Neural system identification and control for Formula Student Driverless Cars

Final presentation

Tudor Oancea

EPFL Racing Team

June 2023

Goals:

- Produce a system model that is more precise than the first-principles models currently used by the EPFL RT Driverless team to create more lightweight and realistic (MiL) tests.
- Devise a control scheme that approximated well the existing MPC scheme while having a lighter computational and memory footprint.

Introduction

Goals:

- Produce a system model that is more precise than the first-principles models currently used by the EPFL RT Driverless team to create more lightweight and realistic (MiL) tests.
- Devise a control scheme that approximated well the existing MPC scheme while having a lighter computational and memory footprint.

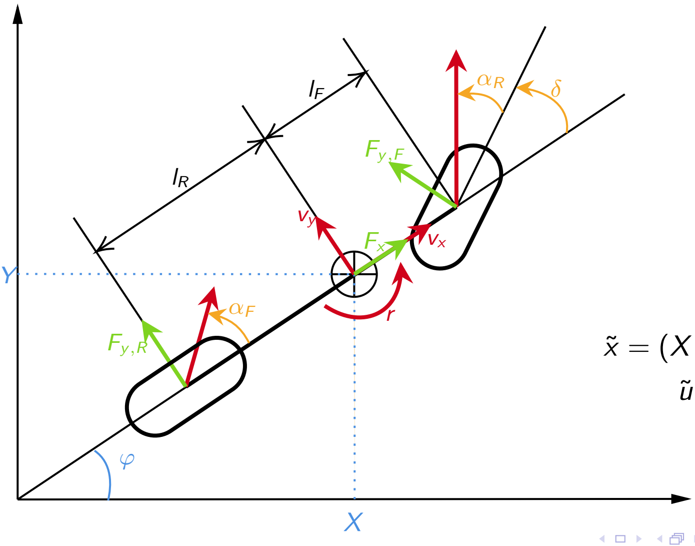
Introduction

Figure: Our test platform: the Formula Student Driverless Simulator (FSDS)

Outline

1. System identification
   - Bicycle model
   - Neural Ordinary Differential Equations
   - Dataset creation
   - Numerical experiments
2. Control
   - Original NMPC controller
   - Differentiable Predictive Control
   - Dataset creation
   - Numerical experiments
3. Conclusion & Outlook

Bicycle model - Notations



$$\tilde{x} = (X, Y, \varphi, v_x, v_y, r)$$
$$\tilde{u} = (T, \delta)$$

Bicycle model - Equations

Dynamic bicycle model (Dyn6)

$$
\begin{aligned}
\dot{X} &= v_x \cos(\varphi) - v_y \sin(\varphi) \\
\dot{Y} &= v_x \sin(\varphi) + v_y \cos(\varphi) \\
\dot{\varphi} &= r \\
\dot{v}_x &= \frac{1}{m}(F_x - F_{y,F} \sin(\delta) + m v_y r) \\
\dot{v}_y &= \frac{1}{m}(F_{y,R} + F_{y,F} \cos(\delta) - m v_x r) \\
\dot{r} &= \frac{1}{I_z}(F_{y,F} l_F \cos(\delta) \; - F_{y,R} l_R) \\
\text{with } F_x &= (C_{m1} - C_{m2} v_x) T - C_{r0} \tanh(C_{r3} v_x) - C_{r2} v_x^2 \\
F_{y,j} &= D_j \sin(C_j \arctan(B_j \alpha_j)),\ j \in \{R, F\}
\end{aligned}
$$

Bicycle model - Equations

Kinematic bicycle model (Kin4)

$$\dot{X} = v\cos(\varphi + \beta)$$
$$\dot{Y} = v\sin(\varphi + \beta)$$
$$\dot{\varphi} = \frac{v}{l_R}\sin(\beta)$$
$$\dot{v} = \frac{F_x}{m}\cos(\beta)$$
$$\text{with } \beta = \arctan\left(\frac{l_R}{l_R + l_F}\tan(\delta)\right)$$

Neural Ordinary Differential Equations - Architecture

Graybox NODE model (NeuralDyn6)

$$\dot{\tilde{x}} = \begin{pmatrix} v_x \cos(\varphi) - v_y \sin(\varphi) \\ v_x \sin(\varphi) + v_y \cos(\varphi) \\ r \\ \nu_\theta(v_x, v_y, r, T, \delta) \end{pmatrix}$$

$$\text{FFNN} \quad \nu_\theta : \begin{array}{ccc} \mathbb{R}^5 & \to & \mathbb{R}^3 \\ (v_x, v_y, r, T, \delta) & \mapsto & (\dot{v}_x, \dot{v}_y \dot{r}) \end{array}$$

