Deep Learning-based Nonlinear Model Predictive Control with Offset-free Tracking for Embedded Applications

Kimberly J. Chan*, Joel. A Paulson**, and Ali Mesbah*

*Department of Chemical and Biomolecular Engineering, University of California, Berkeley

**Department of Chemical and Biomolecular Engineering, The Ohio State University

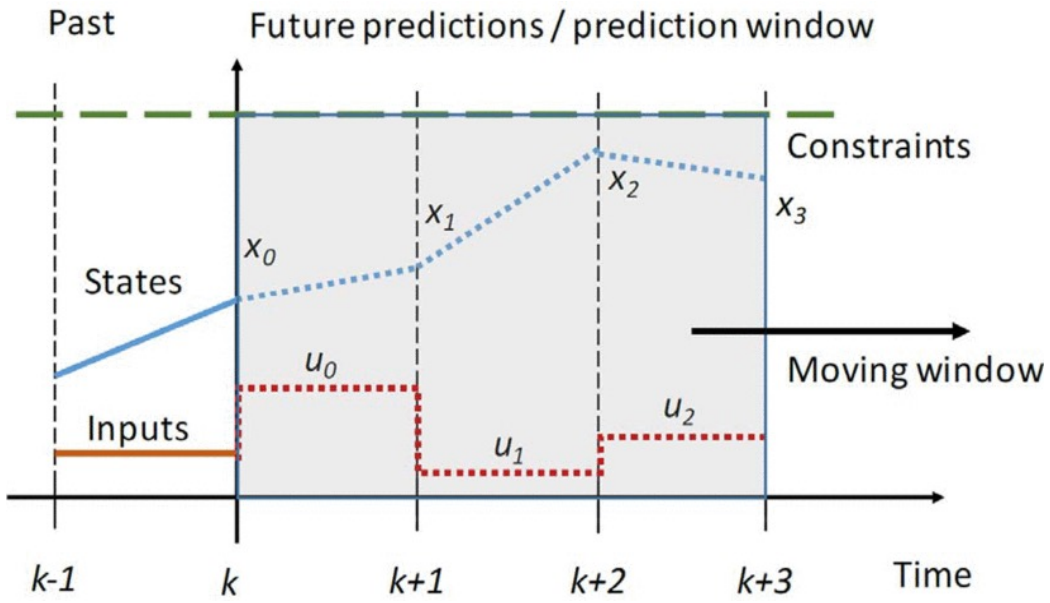
American Control Conference 2021

Berkeley
UNIVERSITY OF CALIFORNIA

Model Predictive Control

Model Predictive Control (MPC)

- **Idea**: repeatedly compute an optimal set of control inputs to reach an objective in a receding-horizon fashion



Advantages:

- Use system model and account for constraints
- Handle general control objectives
- Incorporate system uncertainties

Disadvantages:

