Behavioral System Theory in Safe Predictive Control

Aybüke Ulusarslan
Technische Universität München
Email: aybuke.ulusarslan@tum.de

***Abstract*—We consider the problem of safe control in the absence of explicit models. To that end, we evaluate the Data-Driven Safety Filter, a novel algorithm rooted in Behavioral System Theory, which ensures system safety by modifying unsafe inputs using only input-output data from a single system trajectory. To benchmark its performance, we compare the Data-Driven Safety Filter with other control methodologies, including the Data-Enabled Predictive Control and Model Predictive Control algorithms.**

I. Introduction

Autonomous systems are increasingly being integrated into diverse applications, necessitating robust and reliable control methods to ensure safe operation. Many of the traditional control methods are rooted in the state-space framework and requiring explicit system modeling - a requirement that is often impractical for complex and uncertain problems. In contrast, learning-based methods, such as Reinforcement Learning (RL), can handle such complex and uncertain tasks by leveraging data generated from the agent-environment interaction. However, these methods lack formal safety guarantees, posing significant risks in real-world applications.
To bridge this gap, modular safety filters have been developed to operate alongside learning agents, enforcing predefined safety constraints. Most state-of-the-art safety filters have been developed within the state-space framework, requiring explicit system representations. This reliance contradicts the purpose of employing learning-based controllers and limits their applicability to real-life scenarios involving uncertainty and partial knowledge of the environment.
This study investigates the application of purely data-driven optimization-based control algorithms, with an emphasis on a Data-Driven Safety Filter (DDSF) [1]. The DDSF operates within a so-called input-output framework, relying only on a pre-recorded system trajectory to represent the underlying dynamical system. This approach is rooted in Behavioral Systems Theory (BST), leveraging theoretical guarantees established under specific assumptions, as described in the seminal work by Willems et al. (2005) [2].

For benchmarking, we compare DDSF with other predictive control strategies, such as Data-Enabled Predictive Control (DeePC) and Model Predictive Control (MPC). Through MATLAB implementations, we evaluate these algorithms on various linear systems.

II. Preliminaries and Problem Statement

Consider a linear time-invariant (LTI) system that evolves according to the following equations:

$$x(t+1) = Ax(t) + Bu(t), \tag{1a}$$

$$y(t) = Cx(t) + Du(t), \tag{1b}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$, $D \in \mathbb{R}^{p\times m}$, and $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$, $y(t) \in \mathbb{R}^p$ are respectively the state, control input, and output of the system at time $t \in \mathbb{Z}_{\geq 0}$.
The behavior of the system being described wrt. (A, B, C, D) (i.e., $\mathscr{B}_{ss}(A, B, C, D) := \{w := \mathrm{col}(u, y) \in (\mathbb{R}^{m+p})^{\mathbb{N}} \mid \exists x \in (\mathbb{R}^n)^{\mathbb{N}} \quad s.t. \quad \sigma x = Ax + Bu, y = Cx + Du\}$) is what we refer to as an explicit system representation.

The system from (eq. (1)) can be further subject to constraints wrt. inputs $u \in \mathbb{R}^m$, outputs $y \in \mathbb{R}^p$ and system states $x \in \mathbb{R}^n$. These (safety) constraints can be expressed as linear inequalities as follows:

$$\mathcal{U} := \{u \in \mathbb{R}^m \mid A_u u < b_u, A_u \in \mathbb{R}^{n_u\times m}, b_u \in \mathbb{R}^{n_u}\}, \tag{2a}$$

$$\mathcal{X} := \{x \in \mathbb{R}^n \mid A_x x < b_x, A_x \in \mathbb{R}^{n_x\times n}, b_x \in \mathbb{R}^{n_x}\}, \tag{2b}$$

$$\mathcal{Y} := \{y \in \mathbb{R}^p \mid A_y y < b_y, A_y \in \mathbb{R}^{n_p\times p}, b_y \in \mathbb{R}^{n_p}\}, \tag{2c}$$

,where n_u, n_x, n_y correspond to the input, state, and output constraints respectively.
The control objectives that we are primarily concerned with in this study are

a) Constraint satisfaction: Find control sequence $u \in (\mathbb{R}^m)^T$ of length T s.t. $\forall t \in [T] : \exists x(t) \in \mathcal{X},\ y(t) \in \mathcal{Y},\ u(t) \in \mathcal{U}$ and $\sigma x = Ax + Bu, y = Cx + Du$.
b) Optimal tracking:

$$u^* := \underset{u}{\operatorname{argmin}} \sum_{t\in[T]} \left(\|y(t) - r(t)\|_Q^2 + \|u(t)\|_R^2\right) \tag{3a}$$

$$s.t \quad x_{t+1} = Ax_t + Bu_t \tag{3b}$$

$$y_t = Cx_t + Du_t \tag{3c}$$

,where $r(t) \in \mathbb{R}^p$ is the desired output trajectory, $Q \succeq 0$ and $R \succeq 0$ are weighting matrices that encode the relative importance of minimizing tracking error and control effort, respectively.

A. Behavioral Systems Theory and Non-Parametric System Representation

In behavioral systems theory, a dynamical system is defined as a a 3-tuple $\Sigma = (\mathbb{T}, \mathbb{W}, \mathscr{B})$, where $\mathbb{T}$ is the time axis (often $\mathbb{T} \subseteq \mathbb{R}$ for the continuous, and $\mathbb{T} \subseteq \mathbb{Z}_{\geq 0}$ for the discrete case), $\mathbb{W}$ is the signal space (e.g., $\mathbb{W} = \mathbb{R}^w$ for real-valued w-dimensional signals), and $\mathscr{B} \subseteq \mathbb{W}^{\mathbb{T}}$ is the behavior. The behavior $\mathscr{B}$ comprises all permissible trajectories $w : \mathbb{T} \mapsto \mathbb{W}$ of the system. If the time axis and the signal space are known, as is often the case, the system $\Sigma = (\mathbb{T}, \mathbb{W}, \mathscr{B})$ can be identified with its behavior $\mathscr{B}$.

The behavior $\mathscr{B}$ of a system can be sufficiently represented through Hankel matrices, which are constructed from persistently exciting (Def. 2) input signals by sequentially applying a one-step shift-operator and stacking the resulting vectors row-wise as follows:

Definition 1. ***(Hankel matrix)*** *Given a trajectory $w^d := (w_1, w_2, \dots, w_T)$, $w_i \in \mathbb{R}^m$ $\forall i \in [T]$ a Hankel matrix of order L can be constructed as:*

$$H_L(w^d) = \begin{bmatrix} w_1 & w_2 & \cdots & w_{T-L+1} \\ w_2 & w_3 & \cdots & wT-L+2 \\ \vdots & \vdots & \ddots & \vdots \\ w_L & w_{L+1} & \cdots & wT \end{bmatrix} \tag{4}$$

,where $H_L(w^d) \in \mathbb{R}^{mL\times(T-L+1)}$.

Definition 2. ***(Persistency of excitation)*** *Let $L, T \in \mathbb{Z}_{>0}$ such that $T \geq L$. A signal $u = \operatorname{col}(u_1, \dots, u_T) \in \mathbb{R}^{Tm}$ is said to be persistently exciting of order L if the associated Hankel matrix $H_L(u)$ has full row rank, where:*

$$H_L(u) = \begin{bmatrix} u_1 & u_2 & \cdots & u_{T-L+1} \\ u_2 & u_3 & \cdots & u_{T-L+2} \\ \vdots & \vdots & \ddots & \vdots \\ u_L & u_{L+1} & \cdots & u_T \end{bmatrix}$$

The persistency of excitation ensures that the input signal contains sufficiently rich information to span the input space.

Details on the derivation of the Hankel matrix representation of system behavior from annihilating functions are provided in the appendix for further reference.