RK4 discretization: $\tilde{x}^+ = \tilde{f}_1(\tilde{x}, \tilde{u}) \implies \tilde{x}_{1:N_f} = \tilde{f}_{N_f}(\tilde{x}_0, \tilde{u}_{0:N_f-1})$

$a_{i:j} := (a_i, a_{i+1}, \dots a_j)$

Neural Ordinary Differential Equations - Architecture

Graybox NODE model (NeuralDyn6)

$$\dot{\tilde{x}} = \begin{pmatrix} v_x \cos(\varphi) - v_y \sin(\varphi) \\ v_x \sin(\varphi) + v_y \cos(\varphi) \\ r \\ \nu_\theta(v_x, v_y, r, T, \delta) \end{pmatrix}$$

$$\text{FFNN} \quad \nu_\theta : \begin{array}{ccc} \mathbb{R}^5 & \to & \mathbb{R}^3 \\ (v_x, v_y, r, T, \delta) & \mapsto & (\dot{v}_x, \dot{v}_y \dot{r}) \end{array}$$

RK4 discretization: $\tilde{x}^+ = \tilde{f}_1(\tilde{x}, \tilde{u}) \implies \tilde{x}_{1:N_f} = \tilde{f}_{N_f}(\tilde{x}_0, \tilde{u}_{0:N_f-1})$

$a_{i:j} := (a_i, a_{i+1}, \ldots a_j)$

Neural Ordinary Differential Equations - Problem

Sysid dataset

$$\widetilde{\mathcal{D}} = \left\{\left(\tilde{x}_0^i, \tilde{u}_{0:N_f-1}^i, \tilde{x}_{1:N_f}^i\right)\right\}_{i=1,\dots,I}$$

Sysid problem

$$\begin{aligned} \min_{\theta} \quad & \frac{1}{IN_f} \sum_{i=1}^{I} \sum_{k=1}^{N_f} \|\tilde{x}_k^i - \tilde{x}_k^{\mathrm{pred},i}\|_P^2 \\ \text{s.t.} \quad & \tilde{x}_{1:N_f}^{\mathrm{pred},i} = \tilde{f}_{N_f}(\tilde{x}_0^i, \tilde{u}_{0:N_f-1}^i),\ i = 1, \dots, I \end{aligned}$$

Dataset creation

- Time-series collected at 20Hz in FSDS with a video-game controller $\implies$ transcription in the form $\tilde{x}_0^i, \tilde{u}_{0:N_f-1}^i, \tilde{x}_{1:N_f}^i$ with a sliding window
- Crowdsourced data collection during the EPFL open days $\implies$ high volume of data from a wide range of operating scenarios.
- Pre-processing: yaw signal φ made continuous by appropriately translating each value by multiple of 2π .

Dataset creation

- Time-series collected at 20Hz in FSDS with a video-game controller $\implies$ transcription in the form $\tilde{x}_0^i, \tilde{u}_{0:N_f-1}^i, \tilde{x}_{1:N_f}^i$ with a sliding window
- Crowdsourced data collection during the EPFL open days $\implies$ high volume of data from a wide range of operating scenarios.
- Pre-processing: yaw signal φ made continuous by appropriately translating each value by multiple of 2π .

Dataset creation

- Time-series collected at 20Hz in FSDS with a video-game controller $\implies$ transcription in the form $\tilde{x}_0^i, \tilde{u}_{0:N_f-1}^i, \tilde{x}_{1:N_f}^i$ with a sliding window
- Crowdsourced data collection during the EPFL open days $\implies$ high volume of data from a wide range of operating scenarios.
- Pre-processing: yaw signal φ made continuous by appropriately translating each value by multiple of 2π .