- Generally computationally demanding
- Stability and feasibility

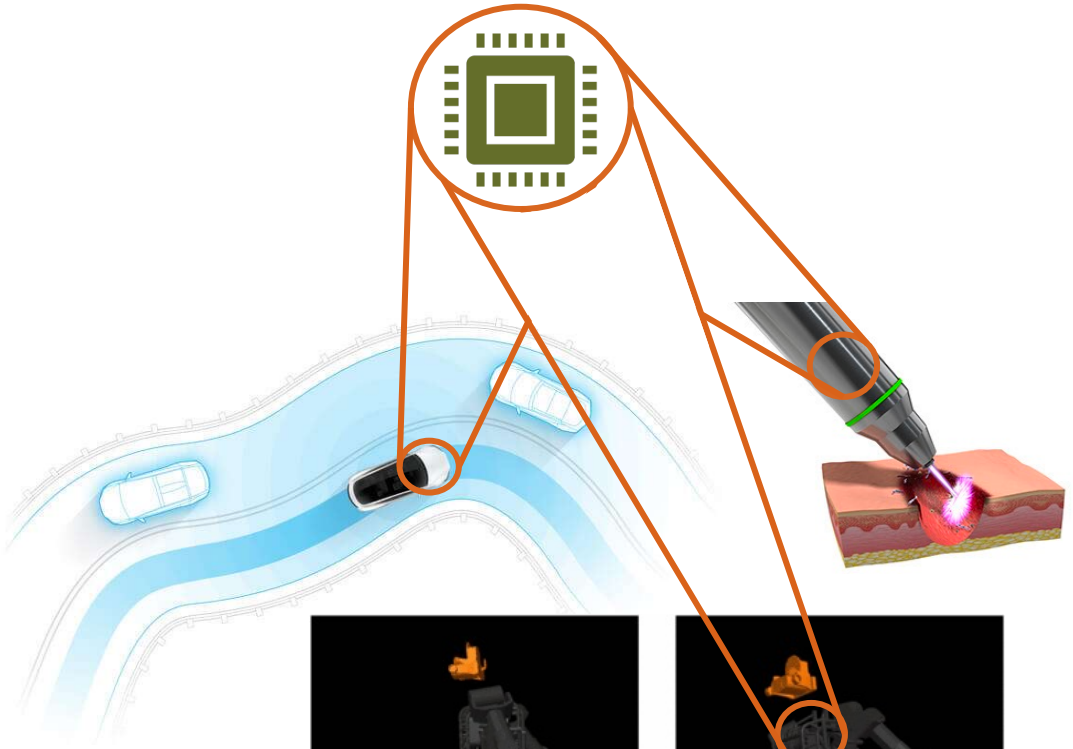
Fast MPC for Embedded Systems

Qualities of Emerging Applications

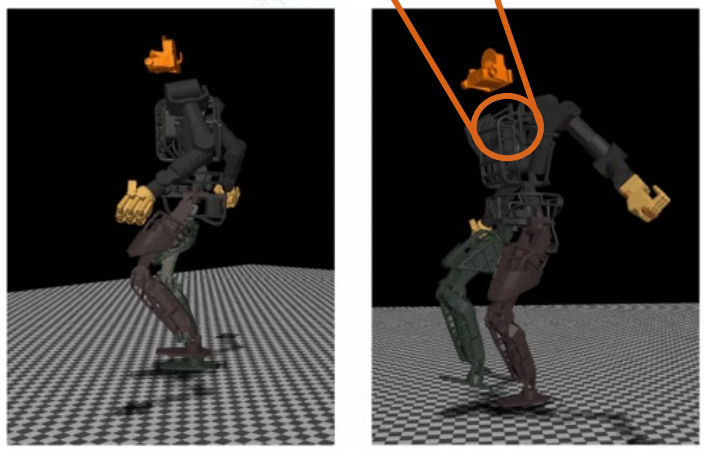
- Large-scale
- Highly nonlinear
- Fast dynamics
- Safety-critical

Challenges of MPC on Embedded Systems

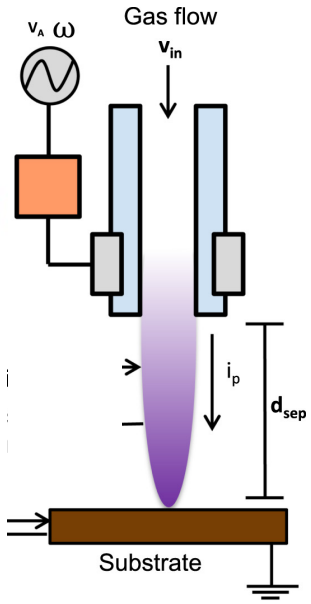
- Real-time computation
- Resource-limited hardware
- Numerical robustness at low computational accuracy



Tesla, 2021



(a) Right stance (b) Right step



Neoplas tools, 2020
Gidon *et al.*, 2017

T. Erez *et al.*, 2013

Goal: Demonstrate an **embedded** control system that results in **fast** operation with **offset elimination** towards automated control.