The **Fundamental Lemma** [2] links the behavior $\mathscr{B}$ with the Hankel matrix: If the input signal $u_d := (u(1), u(2), \dots, u(T))$ (where $u(t) \in \mathbb{R}^m$ is the input vector at time step $t \in \mathbb{T}$) is persistently exciting of order $L + n$, where n is the system state dimension, then $\operatorname{colspan}(H_L(w_d)) = \mathscr{B}_{[1,L]}$ This implies that any trajectory $w \in \mathscr{B}$ over $[1, L]$ can be expressed as a linear combination of columns of $H_L(w_d)$.
In other words: Let $u_d := (u(1), u(2), \dots, u(T))$ be persistently exciting of order $L + n(\mathscr{B})$, and (u_d, y_d) a trajectory of $\mathscr{B}$. Then,

$$(\bar{u}, \bar{y}) \in \mathscr{B}_T \iff \exists \quad \alpha \in \mathbb{R}^{T-L+1} : \begin{bmatrix} H_L(u_d) \\ H_L(y_d) \end{bmatrix} \alpha = \begin{bmatrix} \bar{u} \\ \bar{y} \end{bmatrix}. \tag{5}$$

The problem statement of optimal tracking (eq. (3)) can then be expressed in behavioral representation as:

$$\min_u \sum_{t\in[T]} \left(\|y(t) - r(t)\|_Q^2 + \|u(t)\|_R^2\right) \tag{6a}$$

$$s.t. \quad \exists\alpha \in \mathbb{R}^{T-L+1} : \begin{bmatrix} \bar{u} \\ \bar{y} \end{bmatrix} = \begin{bmatrix} H_L(u_d) \\ H_L(y_d) \end{bmatrix} \alpha \tag{6b}$$

,where $\bar{u} := \left[u_1^{\mathrm{T}}, ..., u_T^{\mathrm{T}}\right]^{\mathrm{T}}$ and $\bar{y} := \left[y_1^{\mathrm{T}}, ..., y_T^{\mathrm{T}}\right]^{\mathrm{T}}$. This BST-based optimal tracking objective is central to the DeePC algorithm, which we will discuss next.

B. Data-Enabled Model-Predictive Control (DeePC)

Given a time horizon $N \in \mathbb{Z}_{>0}$, the DeePC [3] algorithm involves solving the folowing optimization problem in a receding horizon manner:

$$\min_{g,u,y} \sum_{k=0}^{N-1} \left(\|y_k - r_{t+k}\|_Q^2 + \|u_k\|_R^2\right) \quad s.t. \tag{7a}$$

$$\begin{pmatrix} U_p \\ Y_p \\ U_f \\ Y_f \end{pmatrix} g = \begin{pmatrix} u_{\text{ini}} \\ y_{\text{ini}} \\ u \\ y \end{pmatrix}, \tag{7b}$$

$$u_k \in \mathcal{U}, y_k \in \mathcal{Y} \quad \forall k \in \ [0, N-1] \tag{7c}$$

,where $r = (r_0, r_1, \dots) \in (\mathbb{R}^p)^{\mathbb{Z}_{\geq 0}}$ is a reference trajectory, $\mathcal{U} \subseteq \mathbb{R}^m$ the input constraint set, $\mathcal{Y} \subseteq \mathbb{R}^p$ the output constraint set, $Q \in \mathbb{R}^{p\times p}$ is the output cost matrix, and $R \in \mathbb{R}^{m\times m}$ is the control cost matrix.
The trajectory $\operatorname{col}(u_{ini}, y_{ini}) \in \mathscr{B}_{T_{ini}}$ is a $T_{ini} \geq l(\mathscr{B})$ long sequence that is used to fix the underlying initial state x_{ini}, effectively replacing the initial condition $x(0) = x_0$ in a state-space representation. The matrices U_p, U_f, Y_p, Y_f encode system behavior and are constructed in the following manner:

1) A sequence of T inputs $u^d = \operatorname{col}(u_1^d, u_2^d, \dots, u_T^d)$ and the resulting outputs $y^d = \operatorname{col}(y_1^d, y_2^d, \dots, y_T^d)$ are sampled from the underlying LTI system. Hereby required is a minimum sample length of
$$T \geq (m + 1)(T_{ini} + N + n(\mathbb{B})) - 1, \tag{8}$$
and an input sequence u^d that is persistently exciting of order
$$T_{ini} + N + n(\mathbb{B}). \tag{9}$$
2) u^d and y^d are used to construct Hankel matrices of order $T_{ini} + N$:
$$H_{T_{ini}+N}(u^d) = \begin{bmatrix} u_1^d & \cdots & u_{T-T_{\text{ini}}-N+1} \\ u_2^d & \cdots & u_{T-T_{\text{ini}}-N+2} \\ \vdots & \ddots & \vdots \\ u_{T_{\text{ini}}+N}^d & \cdots & u_T \end{bmatrix} \tag{10a}$$

$$H_{T_{ini}+N}(y^d) = \begin{bmatrix} y_1^d & \cdots & y_{T-T_{\text{ini}}-N+1} \\ y_2^d & \cdots & y_{T-T_{\text{ini}}-N+2} \\ \vdots & \ddots & \vdots \\ y_{T_{\text{ini}}+N}^d & \cdots & y_T \end{bmatrix} \tag{10b}$$

3) $H_{T_{ini}+N}(u^d)$ and $H_{T_{ini}+N}(y^d)$ are partitioned into past and future sections:

$$U_p = \begin{bmatrix} u_1^d & \cdots & u_{T-T_{\text{ini}}-N+1} \\ \vdots & \ddots & \vdots \\ u_{T_{\text{ini}}}^d & \cdots & u_{T-N} \end{bmatrix} \in \mathbb{R}^{T_{ini}m \times L}, \tag{11a}$$

$$U_f = \begin{bmatrix} u_{T_{ini}+1}^d & \cdots & u_{T-N+1} \\ \vdots & \ddots & \vdots \\ u_{T_{\text{ini}}+N}^d & \cdots & u_T \end{bmatrix} \in \mathbb{R}^{Nm \times L}, \tag{11b}$$

$$Y_p = \begin{bmatrix} y_1^d & \cdots & y_{T-T_{\text{ini}}-N+1} \\ \vdots & \ddots & \vdots \\ y_{T_{\text{ini}}}^d & \cdots & y_{T-N} \end{bmatrix} \in \mathbb{R}^{T_{ini} \times L}, \tag{11c}$$

$$Y_f = \begin{bmatrix} y_{T_{ini}+1}^d & \cdots & y_{T-N+1} \\ \vdots & \ddots & \vdots \\ y_{T_{\text{ini}}+N}^d & \cdots & y_T \end{bmatrix} \in \mathbb{R}^{Np \times L} \tag{11d}$$

,where $L := T - T_{\text{ini}} - N + 1$.

The optimization problem (eq. (7)) is solved in a receding horizon manner: At each time step, an optimal trajectory of length N is computed by first solving for the canonical variable g, and projecting U_f onto g^*: $u^* := (u_0^*, \ldots, u_N^*) = U_f g^*$. The first $s < N-1$ entries of the optimal input sequence $(u_0^*, \ldots, u_{s-1}^*)$ are applied to the system, and the inital trajectory $\text{col}(u_{ini}, y_{ini})$ updated accordingly.

Algorithm 1 DeePC [3]

```
Input: $\text{col}(u^d, y^d) \in \mathscr{B}_T$, reference trajectory $r$, past input/output data $\text{col}(u_{\text{ini}}, y_{\text{ini}}) \in \mathscr{B}_{T_{\text{ini}}}$, constraint sets $\mathcal{U}$ and $\mathcal{Y}$, performance matrices $Q$ and $R$.
1: Solve eq. (7) for $g^\star$.
2: Compute the optimal input sequence $u^\star = U_f g^\star$.
3: Apply inputs $(u(t), \ldots, u(t+s-1)) = (u_0^\star, \ldots, u_{s-1}^\star)$ for some $s \leq N-1$.
4: Set $t \leftarrow t+s$ and update $u_{\text{ini}}$ and $y_{\text{ini}}$ to the $T_{\text{ini}}$ most recent input/output measurements.
5: Return to step 1.
```

This scheme is closely related to the well-established Model Predictive Control (MPC) algorithm, with the primary distinction lying in their approaches to state estimation:

$$\min_{u,x,y} \sum_{k=0}^{N-1} \left(\|y_k - r_{t+k}\|_Q^2 + \|u_k\|_R^2 \right) s.t. \tag{12a}$$

$$x_{k+1} = Ax_k + Bu_k, \quad \forall k \in [0, N-1], \tag{12b}$$

$$y_k = Cx_k + Du_k, \quad \forall k \in [0, N-1], \tag{12c}$$

$$x_0 = \hat{x}(t), \tag{12d}$$

$$u_k \in \mathcal{U}, y_k \in \mathcal{Y} \quad \forall k \in [0, N-1] \tag{12e}$$

It has been shown that the algorithms (1) and (2) yield equivalent trajectories for linear time-invariant (LTI) systems under the previously mentioned assumptions (eq. (8)), (eq. (9)).