Numerical experiments - Training

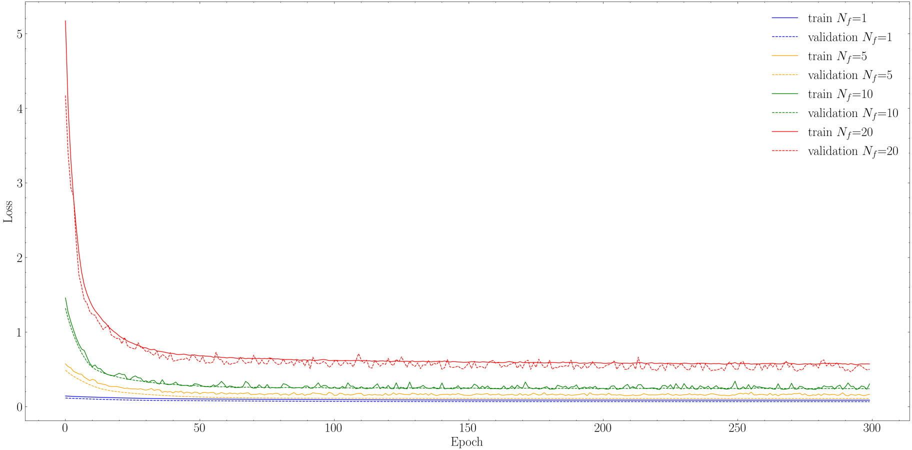


Figure: Training and validation losses in system identification training for $N_f \in \{1, 5, 10, 20\}$

Numerical experiments - Average errors

	Kin4		NeuralDyn6 with Nf=1		NeuralDyn6 with Nf=5		NeuralDyn6 with Nf=10		NeuralDyn6 with Nf=20	
	mean	std	mean	std	mean	std	mean	std	mean	std
XY	7.3967	7.8335	0.6877	1.7896	0.6205	1.6938	0.5944	1.5599	0.5718	0.7547
φ	1.1875	1.2385	0.0713	0.1840	0.0571	0.1702	0.0544	0.1532	0.0496	0.1405
v_x	0.4330	0.6145	0.1844	0.3425	0.1696	0.3263	0.1632	0.3048	0.1835	0.2965
v_y	N/A	N/A	0.0718	0.0944	0.0714	0.0932	0.0756	0.0941	0.0845	0.1059
r	N/A	N/A	0.2873	0.7396	0.2646	0.7384	0.2619	0.7375	0.2666	0.7547