MPC for Tracking

The Tracking Problem

- Nominally, MPC is designed to control a system and reach a target steady state
- *Change* the objective?
- **Issue**: feasibility? → failure to track
- **Solution**: Tracking formulation[1]
  - Define $(x_s, u_s, y_s) \rightarrow \theta$, where $x_s = f(x_s, u_s)$, $y_s = h(x_s)$
  - Modify the cost function
  - Re-formulate the optimization based on this new cost function
- Computation of this optimization problem gets **expensive**! (especially for nonlinear systems)

System

$$x^+ = f(x,u)$$
$$y = h(x)$$

$$\min_{\boldsymbol{u}} V_N(\boldsymbol{u}, \theta; x, r)$$

s.t. $x^+ = f(x,u)$

$x(0) = x_0$

$x(i) \in \mathcal{X}, \forall i \in [0, N]$

$u(i) \in \mathcal{U}, \forall i \in [0, N-1]$

$\boldsymbol{u} = [u_0, u_1, \ldots, u_{N-1}]$

$x_s = g_x(\theta)$

$u_s = g_u(\theta)$

$$V_N(\boldsymbol{u}, \theta; x, r) = \sum_{i=0}^{N-1} \ell(x_i - x_s, u_i - u_s) + V_f(x_N - x_s, y_s) + V_O(r - y_s)$$

Control Law

$$\kappa_N(x, r) = u^\star(0; x, r)$$

[1]D. Limon *et al.*, 2009

How to embed?

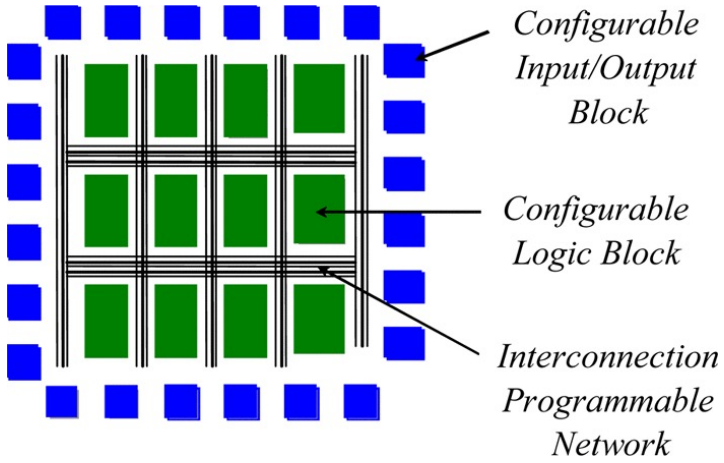
Desirable Properties:

- Low computational complexity
- Small memory footprint

→ **Explicit MPC[1]**
BUT not for nonlinear model and cost functions
→ **Approximate MPC[2,3]**

Field Programmable Gate Array (FPGA)

- **Integrated circuit** (chip) composed of *configurable logic blocks* (CLBs) that are linked and **reprogrammable**