Algorithm 2 MPC [3]

```
Input: State-space matrices $(A, B, C, D)$, reference trajectory $r$, past input/output data $(u, y)$, constraint sets $\mathcal{U}$ and $\mathcal{Y}$, performance matrices $Q$ and $R$.
1: Generate state estimate $\hat{x}(t)$ based on $(u, y)$.
2: Solve eq. (12) for $u^* = (u_0^*, \ldots, u_{N-1}^*)$.
3: Apply inputs $(u(t), \ldots, u(t+s-1)) = (u_0^\star, \ldots, u_{s-1}^\star)$ for some $s \leq N-1$.
4: Set $t \leftarrow t+s$ and update past input/output data.
5: Return to step 1.
```

C. Data-Driven Safety Filter (DDSF)

The data-driven safety filter proposed by Bajelani et al. [1] is a constraint satisfaction mechanism rooted in behavioral systems theory. Its primary objective is to minimally modify potentially unsafe control inputs to ensure that both the input and the resulting system output remain within their respective constraint sets. This approach detects potentially unsafe trajectories using a behavioral representation (eq. (5)) rather than relying on explicit state-space equations.

Definition 3. ***(Safe set, state-space definition [1])*** *The safe set for system (eq.* (1)*) can be written as follows:*

$$\mathcal{S} := \{x_0 \in \mathcal{X} \subseteq \mathbb{R}^m \mid \exists (u_k)_{k=0}^\infty \in \mathcal{U}^\mathbb{N} \quad \forall k \geq 0 : \\ x_{k+1} = Ax_k + By_k \in \mathcal{X}, y_k = Cx_k + Du_k \in \mathcal{Y}\}$$

The objective here is to ensure the safety of output trajectories in the sense of the definition (Def. 3) without requiring access to the system's states. Accordingly, only the constraints $\mathcal{U}$ and $\mathcal{Y}$ are taken into account.

Definition 4. ***(Safety control law)*** *In the state-space framework, a safety law can be formulated as follows:*

$$\min_{u_{[0,N-1]}} \quad \|u_0(t) - u_l(t)\|_R^2 \tag{13a}$$

$$s.t. \quad x_{k+1} = Ax_k + Bu_k, \tag{13b}$$

$$x_0(t) = x_0, \tag{13c}$$

$$x_k(t) \in \mathcal{X} \quad \forall k \in [0, N] \tag{13d}$$

$$u_k(t) \in \mathcal{U} \quad \forall k \in [0, N-1]. \tag{13e}$$

with $N \to \infty$.

Since this infinite-horizon optimization problem (Def. 4) is infeasible in practice, it is necessary to employ an approximation method. This is typically achieved by either selecting a sufficiently large horizon or using a shorter horizon combined with a terminal safety constraint X_T [4]. Bajelani et.al. [1] adopt the latter approach, where safe trajectories of length N

(prediction horizon), leading from the initial condition to the terminal safe set, are computed in a receding horizon manner.

Definition 5. ***(Backup trajectory)*** *In the context of safety filters and the safety control law (Def. 4) stated above, a backup trajectory is a finite-length trajectory $w_{[0,N-1]} \in \mathscr{B}_N$ that starts from the initial state $x_0(t) = x_0$ and leads to a terminal, control-invariant set X_T, while satisfying all admissibility constraints throughout its duration:*

$$\begin{aligned}Traj_b \coloneqq \{(u_k, x_k) \in \mathscr{B}_N \mid \forall k \in [0, N-1], t \in \mathbb{T} :\\ (u_k(t), x_k(t)) \in \mathcal{U} \times \mathcal{X},\\ x_0(t) = x_0, x_{N-1}(t) \in X_t\}\end{aligned}$$

Definition 6. ***(Input-Output Safe Set and Safe Initial Trajectories [1])*** *Let $T_{ini} \coloneqq \{(u_k, y_k)\}_{k=0}^{T_{ini}-1}$ denote the initial trajectory and $T_b \coloneqq \{(u_k, y_k)\}_{k=T_{ini}}^{\infty}$ the infinite-length backup trajectory for the system at $k = T_{ini}$. Both T_ini and T_b are considered safe if they belong to $\mathcal{U} \times \mathcal{X}$ and can be combined seamlessly into a permissible trajectory of the system. The input-output safe set $\mathcal{S} \subseteq \mathcal{U} \times \mathcal{X}$ is the collection of all such initial trajectories T_{ini}.*

Since computing an infinite-length backup trajectory is infeasible, the framework employs a control-invariant safe set $\mathcal{S}_f$ to truncate the trajectory's tail. The resulting trajectory spans N steps (the prediction horizon), and ends in a control-invariant and safe state. This control-invariant terminal safe set can be initialized with equilibrium points of the system.

Definition 7. ***(Equilibrium point, input-output framework)*** *If the sequence $\{(u_k, y_k)\}_{k=0}^{T_{ini}-1}$ satisfies $(u_k, y_k) = (u^s, y^s)$ for all $k \in [0, N-1]$ and $T_{ini} \geq l(A, C)$, then (u^s, y^s) is an equilibrium point of the system.*

Since the pairs $(u, y) \in \mathcal{S}_f$ are control-invariant and remain constant over time, an infinite-length trajectory can be fully represented by a trajectory of length T_{ini}, as $T_{ini} \geq l(A, C)$ steps are sufficient to uniquely determine the system's state. Consequently, the infinite tail-end of the backup trajectory in problem (Def. 4) can be substituted with a T_{ini}-step trajectory $Traj_s \coloneqq \{(u_k, y_k)\}_{k=N}^{N+T_{ini}-1}$ within $\mathcal{S}_f$, resulting in a finite-horizon approximation of the original problem. Once the initial state is fixed using an initial trajectory $Traj_{ini} \coloneqq \{(u_k, y_k)\}_{k=-T_{ini}}^{k=-1}$, the data-driven safety control law objective reduces to computing an intermediate (backup) trajectory that seamlessly connects $Traj_{ini} \in \mathscr{B}_{T_{ini}}$ and $Traj_s \in \mathcal{S}_f \subset \mathscr{B}_{T_{ini}}$.
The computation of such a backup trajectory $Traj_P \coloneqq \{(\bar{u}_k, \bar{y}_k)\}_{k=0}^{N-1}$ can be formulated as an optimization problem:

$$\min_{\alpha(t), \bar{u}(t), \bar{y}(t)} \|\bar{u}_0(t) - u_l(t)\|_R^2 \quad s.t. \tag{14a}$$

$$\begin{bmatrix} \bar{u}_{[-T_{\text{ini}}, N+T_{\text{ini}}-1]}(t) \\ \bar{y}_{[-T_{\text{ini}}, N+T_{\text{ini}}-1]}(t) \end{bmatrix} = \begin{bmatrix} H_L(u^d) \\ H_L(y^d) \end{bmatrix} \alpha(t), \tag{14b}$$

$$\begin{bmatrix} \bar{u}_{[-T_{\text{ini}}, -1]}(t) \\ \bar{y}_{[-T_{\text{ini}}, -1]}(t) \end{bmatrix} = \begin{bmatrix} u_{[t-T_{\text{ini}}, t-1]} \\ y_{[t-T_{\text{ini}}, t-1]} \end{bmatrix}, \tag{14c}$$

$$\bar{u}_k(t) \in \mathcal{U}, \bar{y}_k(t) \in \mathcal{Y} \quad \forall k \in [0, N-1], \tag{14d}$$

$$\bar{u}_k(t), \bar{y}_k(t) \in S_f, \quad \forall k \in [N, N+T_{ini}-1]. \tag{14e}$$

