of a manipulator," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 7330–7336.

[15] S. Summers, C. N. Jones, J. Lygeros, and M. Morari, "A multiresolution approximation method for fast explicit model predictive control," *IEEE Trans. Autom. Control*, vol. 56, no. 11, pp. 2530–2541, Nov. 2011.

[16] M. Kvasnica, J. Hledík, I. Rauová, and M. Fikar, "Complexity reduction of explicit model predictive control via separation," *Automatica*, vol. 49, no. 6, pp. 1776–1781, 2013.

[17] T. Parisini and R. Zoppoli, "A receding-horizon regulator for nonlinear systems and a neural approximation," *Automatica*, vol. 31, no. 10, pp. 1443–1451, Oct. 1995.

[18] T. Parisini, M. Sanguineti, and R. Zoppoli, "Nonlinear stabilization by receding-horizon neural regulators," *Int. J. Control*, vol. 70, no. 3, pp. 341–362, Jan. 1998.

[19] X. Zhang, M. Bujarbaruah, and F. Borrelli, "Safe and near-optimal policy learning for model predictive control using primal-dual neural networks," in *Proc. Amer. Control Conf. (ACC)*, Jul. 2019, pp. 354–359.

[20] J. A. Paulson and A. Mesbah, "Approximate closed-loop robust model predictive control with guaranteed stability and constraint satisfaction," *IEEE Control Syst. Lett.*, vol. 4, no. 3, pp. 719–724, Jul. 2020.

[21] B. Karg and S. Lucia, "Stability and feasibility of neural network-based controllers via output range analysis," in *Proc. 59th IEEE Conf. Decis. Control (CDC)*, Dec. 2020, pp. 4947–4954.

[22] 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*, vol. 135, Jan. 2022, Art. no. 109947.

[23] R. Zoppoli, M. Sanguineti, G. Gnecco, and T. Parisini, *Neural Approximations for Optimal Control and Decision*. Cham, Switzerland: Springer, 2020.

[24] J. Nubert, J. Kohler, V. Berenz, F. Allgower, and S. Trimpe, "Safe and fast tracking on a robot manipulator: Robust MPC and neural network control," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 3050–3057, Apr. 2020.

[25] P. D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–15.

[26] D. O. B. Anderson and J. B. Moore, *Optimal Control: Linear Quadratic Methods*. New York, NY, USA: Dover, 2018, pp. 53–54.

[27] K. H. Khalil, *Nonlinear Systems*, 3rd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2002.

[28] H.-D. Chiang, M. W. Hirsch, and F. F. Wu, "Stability regions of nonlinear autonomous dynamical systems," *IEEE Trans. Autom. Control*, vol. AC-33, no. 1, pp. 16–27, Jan. 1988.

[29] U. Topcu, A. Packard, and P. Seiler, "Local stability analysis using simulations and sum-of-squares programming," *Automatica*, vol. 44, no. 10, pp. 2669–2675, Oct. 2008.

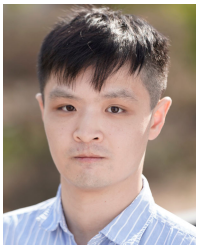
[30] B. K. Colbert and M. M. Peet, "Using trajectory measurements to estimate the region of attraction of nonlinear systems," in *Proc. IEEE Conf. Decis. Control (CDC)*, Dec. 2018, pp. 2341–2347.

[31] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[32] S. M. Richards, F. Berkenkamp, and A. Krause, "The Lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems," in *Proc. Conf. Robot Learn.*, 2018, pp. 466–476.

[33] S. Chen, M. Fazlyab, M. Morari, G. J. Pappas, and V. M. Preciado, "Learning Lyapunov functions for hybrid systems," in *Proc. 24th Int. Conf. Hybrid Syst., Comput. Control*, May 2021, pp. 1–11.