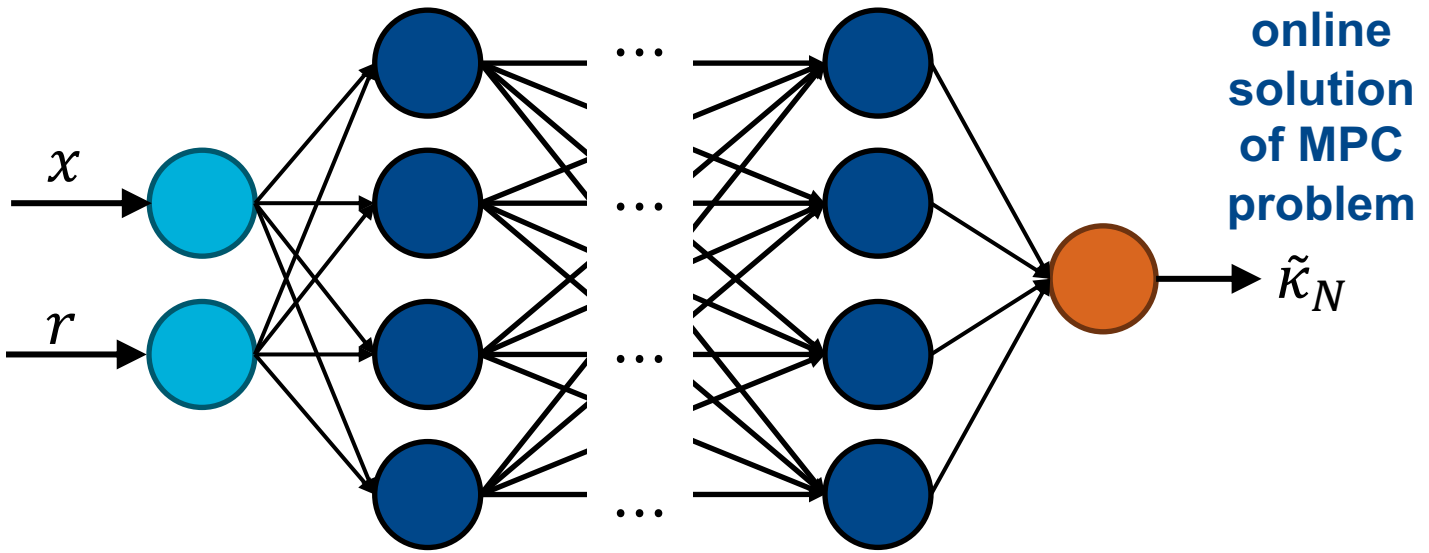
Generic Architecture of FPGA

- Resource-limited Hardware ⇒ limited power, memory, computational resources
  - Cannot handle predictive control algorithms at fast sampling rates AND ensure constraint satisfaction

[1]Bemporad *et al.*, 2002
[2]Parisini and Zoppoli, 1995
[3]Karg and Lucia, 2018

Approximate MPC using Deep Neural Networks (DNNs)

$$\underbrace{\tilde{\kappa}_N(x,r)}_{\text{Explicit control law}} \approx \underbrace{\kappa_N(x,r)}_{\text{MPC law (implicit)}}$$



Replaces online solution of MPC problem

Desirable properties

- Cheap to evaluate
- Low memory footprint

Problem: Induced error from approximation → no longer offset free!

$$\|\kappa_N(x,r) - \tilde{\kappa}_N(x,r)\| \leq \epsilon_{\text{approx}}$$

S. Chen *et al.*, 2018
B. Karg and S. Lucia, 2020

A. Bonzanini *et al.*, 2020
M. Hertneck *et al.*, 2018

J. A. Paulson and A. Mesbah, 2020

Offset Elimination for DNN-based Approximations

- Introduce a correction factor based on a **steady-state target tracker**
- Steady-state target tracker is another optimization problem
  - ℓ_s is a steady-state cost function, assumed to guarantee this optimization problem has a unique solution
- *Theorem 1*: Assume $\mathcal{X} \times \mathcal{U}$ contains at least one feasible steady state, and $V_O(\varepsilon)$ satisfies the properties of an exact penalty function[1].
  - $\kappa_{OF}(x, r)$ exists for any r and guarantees a closed-loop equilibrium point (x_∞, u_∞) exists such that $r = h(x_\infty)$

$$\kappa_{OF}(x, r) = \tilde{\kappa}_N(x, r) + (u_s^\star(r) - \tilde{\kappa}_N(x_s^\star(r), r))$$

$$\theta^\star(r) = \underset{\theta}{\operatorname{argmin}} \ \ell_s(y_s, u_s) + V_O(r - y_s)$$
$$\text{s.t.} \quad x_s = f(x_s, u_s)$$
$$y_s = h(x_s)$$
$$(x_s, u_s) \in \mathcal{X} \times \mathcal{U}$$