Table: L^2 errors distributions of Kin4 and NeuralDyn6 by variable

Variable	XY	φ	v_x
Improvement	91.96%	95.42%	62.31%

Table: Improvements in mean error of NeuralDyn6 with $N_f = 10$ over Kin4

Numerical experiments - Average errors

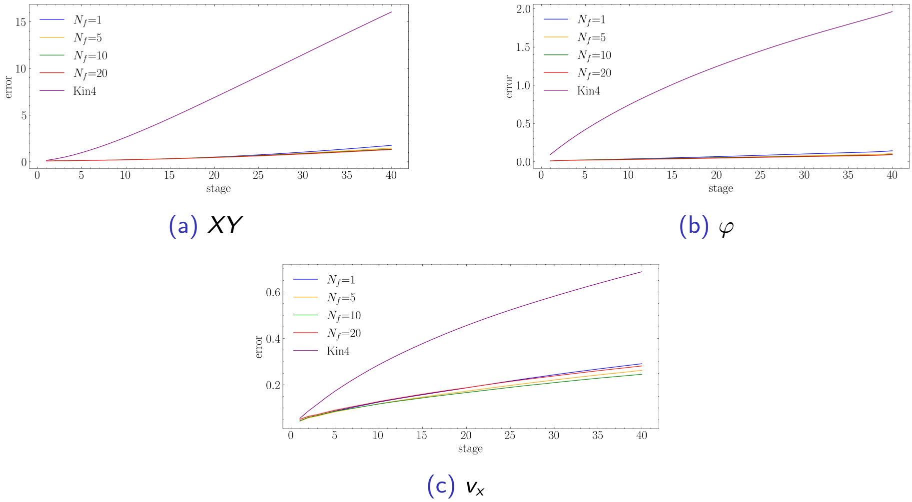


Figure: Average L^2 errors of Kin4 and NeuralDyn6 by stage

Numerical experiments - Open loop trajectories

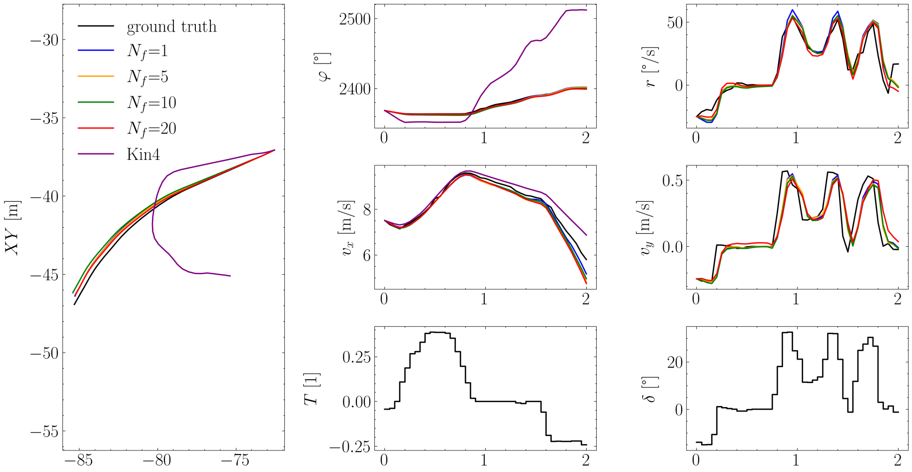


Figure: Open loop trajectories of Kin4 and NODE models

Original NMPC controller

Lifted state and dynamics

$$x_k = (X_k, Y_k, \varphi_k, v_{x,k}, v_{y,k}, r_k, \delta_{k-1}), u_k = (T_k, d\delta_k)$$

$$\implies x_{k+1} = f_1(x_k, u_k) = \begin{pmatrix} \tilde{f}_1(X_k, Y_k, \varphi_k, v_{x,k}, v_{y,k}, r_k, T_k, \delta_{k-1} + d\delta_k) \\ \delta_{k-1} + d\delta_k \end{pmatrix}$$

Original NMPC controller

Lifted state and dynamics

$$x_{k+1} = f_1(x_k, u_k)$$

NMPC

$$\begin{aligned}
\min_{x,u} \quad & \sum_{k=0}^{N_f-1} \underbrace{\|x_k - x_k^{\text{ref}}\|_Q^2 + \|u_k\|_R^2}_{=:l(x_k,u_k)} + \underbrace{\|x_{N_f} - x_{N_f}^{\text{ref}}\|_{Q_{N_f}}^2}_{=:F(x_{N_f})} \\
\text{s.t.} \quad & x_0 = \hat{x}_0 \\
& x_{k+1} = f_1(x_k, u_k),\ k = 0, \dots, N_f - 1 \\
& \underline{x} \leq x_k \leq \bar{x}, k = 0, \dots, N_f \\
& \underline{u} \leq u_k \leq \bar{u}, k = 0, \dots, N_f - 1
\end{aligned}$$

Differentiable Predictive Control

Control dataset

$$\mathcal{D} = \left\{ (x_0^j, x_{0:N_f}^{\mathrm{ref},j}) \right\}_{j=1,\dots,J}$$

DPC

$$\begin{aligned}
\min_{\vartheta} \quad & \frac{1}{JN_f} \sum_{j=1}^{J} \sum_{k=1}^{N_f} l(x_k^j, u_k^j) + p_{S_x}(x_k^j, \underline{x}, \bar{x}) + p_{S_u}(u_k^j, \underline{u}, \bar{u}) \\
& \qquad + \; F(x_{N_f}^j) + p_{S_x}(x_{N_f}^j, \underline{x}, \bar{x}) \\
\text{s.t.} \quad & \left.\begin{aligned} u_{0:N_f-1}^j &= \pi_\vartheta(x_0^j, x_{0:N_f}^{\mathrm{ref},j}) \\ x_{1:N_f}^j &= f_{N_f}(x_0^j, u_{0:N_f-1}^j) \end{aligned}\right\} j = 1, \dots, J
\end{aligned}$$

Dataset creation

- Time-series collected at 20Hz in FSDS while running NMPC $\implies$ transcription in the form $x_0^j, x_{0:N_f}^{\mathrm{ref},j}$ with a sliding window
- Pre-processing: yaw signal $\varphi, \varphi^{\mathrm{ref}}$ made continuous by appropriately translating each value by multiple of 2π .

Dataset creation

- Time-series collected at 20Hz in FSDS while running NMPC $\implies$ transcription in the form $x_0^j, x_{0:N_f}^{\mathrm{ref},j}$ with a sliding window
- Pre-processing: yaw signal $\varphi, \varphi^{\mathrm{ref}}$ made continuous by appropriately translating each value by multiple of 2π .