,where $H_L(u^d)$ and $H_L(y^d)$ are Hankel matrices constructed from the control input and output vectors obtained during T-step offline data collection, u_l corresponds to a so-called learning input generated by an external entity such as a Reinforcement Learning agent, R is a positive definite cost matrix, and $\alpha \in$ is a canonical variable that linearly combines columns of the stacked Hankel matrices into admissible trajectories of the system. Hereby, it is assumed that the stacked Hankel matrix from (17b) is persistently exciting of order $L = N + 2T_{ini}$ and that the prediction horizon satisfies $N \geq l(A, C)$.
This optimization problem is solved in each time step t in a receding-horizon manner:

Algorithm 3 DDSF[1]

```
Input: $u^d, y^d, T_{ini}, N, \mathcal{S}_f, T_{sim}$
1: $t \leftarrow 0$
2: while $t < T_{sim}$ do
3:     Compute a backup trajectory $u* \coloneqq [u_0^*, \dots u_{N-1}^*]$ by
       solving the optimization problem (eq. (14))
4:     $u(t) \leftarrow u_0^*, y(t) \leftarrow y_0^*$
5:     Update $Traj_{ini}$
6:     $t \leftarrow t + 1$
7: end while
```

In an application scenario, the stopping condition of the receding horizon loop can be a criterion such as the learning agent's convergence to a policy. Here we used $t \geq T_{sim}$ to better reflect our own experimental setup, where the safety filter operates over a fixed number of simulation steps.

III. Experiments

A. General Setup

The experiments conducted in this project were implemented in MATLAB, with the code organized into several core functionalities, which will be outlined in this section.

1) Data Collection: The dynamical systems of interest are represented through state-space matrices and simulated over T steps to obtain an input/output trajectory (u^d, y^d), where $u^d \coloneqq \left[(u_1^d)^\mathrm{T}, \dots, (u_T^d)^\mathrm{T}\right] \in \mathbb{R}^{mT}$ and $y^d \coloneqq \left[(y_1^d)^\mathrm{T}, \dots, (y_T^d)^\mathrm{T}\right] \in \mathbb{R}^{pT}$. Hereby, it is important to point out that the usage of the state-space representation replaces real-life data collection from the system.
The input vectors u^d used in the simulations are obtained by generating random vectors of appropriate shape. In case of the DDSF we generate random vectors within the safe set $\mathcal{U}$, and multiply them with a scaling factor > 1 to obtain partially unsafe input signals.
We have implemented an optional warm-starting mechanism to align the initial state assumed in offline data collection with the online initial state. This adjustment accommodates potential local variations in system behavior, thereby

enhancing the robustness of trajectory predictions. Hereby we make certain assumptions as stated below.
Assumption 1: $N > T_{ini} \geq l(A, C)$, where N is the prediction horizon, T_{ini} the length of the initial trajectory, and l the system's lag.
Assumption 2: The system lag l is known.
Assumption 3: The system state dimension **Assumption 4: (DeePC)** $T \geq (m+1)(T_{ini}+N+n(\mathscr{B}))-1$. $n(\mathscr{B})$ is known.

Assumptions 2 and 3 are necessary because the use of state-space matrices in defining the systems provides direct access to both the system lag l and the system state dimension $n(\mathscr{B})$. However, in practical implementations aligned with the intended use of this framework, these parameters would need to be identified through data-driven techniques that do not rely on explicit knowledge of the state-space matrices (A, B, C, D).

2) Building the Hankel Matrices: The Hankel matrices $H_L(u^d), H_L(y^d)$ are constructed from the offline data obtained in the previous step. These matrices are formed by shifting and vertically stacking input and output data vectors as follows:

$$H_L(u^d) = \begin{bmatrix} u_1^d & \cdots & u_{T-L+1} \\ u_2^d & \cdots & u_{T-L+2} \\ \vdots & \ddots & \vdots \\ u_L^d & \cdots & u_T \end{bmatrix},$$

$$H_L(y^d) = \begin{bmatrix} y_1^d & \cdots & y_{T-L+1} \\ y_2^d & \cdots & y_{T-L+2} \\ \vdots & \ddots & \vdots \\ y_L^d & \cdots & y_T \end{bmatrix}$$

Hereby, we only ensure that $L \leq T$ in case of DeePC, and set $L = N + 2T_{ini}$ for DDSF. Additionally, we verify that the Hankel matrix $H_L(u^d)$ has full rank according to the persistency of excitation condition stated in (Def. 2).

3) The Receding Horizon Loop: For T_{sim} simulation steps, starting from either a predefined initial state $x_0 \in \mathbb{R}^n$ or an initial trajectory $Traj_{ini} \in \mathbb{R}^{(m+p)T_{ini}}$, an optimization problem is solved to achieve the specified control objective. The solution yields a sequence of optimal input vectors. From this sequence, the first s control inputs $(u_1^*, \ldots, u_s^*)$ are applied to the system and, along with the corresponding system outputs $(y_1^*, \ldots, y_s^*)$, recorded into a dictionary-like data structure that stores simulation results. While employing step sizes $s > 1$, when feasible, would reduce computational costs, our experiments have been conducted exclusively with $s = 1$.

4) Encoding and Solving the Optimization Problem: Using the Hankel matrices $H_L(u^d), H_L(y^d)$ obtained in the previous step, we encode the DDSF optimization problem (eq. (14)). We define the decision variables of the optimization problem as symbolic variables $\alpha \in \mathbb{R}^{T-L+1}$, $\bar{u} \in \mathbb{R}^{mL}$, $\bar{y} \in \mathbb{R}^{pL}$. Here, $\bar{y}$ is fully determined by α and $\bar{u}^*$, but we encoded it as an independent variable for easier handling both during the encoding of the constraints and the extraction of the resulting optimal value for $\bar{y}$. For each system, we compute equilibria by solving $Ax_t + Bu_t = 0$ in the linear case and $f(x_t, u_t) = 0$ in the nonlinear case.
The equilibrium points are used to populate the terminal safe trajectory $Traj_s = \{(u_{eq}, y_{eq})\}_{k=N}^{N+T_{ini}-1}$. The initial trajectory $Traj_{ini} = \{(u_k, y_k)\}_{k=-T_{ini}}^{k=-1}$ corresponds to the initial T_{ini} pairs of the offline trajectory (u^d, y^d). We have implemented various solver options to perform the optimization task. For most experiments, we opted to use the OSQP [5] solver configured with a maximum iteration limit of 30000 and both absolute and relative tolerances set to 10^{-5}. Unless stated otherwise, this configuration can be assumed as the default for the optimizer.
The DeePC optimization problem was encoded in a similar manner using symbolic variables $g \in \mathbb{R}^{T-N-T_{ini}+1}, u \in \mathbb{R}^{mN}, y \in \mathbb{R}^{pN}$ and the same set of optimizer options.

5) Regularization Methods: To ensure the numerical stability of the optimization problem, we employed standard regularization techniques, including normalization and preconditioning of the Hankel matrices, as well as ridge regularization of the parameter $\alpha(t)$ in equation (14b). The default mode of regularization used in the experiments is detailed below.
Regularized objective:

$$\min_{\alpha(t),\bar{u}(t),\bar{y}(t)} \|\bar{u}_0(t) - u_l(t)\|_R^2 + \|\alpha(t)\|_\lambda^2 \tag{15a}$$

$$H := \begin{bmatrix} H_L(u^d) \\ H_L(y^d) \end{bmatrix} \tag{15b}$$

Regularized Hankel matrix:

$$H_{reg} := \|H\|_F^{-1} H + \varepsilon\, \mathbb{I} \tag{15c}$$

Trajectory representation with the regularized Hankel matrix:

$$\begin{bmatrix} \bar{u}_{[-T_{\text{ini}}, N+T_{\text{ini}}-1]}(t) \\ \bar{y}_{[-T_{\text{ini}}, N+T_{\text{ini}}-1]}(t) \end{bmatrix} = H_{reg}\, \alpha(t) \tag{15d}$$

,where $\|H\|_F := \sqrt{\sum_{i=1}^{d_1} \sum_{j=1}^{d_2} |h_{ij}|^2}$ and $H, H_{reg} \in \mathbb{R}^{d_1 \times d_2}$ with $d_1 := (m+p)(N+2T_{ini})$, $d_2 := (T - N - 2T_{ini} + 1)$. In addition, the safety constraints were relaxed by a constant fraction $0 < \varepsilon \ll 1$ of the available range:

$$v_{\min} \leftarrow v_{\min} - \frac{\varepsilon}{2} \cdot (v_{\max} - v_{\min}), \tag{16a}$$

$$v_{\max} \leftarrow v_{\max} + \frac{\varepsilon}{2} \cdot (v_{\max} - v_{\min}) \tag{16b}$$

Unless otherwise specified, the experiments described here assume no additional regularization.