- Note: Theorem 1 demonstrates that an offset-free equilibrium point exists, but no guarantee that the controller converges to this equilibrium.
  - In practice, some stability analysis and/or performance validation should be performed (e.g., see [2] and [3])

[1]E. C. Kerrigan and J. M. Maciejowski, 2000
[2]J. A. Paulson and A. Mesbah, 2020
[3]D. Krishnamoorthy *et al.*, 2021

Steady-state Target Tracker Properties

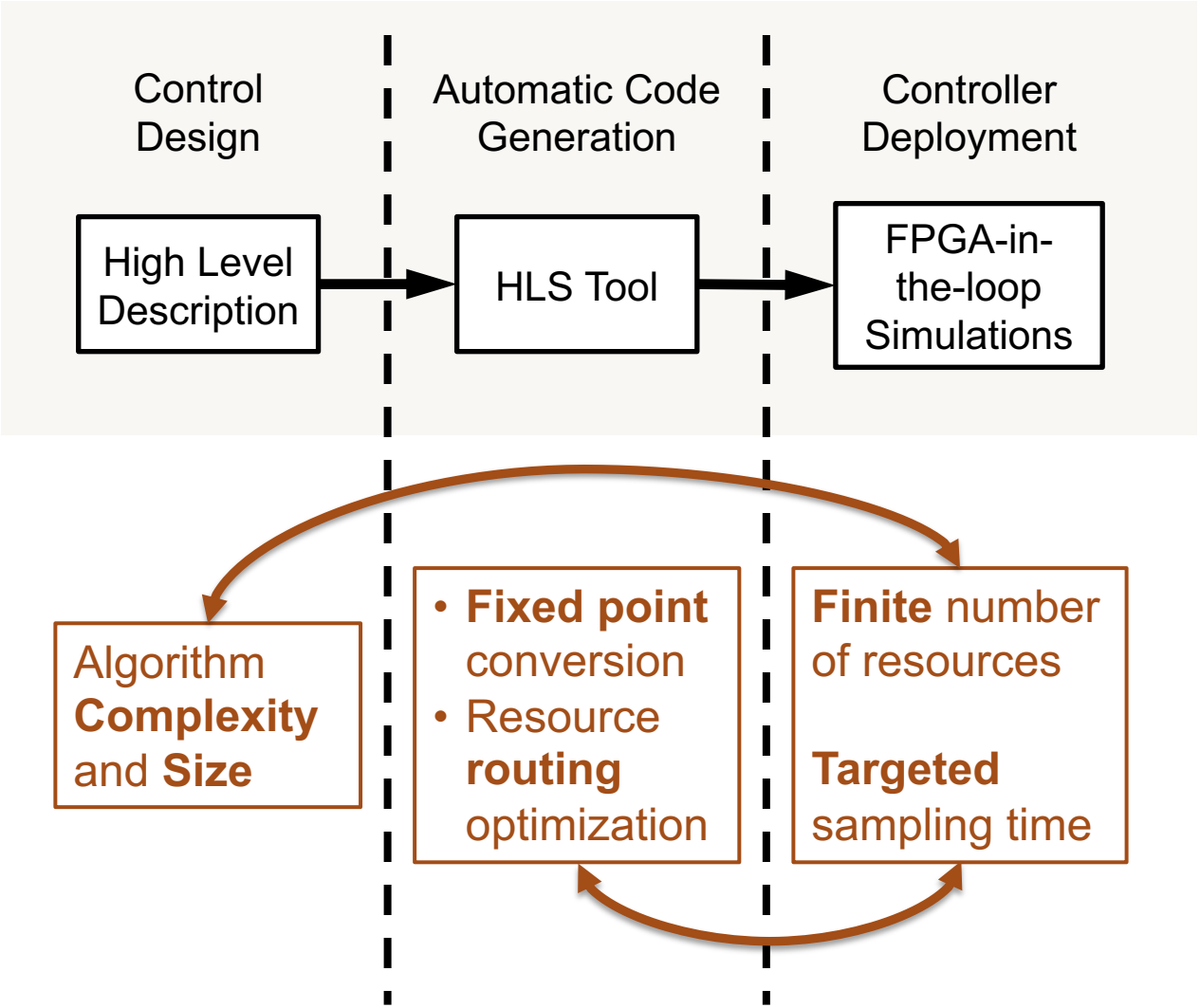
$$\kappa_{OF}(x,r) = \tilde{\kappa}_N(x,r) + (u_s^\star(r) - \tilde{\kappa}_N(x_s^\star(r),r))$$