Numerical experiments - Constraint violation losses

Constraint	δ lower bound	δ upper bound	v_x lower bound	v_x upper bound
Violation	1.03e-5	2.14e-5	0	0

Table: Mean constraint violation committed by DPC on the validation set

Numerical experiments - Open loop behavior

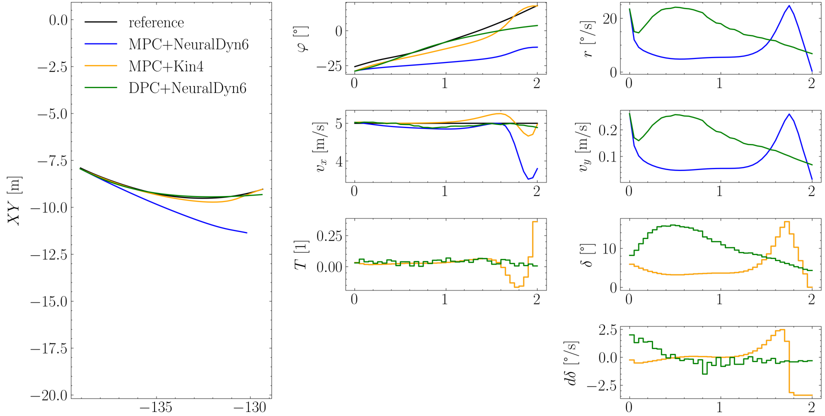


Figure: Open-loop predictions of different pairs of controllers and system models.
Remark: in the subplots T, δ and $d\delta$, the blue and yellow curves are overlapping.

Numerical experiments - Closed loop behavior

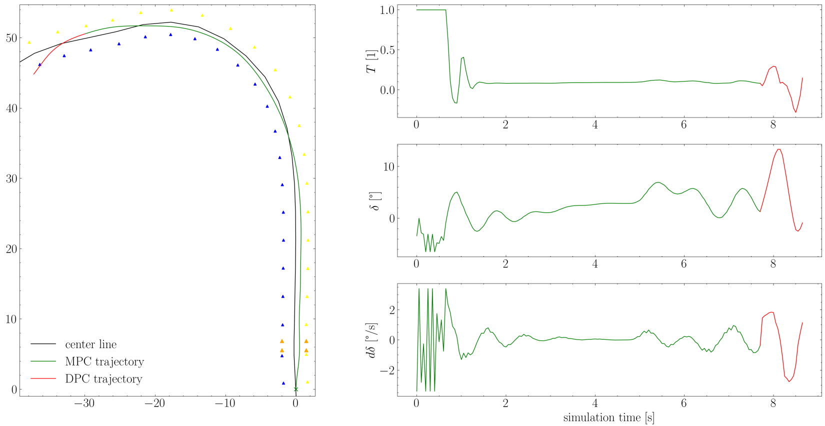


Figure: Closed-loop trajectories of MPC followed by DPC in FSDS

Conclusion & Outlook

- Significant progress in the system identification, as demonstrated by the improved robustness and accuracy of NeuralDyn6 compared to Kin4.
- However, in the realm of control, we encountered challenges with the deployment of DPC in closed-loop scenarios and not enough robustness yet.
- Further improvement strategies:
  - Naive data augmentation
  - Combination of DPC with imitation learning methods such as DAgger. $\implies$ use (predictive) safety filters for safe deployment and training on the real car

Conclusion & Outlook

- Significant progress in the system identification, as demonstrated by the improved robustness and accuracy of NeuralDyn6 compared to Kin4.
- However, in the realm of control, we encountered challenges with the deployment of DPC in closed-loop scenarios and not enough robustness yet.
- Further improvement strategies:
  - Naive data augmentation
  - Combination of DPC with imitation learning methods such as DAgger. $\implies$ use (predictive) safety filters for safe deployment and training on the real car

Conclusion & Outlook

- Significant progress in the system identification, as demonstrated by the improved robustness and accuracy of NeuralDyn6 compared to Kin4.
- However, in the realm of control, we encountered challenges with the deployment of DPC in closed-loop scenarios and not enough robustness yet.
- Further improvement strategies:
  - Naive data augmentation
  - Combination of DPC with imitation learning methods such as DAgger. $\Longrightarrow$ use (predictive) safety filters for safe deployment and training on the real car

Conclusion & Outlook

- Significant progress in the system identification, as demonstrated by the improved robustness and accuracy of NeuralDyn6 compared to Kin4.
- However, in the realm of control, we encountered challenges with the deployment of DPC in closed-loop scenarios and not enough robustness yet.
- Further improvement strategies:
  - Naive data augmentation
  - Combination of DPC with imitation learning methods such as DAgger. $\Longrightarrow$ use (predictive) safety filters for safe deployment and training on the real car