6) Computing the Safe Terminal Set: As described in [1], the safe terminal set was constructed from the system's equilibria, as defined in Def. 7. For linear systems, these equilibria were obtained by solving $Ax_t + Bu_t = 0$, where the trvial solution $(\vec{0}, \vec{0})$ always exists. For nonlinear systems, equilibria were determined by solving the steady-state condition $\dot{x} = f(x, u) = 0$, where $f(x, u)$ describes the system's dynamics. In addition to a real-valued solution, we computed parametrized solutions to the steady-state condition, and used them, whenever possible, to identify the equilibrium that lay closest to the systems' current operating point. The intention behind this approach was to enable the safe terminal set to reflect the operational context of the system, and to observe the effect of this relaxation on optimizer performance.

B. Software

The entire codebase for this project was implemented in MATLAB and is organized into three core modules:

1) Algorithms Module (`../algorithms`): This module contains the implementations of the key control algorithms: Data-Driven Safety Filters (DDSF), Data-Enabled Predictive Control (DeePC), and Model Predictive Control (MPC).

2) Utilities Module (`../utils`): This module provides essential utility functions that underpin the repository's core operations. It includes:

- `../utils/data`: Implements methods for generating input-output trajectories from both linear and nonlinear systems.
- `../utils/systems`: Provides a structured and coherent way of defining dynamical systems, allowing seamless modeling of linear and nonlinear systems. This abstraction ensures consistency across the codebase and simplifies experimentation with different system configurations.
- `../utils/bst`: Implements key BST utilities, such as constructing Hankel matrices and validating their rank condition. Other functions ensure that the collected data satisfies necessary conditions, such as persistency of excitation, to enable accurate system identification and control.
- `../utils/evaluation`: Implements a suite of performance metrics, such as convergence speed, final accuracy, and conservatism (described in the discussion), to systematically assess and compare the behavior of each algorithm.

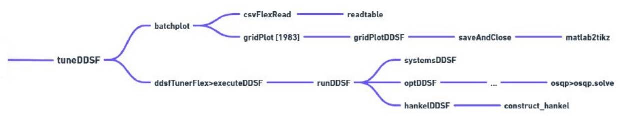


Fig. 1. (Simplified) call hierarchy for `../experiments/tuneDDSF.m`

3) Experiments Module (`../experiments`): This module contains scripts for testing, tuning, and evaluating the implemented algorithms:

- `../experiments/tuneDDSF.m`: Runs the DDSF algorithm under a range of configurations. It optionally stores simulation results as `.csv` files and generates plots illustrating the evolution of system inputs and outputs over time, saving these outputs for detailed post-analysis.
- `../experiments/tuneDeePC.m`: Performs similar tasks for the DeePC algorithm, allowing users to explore its performance under diverse parameter settings.

The `DDSF` and `DeePC` tuner scripts serve as primary entry points for the repository, demonstrating how to utilize the core functionalities for experimental scenarios.

4) Code Availability: The complete codebase for this project is publicly accessible at [6].

C. Results

1) Data-Driven Safety Filter - The Quadrotor: This section demonstrates the behavior of the Data-Driven Safety Filter (DDSF) algorithm using a simplified linear approximation of a quadrotor model, directly adapted from the method described in the DDSF paper.

The system state is represented by a vector $x(t) \in \mathbb{R}^{12}$, containing spatial and angular positions, as well as velocities: $x := [\phi,\ \theta,\ \psi,\ p,\ q,\ r,\ u,\ v,\ w,\ x,\ y,\ z]^\top$. The control input $[u_1,\ u_2,\ u_3,\ u_4]^\top$ is 4-dimensional vector representing the rotor thrusts. The quadrotor has six degrees of freedom, three translational (x, y, z) and three rotational (ϕ, θ, ψ), therefore the output vector can be expressed as $[\phi,\ \theta,\ \psi,\ x,\ y,\ z]^\top$. System properties as well as the parameter configuration under which the results in this section were obtained can be found in the appendix (Sec. IV-C1).

The figures below show the evolution of the control inputs $\{u_i(t)\}_{t \in T_{\text{sim}}}$ (rotor thrusts) over $T_{sim} = 25$ iterations, individually for each rotor, compared to the learning inputs $\{ul_i(t)\}_{t \in T_{\text{sim}}}$

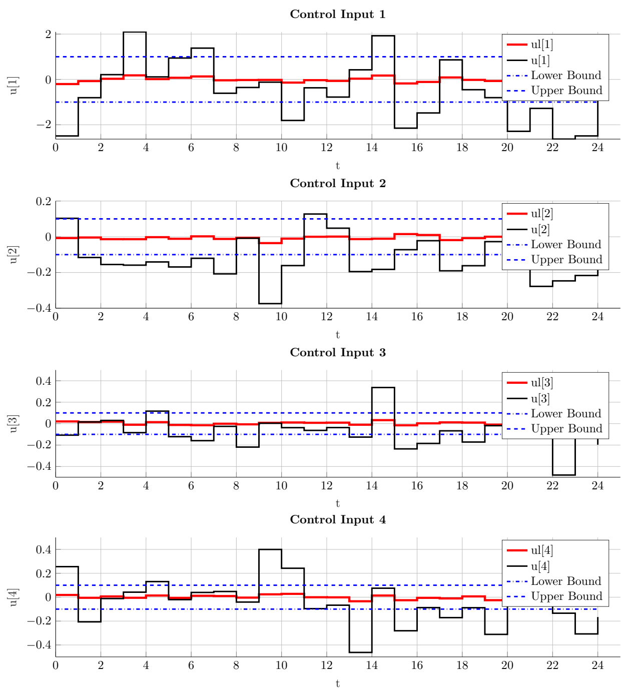


Fig. 2. DDSF inputs

The learning inputs were obtained by genrating random input signals of appropriate dimensions.

The figures below compare the resulting output sequence to the projected output sequence computed from the learning inputs $\{ul_i(t)\}$ by solving eq. (14b) for a single step.

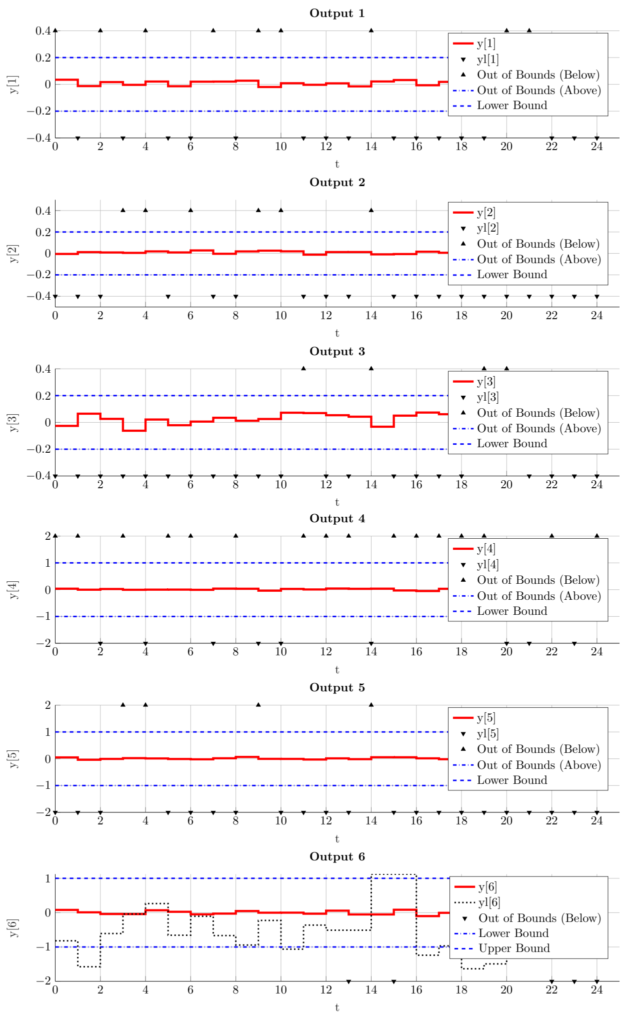


Fig. 3. DDSF outputs

Overall, it can be said that the filter demonstrates rather conservative behavior, which is expected given the high dimensionality and coupling inherent in the dynamics of the quadrotor. The conservatism score for this experiment according to Def. 13 was `0.9366`.