- When $\epsilon_{\text{approx}} = 0$, correction term → 0, so $\kappa_{OF}(x,r) = \tilde{\kappa}_N(x,r) = \kappa_N(x,r)$
  - Correction term corrects for bias around the equilibrium condition
  - As $\epsilon_{\text{approx}} \to 0$, the closed-loop system becomes more attracted to the equilibrium manifold
  - Consequence: Any ϵ_{approx} can be overcome as no other equilibrium with the same steady-state input exists
- Desirable properties of the steady-state tracker:
  - Small scale
  - Static

FPGA Implementation

- Embedding requires translation to/generation of hardware level code
- High Level Synthesis (HLS)
  - Takes a high level description (e.g. MATLAB m-code, C/C++) and automatically generates hardware code

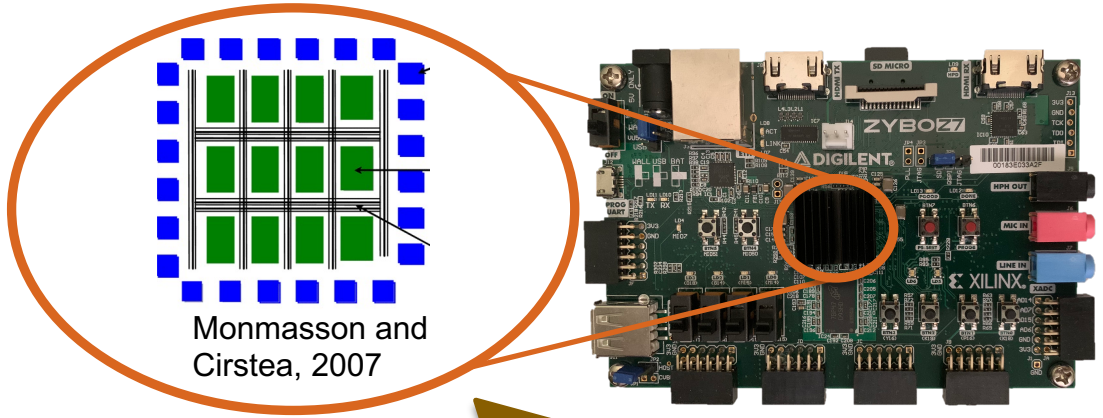
- Recall: **resource-limited**
- Design considerations ("tuning knobs") at each step affect the feasibility of final implementation.



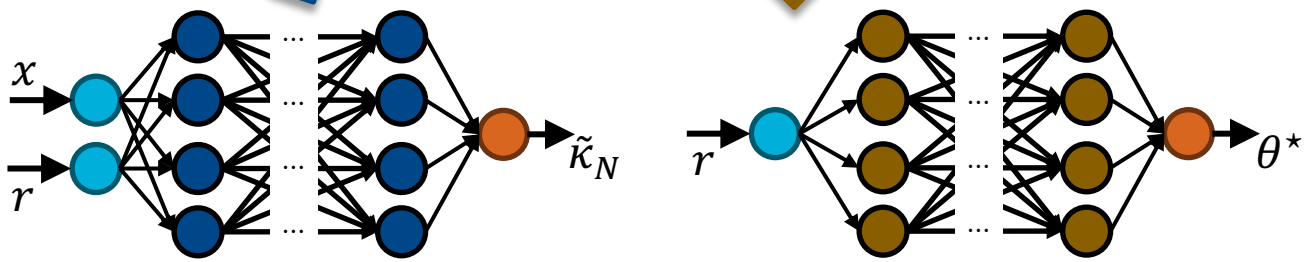
Case Study

- **Benchmark bilinear system** with plant-model mismatch[1]
  - SISO with controlled output: $[x]_1$
  - Control objective: track multiple reference values over time
- **Embedded Device**: programmable logic (PL) side of a Zybo Z7 development board by Digilent

