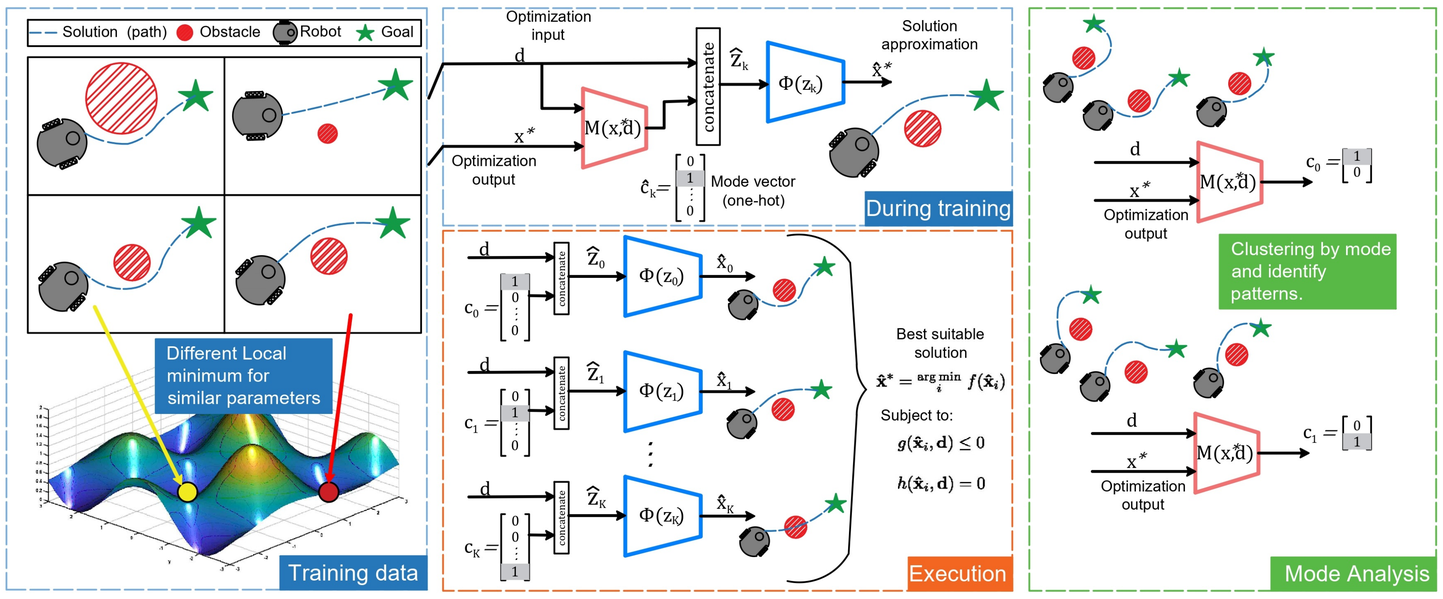
# Non-Convex Model Predictive Control Fast Inference using Context Sensitive Neural Networks

Arturo D. Lopez-Rojas[†], Karinne Ramirez-Amaro[†] and Emmanuel Dean[†]

[†] Division of Systems and Control, Chalmers University of Technology, Sweden

## Context Sensitive Learning Framework.




## Loss function.

$$\mathcal{L}_{mse}(\mathbf{x}^*, \mathbf{d}) = \frac{1}{m}\sum_{i=1}^{m}\left(\Phi(\mathbf{z}_k^{(i)}) - \mathbf{x}^{*(i)}\right)^2$$

where

$$\mathbf{z}_k^{(i)} = \begin{bmatrix}\mathbf{d}^{(i)} \\ \hat{c}_k^{(i)}\end{bmatrix}$$