2) Data-Driven Safety Filter - The Inverted Pendulum: This second example for the DDSF focuses on a simple inverted pendulum system, a classic benchmark in control theory. The system comprises a cart and a pendulum, with dynamics governed by a nonlinear interaction between gravity, inertia, and external forces. The detailed state-space representation, system parameters, and the specific configuration of the DDSF applied to this system are provided in Sec. IV-C2.

Compared to the quadrotor example, the DDSF demonstrates a less conservative behavior for this system: the output tracks closely track the learning inputs as long as they remain within safe operational boundaries of the system. When the learning inputs approach or exceed these safety limits, the safety filter adjusts the outputs to comply with the relaxed (see eq. (16)) boundary values. For this system and configuration,

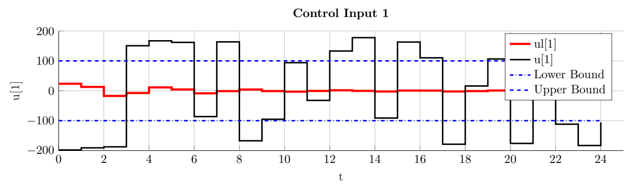


Fig. 4. DDSF - inputs

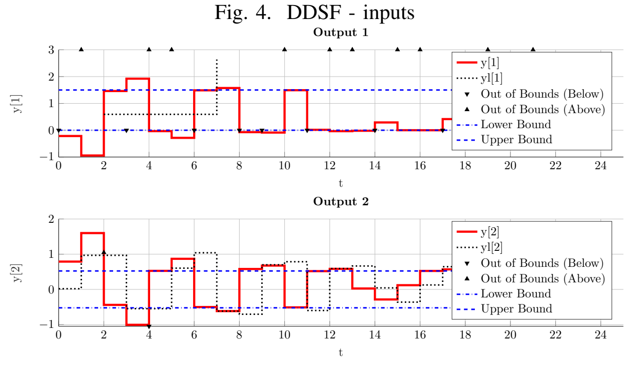


Fig. 5. DDSF - outputs

we compute a conservatism score of `0.5735`.

3) Data-Enabled Model-Predictive Control: To demonstrate and verify the algorithm's behavior and correctness, we use a simple linear vertical mass-damper system, which can be interpreted as a simplified model of a car suspension. The details of the system dynamics, parameters, and the algorithmic configuration used to run the DeePC algorithm are provided in Sec. IV-C3. The figures above illustrate

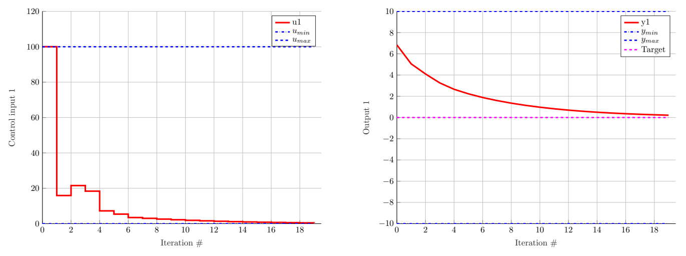


Fig. 6. DeePC - Damper

the evolution of the controlled mass-spring-damper system, starting from an initial vertical displacement of 9 units above the equilibrium and converging toward the target state of zero vertical displacement.

IV. Conclusion

This study explored the translation of classical algorithms into the Behavioral Systems Theory (BST) framework. While the results demonstrated that such translations are feasible, their performance remained constrained by numerical stability issues observed during implementation. Moreover, the study was limited to linear systems, leaving open the question of applicability to nonlinear dynamics. A critical limitation identified was the dependency of certain computations—such as system lag and equilibrium points—on the state-space representation, which partially undermines the intended independence from explicit system models. These findings highlight the need for further research to address numerical challenges, extend the framework to nonlinear systems, and eliminate the methodology's reliance on model-based computations by fully transitioning them into the data-driven paradigm.

For our simulations, we relied on explicit physical models of the systems and deliberately avoided perturbing the data generation process to introduce measurement noise, as the primary objective was to provide a simplified assessment of the framework's fundamental viability. However, realistic application scenarios for this framework would involve measurement noise, as inputs and outputs would need to be directly measured from the system during operation. Consequently, future research must either test the framework on realistic application scenarios by collecting data online from the system of interest or simulate various noise sources to mimic the potential interference encountered in real-life data collection.

Additionally, our experiments assumed zero time delay, which is highly unrealistic in most practical scenarios. While the original work on the Data-Driven examined the effect of varying levels of time-delay on the filter's behavior, those findings were limited to a one-dimensional system with relatively simple dynamics. Further research should investigate whether the filter retains its effectiveness when applied to higher-dimensional systems with more complex dynamics, such as a quadrotor, under realistic levels of time delay.

To conclude, it would be valuable to systematically evaluate and quantify the impact of different types and levels of noise, as well as varying levels of time delay, on the performance of the Data-Driven Safety Filter to better understand its robustness and limitations in practical settings.

Appendix

A. Notation

- N : Prediction horizon
- T_{ini} : Length of the initial trajectory
- T : Length of the offline trajectory
- T_{sim} : Simulation length
- L : Order of the Hankel matrix (DDSF: $L = N + 2T_{ini}$)
- m : Input dimension
- p : Output dimension
- n : System state dimension
- $u^d (:= \{u_k^d\}_{k=0}^{T-1})$: Offline control input sequence
- $y^d (:= \{y_k^d\}_{k=0}^{T-1})$: Offline output sequence
- $H_L(w) \; :\in \mathbb{R}^{dL\times(T-L+1)}$ Hankel matrix of order L, constructed from vector $w \in \mathbb{R}^{d\times T}$
- $\mathscr{B}$: System behavior / system
- $n(\mathscr{B})$: System state dimension
- $l(A, C)$: Latency
- σ : Shift operator

B. Background Material

Definition 8. ***(LTI system)*** *A system $\Sigma = (\mathbb{T}, \mathbb{W}, \mathscr{B})$ is called*

(i) ***linear****, if $\mathscr{B}$ is a vector space*

(ii) ***time-invariant****, if $\mathscr{B} \subseteq \sigma\mathscr{B}$, where σ is the backward shift operator defined as $(\sigma w)(t) := w(t + 1)$*

(iii) ***complete****, if $\mathscr{B}$ is closed in the topology of pointwise convergence (also see Def. 11): $(\forall\{w_i\}_{i\in\mathbb{N}} \subseteq \mathscr{B} : \lim_{i\to\infty} w_i(t) := w_\infty) \implies w_\infty \in \mathscr{B}$. If the properties (i) and (ii) are satisfied, completeness of Σ becomes equivalent to the finite dimensionality of the signal space $\mathbb{W}$.*

(iv) an ***LTI system*** *if properties (i)-(iii) hold.*

Definition 9. ***(LTI system trajectory)*** *Let $\mathscr{B}$ be an LTI system and (A, B, C, D) its minimal realization, then $\{u_k, y_k\}_{k=0}^{N-1}$ is an input-output sequence of this system if there exists an initial condition $x_0 \in \mathbb{R}^n$ and a state sequence $\{x_k\}_{k=0}^{N}$ such that equations (1a), (1b) hold for all $k \in \{0, 1, 2, \dots, N - 1\}$.*

Definition 10. ***(LTI system lag)*** *The lag of the system from (1) is denoted as $l(A, C)$, and is defined as the smallest integer l, for which the observability matrix*

$$O_l(A, C) := \left(C, CA, \dots, CA^{l-1}\right) \tag{17}$$

has full rank ([7], Ch. 7.8). For minimal realizations of LTI systems, system state dimension and lag are equal ($l(\mathscr{B}) = n(\mathscr{B})$).