$$[x]_1^+ = 0.95[x]_1 - 0.25[x]_1[x]_2 + [x]_2$$
$$[x]_2^+ = 0.7[x]_2 + u$$



- Embedding $\kappa_{OF}(x, r)$ consists of **both DNN and target tracker** → consider approximating target tracker with a **second DNN**

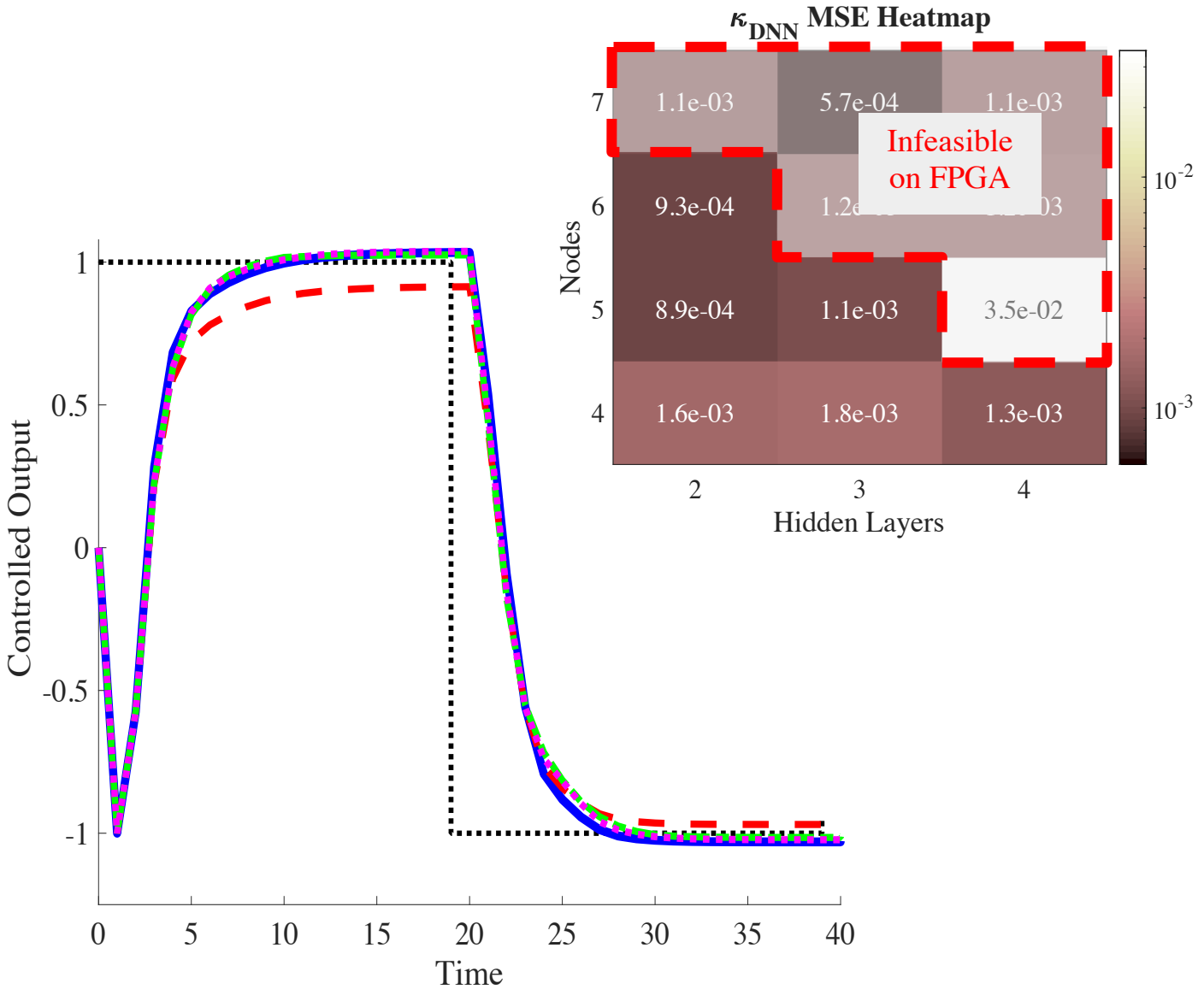


[1]M. Morari and U. Maeder, 2012

Closed-loop Simulations: Setpoint Tracking Results

- **Optimally choose hyperparameters** for each DNN
  - Choose best DNN for target tracker
  - Utilize remaining resources for $\tilde{\kappa}_N(x, r)$

- **Offset-eliminated tracking** can be achieved:
  - $\kappa_{OF}(x, r)$ on CPU: solid blue
  - $\tilde{\kappa}_N(x, r)$ [without correction] on CPU: dashed red
  - $\kappa_N(x, r)$ on CPU: solid green