and

$$\hat{c}_k^{(i)} = M(\mathbf{x}^{*(i)}, \mathbf{d}^{(i)})$$

for each $i$ sample in the dataset

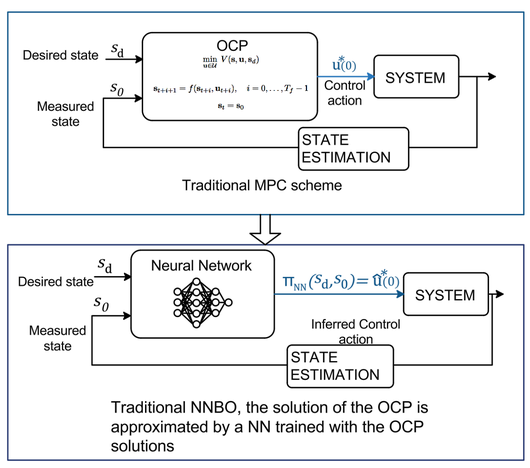
## Background

Neural Networks (NNs) can approximate optimization solvers in Model Predictive Control (MPC), enabling real-time optimization through Neural Network-Based Optimization (NNBO), offering reduced execution time, adaptability through fine-tuning, and portability across computational platforms [1]. However, existing approaches are mainly suited to convex or quadratic problems. For non-convex problems, NNs can produce infeasible or suboptimal solutions, especially when **multiple valid solutions exist for the same input**.

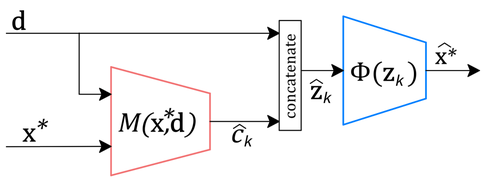
## Objectives

- Extend NN-based approximation of optimization problem solutions to non-convex problems by learning multiple solutions when present in the dataset.
- Reduce the data acquisition requirements for training NNs to approximate optimization problem solutions.
- Enable the exploration of multiple solutions while providing interpretability of the NN outputs.

## Traditional NNBO framework.




## Teacher Network.




## General Optimization problem.

$$\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}, \mathbf{d})$$

subject to

$$g_i(\mathbf{x}, \mathbf{d}) \leq 0 : \ i = 1, \ldots, m_{\text{ineq}}$$

$$h_j(\mathbf{x}, \mathbf{d}) = 0 : \ j = 1, \ldots, m_{\text{eq}}$$

## Training algorithm.

**Algorithm 1** Training stage.

1: Collect $m$ pairs of sample data $p^{(m)} = \{\mathbf{d}^{(m)}, \mathbf{x}^{*(m)}\}$.
2: $\hat{K} \leftarrow 0$.
3: Define a regression oriented loss function $\mathcal{L}(\mathbf{x}^*, \mathbf{d})$.
4: Define an acceptable $\epsilon$
5: **do**
6: $\quad \hat{K} \leftarrow \hat{K} + 1$
7: $\quad$ Build Neural Networks $\Phi$ and $M$.
8: $\quad$ Train Teacher Network as a regression task using $\mathcal{L}(\mathbf{x}^*, \mathbf{d})$
9: **while** $\epsilon_d < \mathcal{L}(\mathbf{x}^*, \mathbf{D})$
10: Consider Neural Network $\Phi(\mathbf{d}, c_k) \approx S_k(\mathbf{d})$.

## Execution algorithm.

**Algorithm 2** Mode-Aware NNBO

1: **Input:** problem parameters $\mathbf{d}$, learned network $\Phi(\cdot)$, number of modes $\hat{K}$
2: **Output:** selected solution $\mathbf{x}^*_{\text{sel}}$
3: **for** $k = 1, \ldots, \hat{K}$ **do**
4: $\quad$ Construct mode vector $c_k$ (one-hot encoding)
5: $\quad$ Compute candidate solution: $\hat{\mathbf{x}}^*_k = \Phi(\mathbf{d}, c_k)$
6: **end for**
7: Define feasible index set:

$$\mathcal{K}_{\text{feas}} = \left\{k \in \{1, \ldots, \hat{K}\} \;\middle|\; \hat{\mathbf{x}}^*_k \in \mathcal{X}\right\}$$