Definition 11. ***(Completeness)*** *A dynamical system $\Sigma = (\mathbb{T}, \mathbb{W}, \mathscr{B})$ is complete if the following property holds:*

$$\left(\forall t_0, t_1 \in \mathbb{T}, t_0 \leq t_1 : w|_{[t_0,t_1]} \in \mathscr{B}|_{[t_0,t_1]}\right) \implies w \in \mathscr{B} \tag{18}$$

,where $w : \mathbb{T} \mapsto \mathbb{W}$ is a trajectory of the system, $w|_{[t_0,t_1]}$ its restriction to a finite time interval $[t_0, t_1]$ and $\mathscr{B}|_{[t_0,t_1]}$ is the set of all trajectories in $\mathscr{B}$ restricted to $[t_0, t_1]$. In other words, under completeness, if a finite segment of a trajectory w is consistent with the system behavior for the time interval $[t_0, t_1]$, the entire trajectory w must lie in $\mathscr{B}$.

Definition 12. ***(Kernel representation)*** *The behavior $\mathscr{B}$ of a system can be represented through so-called annihilating functions defined as: $f : \mathbb{W}^{\mathbb{T}} \to \mathbb{R}^g$ s.t. $f(w) = 0 \iff w \in \mathscr{B}$ (or equivalently, $\mathscr{B} = \{w \in \mathbb{W}^{\mathbb{T}} \mid f(w) = 0\}$). A more compact representation of system behavior can be obtained through Kernel representation as follows:*

1) *Assuming an LTI system, the relationships between consecutive trajectory samples $w(t), w(t + 1)$ are linear, making it possible to represent system behavior through finite difference equations:*
   $R_0w(t) + R_1w(t + 1) + \cdots + R_lw(t + l) = 0, \quad \forall\, t \in T$ *, where $R_0, R_1, \dots, R_l \in \mathbb{R}^{g\times w}$ are coefficient matrices and l is the maximum lag.*
2) *With the help of a shift operator σ, defined as $(\sigma w)(t) = w(t+1)$, the difference equations can be re-written more compactly as: $R_0w(t)+R_1(\sigma w)(t)+\cdots+R_l(\sigma^l w)(t) = 0$, and even more compactly as $R(\sigma)w = 0$ using the polyomial matrix $R(\sigma) := R_0 + R_1\sigma + \cdots + R_l\sigma^l$.*
3) *This gives rise to a Kernel representation of system behavior as:*

$$\mathscr{B} = \ker(R(\sigma)) = \{w \in W^T \mid R(\sigma)w = 0\} \tag{19}$$

To obtain an inpute-output relationship based on this behavioral system representation, the trajectories w can be partitioned into $\begin{bmatrix} u \\ y \end{bmatrix}$ via a projection operator Π s.t. $w = \Pi \begin{bmatrix} u \\ y \end{bmatrix}$, which leads to the following input/output representation:

$$\begin{aligned} R(\sigma) &:= \begin{bmatrix} Q(\sigma) & -P(\sigma) \end{bmatrix}; \\ R(\sigma)w = 0 \iff & \begin{bmatrix} Q(\sigma) & -P(\sigma) \end{bmatrix} \begin{bmatrix} u \\ y \end{bmatrix} = 0 \\ \iff & P(\sigma)y = Q(\sigma)u \end{aligned}$$

where $P(\sigma)y$ and $Q(\sigma)$ are polynomial matrices as well. The I/O relationship $P(\sigma)y = Q(\sigma)u$ implies that any trajectory $w = (u, y) \in \mathscr{B}$ can be expressed as a linear combination of the columns of the Hankel matrix (see Def. 1) of sufficient order, consctructed from a signal of sufficient length (see Def. 2).

Definition 13. ***(Conservatism of the Safety Filter)*** *Given the filtered (safe) outputs $y^d := \{y^d(t)\}_{t\in[T_{sim}]}$, the hypothetical outputs projected from the learning agent's control inputs $y_l^d := \{y_l^d(t)\}_{t\in[T_{sim}]}$, and a cutoff value $thr \in [0, 100]$, we define the conservatism of the safety filter as:*

$$c := \frac{1}{p} \sum_{d=1}^{p} c^d \tag{20a}$$

where c^d measures conservatism along the d-th output axis and is defined as:

$$c^d := 1 - \frac{R^d_{thr}}{Var[\hat{y}^d_l] + 1}, \quad \forall d \in [p]. \tag{20b}$$

Here, R^d_{thr} quantifies the percentage of the safe output range $[y^d_{\min}, y^d_{\max}]$ covered by the filtered signal, while excluding outliers based on the threshold thr:

$$R^d_{thr} = \frac{Q(y^d, 1 - thr/2) - Q(y^d, thr/2)}{y^d_{\max} - y^d_{\min}}. \tag{20c}$$

The function $Q(x, p)$ represents the p-th quantile of the signal x, capturing the effective range of the filtered signal y^d.

To normalize the variance of the learning agent's input signal, we define the sliding-window normalized signal $\hat{y}^d_l$ as:

$$\hat{y}^d_l := \{\hat{y}^d_l(t)\}_{t\in T_{sim}}, \tag{20d}$$

$$\hat{y}^d_l(t) := \frac{y^d_l(t) - \mu_t}{\sigma_t}, \tag{20e}$$

$$\mu_t := \frac{1}{l} \cdot \sum_{n=0}^{l-1} y^d_l(t-n), \tag{20f}$$

$$\sigma_t := \sqrt{\frac{1}{l} \cdot \sum_{n=0}^{l-1} (y^d_l(t-n) - \mu_t)^2}. \tag{20g}$$

Thus, the overall metric $c \in [0, 1]$ represents the portion of the available output range covered by the filtered signal, normalized by the variance of the hypothetical outputs computed from the learning agent's control inputs. Higher values of c correspond to more conservative filtering.

Definition 14. ***Performance of the DeePC algorithm*** *Given an output sequence $\{y(t)\}_{t\in[T_{sim}]}$, we use the following metric to evaluate the DeePC algorithm's performance on a given parameter configuration:*

$$m_{DeePC} := \hat{w}_1 \cdot \left(1 - \frac{\tilde{T}_{conv}}{T_{sim}}\right) + \hat{w}_2 \cdot \left(1 - \frac{\|y(T_{sim}) - y_{target}\|}{\|y_{target}\|}\right) \tag{21a}$$

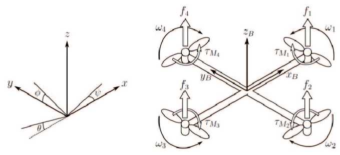
,where $\tilde{T}_{conv}$ measures the number of simulation steps until convergence:

$$T_{conv} := \left\{ t \in [0, T_{sim}] \;\middle|\; \frac{\|y(t) - y_{target}\|}{\|y_{target}\|} \le \varepsilon \right\} \tag{22a}$$

$$\tilde{T}_{conv} = \begin{cases} \min \; T_{conv}, & \text{if } T_{conv} \neq \emptyset, \\ T_{sim}, & \text{otherwise} \end{cases} \tag{22b}$$

,and $\hat{w}_1 . \hat{w}_2$ are normalized weight factors. Thus, the overall metric balances convergence speed and last measured distance to target.