  - $\kappa_{OF}(x, r)$ on FPGA: dotted magenta

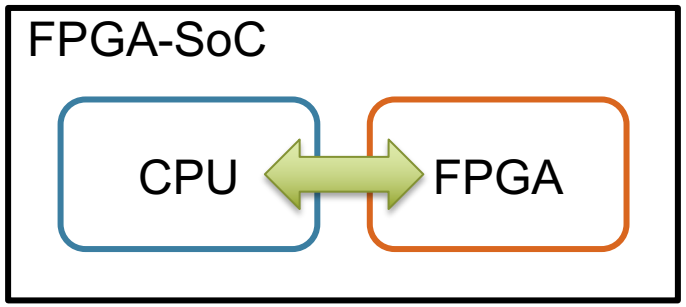


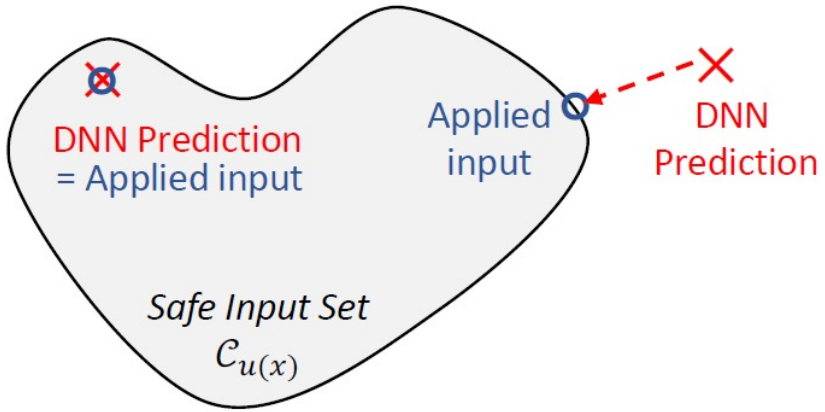
Conclusions and Future Work

- Proposed a **novel correction factor** to **eliminate offset** in a DNN-based tracking MPC
- Demonstrated an **embedded** version of the modified control law on an FPGA device

Future Work:

- Develop a *systematic framework* for embedding deep learning-based MPC algorithms on resource-limited devices
- Explore emerging *system-on-chip* (SoC) architectures
- Investigate *robust* control formulation to explicitly account for uncertainty
- Incorporate *guaranteed constraint satisfaction* techniques (e.g. using projection-based strategies[1])





[1]J. A. Paulson and A. Mesbah, 2020