8: **for** $k \in \mathcal{K}_{\text{feas}}$ **do**
9: $\quad$ Evaluate objective: $J_k = f(\hat{\mathbf{x}}^*_k, \mathbf{d})$
10: **end for**
11: Select best candidate:

$$k^* = \arg\min_{k\in\mathcal{K}_{\text{feas}}} J_k$$

12: $\mathbf{x}^*_{\text{sel}} \leftarrow \hat{\mathbf{x}}_{k^*}$
13: Apply $\mathbf{x}^*_{\text{sel}}$ to the system

## Numerical experiment.

Generate collision-free trajectories from $\mathbf{p}_i = (x_i, y_i, z_i)$ to $\mathbf{p}_f = (x_f, y_f, z_f)$ while avoiding a spherical obstacle centered at $\mathbf{p}_o = (x_o, y_o, z_o)$. Data are generated using ACADOS [2], with the initial guess randomly selected from four pre-computed trajectories leading to a fixed point beyond the obstacle.

## The OCP $P_{3D}$ is defined as.

$$\min_{\mathbf{X}_r, \mathbf{U}} \quad \sum_{j=1}^{50}\left[(x_r(j) - x_{\text{f}})^2 + (y_r(j) - y_{\text{f}})^2 + (z_r(j) - z_{\text{f}})^2\right]$$

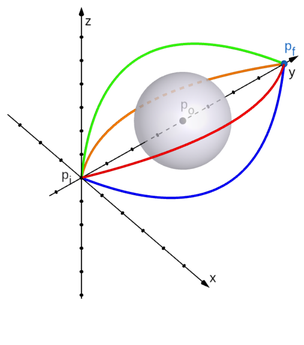
s.t.

$$\mathbf{X}_r(i+1) = \mathbf{X}_r(i) + h f_c\left(\mathbf{X}_r(i), \mathbf{U}(i)\right)$$

,

$$g(i) \leq 0$$
$$-\nabla_{\mathbf{p}} g(i)^T \mathbf{v}_r(i) - \alpha g(i) \leq 0$$
$$|u_q(i)| - u_{\max} \leq 0, \qquad q \in \{x, y, z\},$$
$$|z_r(i)| - z_{\max} \leq 0$$

## Graphic representation of possible multiple solutions.




## Numerical results. where =1 is the traditional NNBO.

**TABLE I. Time and precision performance of $P_{3D}$ approximation**

| Statistic | ACADOS | $\hat{K}=1$ | $\hat{K}=2$ | $\hat{K}=3$ | $\hat{K}=4$ |
|---|---|---|---|---|---|
| MAE (cm) | – | 5.60 | 2.90 | 1.79 | **0.67** |
| Mean time (ms) | 12.96 | 0.34 | 0.64 | 1.03 | **1.41** |
| Minimum time (ms) | 1.59 | 0.25 | 0.50 | 0.82 | **1.10** |
| Maximum time (ms) | 692.42 | 2.94 | 5.00 | 5.37 | **5.59** |
| Std. deviation (ms) | 12.75 | 0.18 | 0.28 | 0.39 | **0.53** |
| Feasible solutions (%) | 100 | 94.64 | 100 | 100 | **100** |

## References

[1] Lopez-Rojas, A.D. and Cruz-Villar, C.A. (2024). Neural networks as an approximator for a family of optimization algorithm solutions for online applications. *Neural Computing and Applications.*, 36(6), 3125–3140.
[2] Verschueren, R., Frison, G., Kouzoupis, D., Frey, J., Duijkeren, N. V., Zanelli, A., ... & Diehl, M. (2022). acados—a modular open-source framework for fast embedded optimal control. Mathematical Programming Computation, 14(1), 147-183.

CHALMERS UNIVERSITY OF TECHNOLOGY