C. Experimental Details



1) 6-DOF Quadrotor: The minimal state-space representation of the linearized 6-DOF quadrator model is given as:

$$\begin{aligned} \dot{x_t} &= Ax_t + Bu_t \\ y_t &= Cx_t + Du_t \end{aligned}$$

,where $A \in \mathbb{R}^{12\times12}, B \in \mathbb{R}^{12\times4}, C \in \mathbb{R}^{6\times12}, D \in \mathbb{R}^{6\times4}$ are sparse matrices with:

$$A = \begin{cases} A[1,4] = 1, & A[2,5] = 1, \\ A[3,6] = 1, & A[7,2] = -g, \\ A[8,1] = g, & A[10,7] = 1, \\ A[11,8] = 1, & A[12,9] = 1. \end{cases}$$

$$B = \begin{cases} B[4,2] = I_{xx}^{-1}, & B[5,3] = I_{yy}^{-1}, \\ B[6,4] = I_{zz}^{-1}, & B[9,1] = m^{-1}. \end{cases}$$

$$C = \begin{bmatrix} C[1,1] & = & 1, \\ C[2,2] & = & 1, \\ C[3,3] & = & 1, \\ C[4,10] & = & 1, \\ C[5,11] & = & 1, \\ C[6,12] & = & 1. \end{bmatrix}$$

$$D[i,j] = 0 \quad \forall\, i \in [6],\, j \in [4].$$

The results presented in Sec. III-C1 were obtained under the following parameter configurations:

TABLE I
QUADROTOR: FILTER PARAMETERS

Parameter	Value	Description
N_p	15	Prediction horizon
T_{ini}	2	Initial trajectory length
S_f	Equilibrium point	Safe terminal set
R	$\mathbb{I}_{4\times 4}$	Cost matrix

TABLE II
QUADROTOR: SYSTEM PROPERTIES

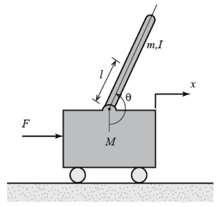
Parameter	Value	Units
I_{xx}	10^{-3}	[Kg.m^2]
I_{yy}	10^{-3}	[Kg.m^2]
I_{zz}	10^{-3}	[Kg.m^2]
mass	0.2	[Kg]
g	9.81	[m/s^2]
T_s	0.1	[s]

TABLE III
QUADROTOR: INPUT CONSTRAINTS

Input	Min Value	Max Value
u_1	-1 [N]	$+1$ [N]
u_2	-0.1 [N.m]	$+0.1$ [N.m]
u_3	-0.1 [N.m]	$+0.1$ [N.m]
u_4	-0.1 [N.m]	$+0.1$ [N.m]

TABLE IV
QUADROTOR: STATE CONSTRAINTS

State	Min Value	Max Value
θ	-0.2 [rad]	$+0.2$ [rad]
ϕ	-0.2 [rad]	$+0.2$ [rad]
ψ	-0.2 [rad]	$+0.2$ [rad]
x	-1 [m]	$+1$ [m]
y	-1 [m]	$+1$ [m]
z	-1 [m]	$+1$ [m]



2) Inverted Pendulum: The inverted pendulum system consists of a cart of mass M with a pendulum of mass m and length l attached. The system dynamics are influenced by gravity g, friction b, and the pendulum's moment of inertia I.
The continuous-time state-space matrices are computed as follows:

$$A_c = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{(I+ml^2)b}{p} & \frac{m^2gl^2}{p} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{mlb}{p} & \frac{mgl(M+m)}{p} & 0 \end{bmatrix}, \quad B_c = \begin{bmatrix} 0 \\ \frac{I+ml^2}{p} \\ 0 \\ \frac{ml}{p} \end{bmatrix}.$$

,where $p = I(M+m) + Mml^2$. For simulation purposes, the system is discretized using a sampling time of $\Delta t = 0.1\,\mathrm{s}$, resulting in discrete-time matrices A and B.

The output matrix C and feedthrough matrix D are given by:

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

The system parameters, constraints, and filter configuration used to obtain the results in Sec. III-C2 are detailed in the tables below:

TABLE V
INVERTED PENDULUM: SYSTEM PARAMETERS

Parameter	Value	Units
M	50	[Kg]
m	2	[Kg]
I	0.6	[Kg $\cdot$ m^2]
l	3	[m]
g	9.81	[m/s^2]
b	0.1	[N $\cdot$ s/m]
Δt	0.1	[s]

TABLE VI
INVERTED PENDULUM: INPUT CONSTRAINTS

Input	Min Value	Max Value
Force (F)	-100 [N]	$+100$ [N]

TABLE VII
INVERTED PENDULUM: OUTPUT CONSTRAINTS

Output	Min Value	Max Value
Position (x)	0 [m]	1.5 [m]
Angle (θ)	$-\frac{\pi}{6}$ [rad]	$+\frac{\pi}{6}$ [rad]

TABLE VIII
INVERTED PENDULUM: INITIAL AND TARGET STATES

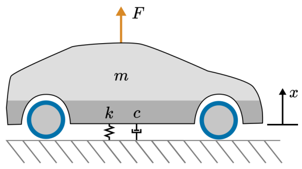
State	Value
Initial State (x_{ini})	$\begin{bmatrix} 0.5 & 0 & 0.087 & 0 \end{bmatrix}^\top$
Target State (y_{target})	$\begin{bmatrix} 1.45 & 0 & 0 & 0 \end{bmatrix}^\top$

3) Mass-Spring Damper: The mass-spring-damper system is a classic second-order mechanical system consisting of a mass m, connected to a spring with stiffness k, and a damper with a damping coefficient b.

The dynamics of the system are represented in state-space form as:

TABLE IX
INVERTED PENDULUM: FILTER PARAMETERS

Parameter	Value	Description
T	49	Total timesteps
T_{ini}	5	Initial trajectory length
N	15	Prediction horizon



$$x_{t+1} = Ax_t + Bu_t,$$
$$y_t = Cx_t + Du_t,$$

where:

$$A = \begin{bmatrix} 1 & dt \\ -\frac{k \cdot dt}{m} & 1 - \frac{b \cdot dt}{m} \end{bmatrix}, B = \begin{bmatrix} 0 \\ \frac{dt}{m} \end{bmatrix},$$
$$\mathbf{C} = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad \mathbf{D} = 0.$$

The state vector is defined as $x_t = [x_t, v_t]^\top$, where x_t is the vertical displacement, and v_t is the vertical velocity. The input u_t represents the external force applied to the mass. The system output $y_t = x_t$ tracks the vertical displacement.

System parameters, safety constraints and filter configuration used to obtain the results presented in Sec. III-C3 can be found in Tab. X - Tab. IX.

TABLE X
MASS-SPRING DAMPER: SYSTEM PARAMETERS

Parameter	Value	Units
m	100	[Kg]
k	1	[N/m]
b	0.2	[Ns/m]
dt	1	[s]

TABLE XI
MASS-SPRING DAMPER: INPUT CONSTRAINTS

Input	Min Value	Max Value
u_t	0 [N]	100 [N]

TABLE XII
MASS-SPRING DAMPER: OUTPUT CONSTRAINTS

Output	Min Value	Max Value
y_t	-10 [m]	$+10$ [m]

TABLE XIII
MASS-SPRING DAMPER: INITIAL AND TARGET STATES

State	Value
Initial State (x_{ini})	$\begin{bmatrix} 9 & 2 \end{bmatrix}^\top$
Target State (y_{target})	0

TABLE XIV
MASS-SPRING DAMPER: FILTER PARAMETERS

Parameter	Value	Description
T_{ini}	1	Initial trajectory length
N	10	Prediction horizon
Q	$100 \cdot \mathbf{I}$	Output weight matrix
R	$0.01 \cdot \mathbf{I}$	Input weight matrix

REFERENCES

[1] M. Bajelani and K. van Heusden, "Data-driven safety filter: An input-output perspective," 2023. [Online]. Available: https://arxiv.org/abs/2309.00189

[2] J. C. Willems, P. Rapisarda, I. Markovsky, and B. L. De Moor, "A note on persistency of excitation," Systems & Control Letters, vol. 54, no. 4, pp. 325–329, 2005.

[3] J. Coulson, J. Lygeros, and F. Dörfler, "Data-enabled predictive control: In the shallows of the deepc," 2019. [Online]. Available: https://arxiv.org/abs/1811.05890

[4] H. Chen and F. Allgöwer, "A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability," Automatica, vol. 34, no. 10, pp. 1205–1217, 1998.

[5] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: an operator splitting solver for quadratic programs," Mathematical Programming Computation, vol. 12, no. 4, pp. 637–672, 2020. [Online]. Available: https://doi.org/10.1007/s12532-020-00179-2

[6] A. Ulusarslan, "Behavioral systems theory in optimization-based control: A survey," https://github.com/aiulus/ddsf.git, 2025.

[7] I. Markovsky, J. C. Willems, S. V. Huffel, and B. D. Moor, "Exact and approximate modeling of linear systems: A behavioral approach," 2006. [Online]. Available: https://api.semanticscholar.org/CorpusID:14612268