[34] F. Wu. (2021). *Memory-Augmented Model Predictive Control: Numerical Experiments*. [Online]. Available: https://github.com/fywu85/mampc



Fangyu Wu received the B.S. and M.S. degrees in civil engineering from the University of Illinois at Urbana–Champaign in 2015 and 2018, respectively, and the M.Eng. degree in electrical engineering and computer sciences from the University of California at Berkeley in 2019, where he is currently pursuing the Ph.D. degree in electrical engineering and computer sciences.

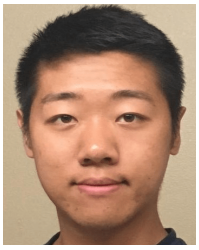
His research interests include optimal control, motion planning, and scalable scheduling for multi-agent robotic systems

Guanhua Wang received the B.Eng. degree in computer science from Southeast University, China, in 2012, the M.Phil. degree in computer science and engineering from The Hong Kong University of Science and Technology in 2015, under the supervision of Prof. Lionel M. Ni, and the Ph.D. degree in electrical engineering and computer sciences from the University of California at Berkeley in 2022, under the supervision of Prof. Ion Stoica.

He was a member of the RISELab, University of California at Berkeley. His research interests include the intersection of machine learning and systems, encompassing topics, such as fast collective communication schemes for model synchronization, efficient parallel model training, and real-time model serving.

Siyuan Zhuang received the B.Eng. degree in computer science from the University of Science and Technology, China, in 2018. He is currently pursuing the Ph.D. degree in computer science with the University of California at Berkeley, under the supervision of Prof. Dawn Song and Prof. Ion Stoica.

His research interests include machine learning systems and distributed systems, exploring the intricacies and challenges of these interrelated fields.



Kehan Wang received the B.A. degree in computer science from the University of California at Berkeley in 2021, where he is currently pursuing the M.S. degree in electrical engineering and computer sciences.

In 2020, he was with Microsoft, contributing to the Microsoft teams platform. He was a Research Assistant with the Berkeley Artificial Intelligence Research Laboratory. His research interests include computer vision and deep learning.

Alexander Keimer received the Diploma degree in mathematics and the Ph.D. degree from Friedrich-Alexander-Universität Erlangen-Nürnberg in 2008 and 2014, respectively.

He was a Senior Researcher with the Institute of Transportation Studies, University of California at Berkeley. His research interests include nonlocal conservation laws, optimal control with partial differential equations, and traffic flow modeling.

Ion Stoica received the M.S. degree in electrical engineering and computer science from the Polytechnic University of Bucharest in 1989 and the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University in 2000, under the supervision of Hui Zhang.

In past, he worked with Apache Spark, Apache Mesos, Tachyon, Chord DHT, and Dynamic Packet State. In 2013, he co-founded Databricks, a startup to commercialize technologies for big data processing. In 2006, he co-founded Conviva, a startup to commercialize technologies for large scale video distribution. He is currently a Professor with the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley. His research interests include cloud computing and networked computer systems.

Prof. Stoica is an ACM Fellow. He has received numerous awards, including the SIGOPS Hall of Fame Award in 2015, the SIGCOMM Test of Time Award in 2011, and the ACM Doctoral Dissertation Award in 2001.

Alexandre Bayen (Fellow, IEEE) received the Engineering degree in applied mathematics from Ecole Polytechnique, France, in 1998, and the M.S. and Ph.D. degrees in aeronautics and astronautics from Stanford University in 1999 and 2004, respectively.

He was a Visiting Researcher with the NASA Ames Research Center from 2000 to 2003. From January 2004 to December 2004, he worked as the Research Director of the Autonomous Navigation Laboratory, Laboratoire de Recherches Balistiques et Aerodynamiques, Ministere de la Defense, Vernon, France, where he holds the Rank of Major. He has been with the Faculty of University of California at Berkeley since 2005. He is currently the Liao-Cho Professor of Engineering with the University of California at Berkeley, a Professor of Electrical Engineering and Computer Science and Civil and Environmental Engineering, the Director of the Institute of Transportation Studies, and the Faculty Scientist of Mechanical Engineering with the Lawrence Berkeley National Laboratory. He has authored two books and more than 200 articles in peer-reviewed journals and conferences.