DFG Deutsche Forschungsgemeinschaft
Priority Programme 1962

On the Universal Transformation of Data-Driven Models to Control Systems

Sebastian Peitz, Katharina Bieker

SPP 1962 | Non-smooth and Complementarity-based Distributed Parameter Systems: Simulation and Hierarchical Optimization

Preprint Number SPP1962-156

received on February 8, 2021

Edited by
SPP1962 at Weierstrass Institute for Applied Analysis and Stochastics (WIAS)
Leibniz Institute in the Forschungsverbund Berlin e.V.
Mohrenstraße 39, 10117 Berlin, Germany
E-Mail: spp1962@wias-berlin.de

World Wide Web: http://spp1962.wias-berlin.de/

On the Universal Transformation of Data-Driven Models to Control Systems

Sebastian Peitz[1] and Katharina Bieker[1]

[1]Department of Mathematics, Paderborn University, Germany

Abstract

As in almost every other branch of science, the major advances in data science and machine learning have also resulted in significant improvements regarding the modeling and simulation of nonlinear dynamical systems. It is nowadays possible to make accurate medium to long-term predictions of highly complex systems such as the weather, the dynamics within a nuclear fusion reactor, of disease models or the stock market in a very efficient manner. In many cases, predictive methods are advertised to ultimately be useful for control, as the control of high-dimensional nonlinear systems is an engineering grand challenge with huge potential in areas such as clean and efficient energy production, or the development of advanced medical devices. However, the question of how to use a predictive model for control is often left unanswered due to the associated challenges, namely a significantly higher system complexity, the requirement of much larger data sets and an increased and often problem-specific modeling effort. To solve these issues, we present a universal framework (which we call QuaSiModO: Quantization-Simulation-Modeling-Optimization) to transform arbitrary predictive models into control systems and use them for feedback control. The advantages of our approach are a linear increase in data requirements with respect to the control dimension, performance guarantees that rely exclusively on the accuracy of the predictive model, and only little prior knowledge requirements in control theory to solve complex control problems. In particular the latter point is of key importance to enable a large number of researchers and practitioners to exploit the ever increasing capabilities of predictive models for control in a straight-forward and systematic fashion.

Many challenges of our modern society could be addressed by significantly improving the control performance for highly complex systems in real-time, one example being the ever increasing demand for energy and the associated question of how it can be met in a cost-efficient and sustainable manner. The World Energy Outlook 2018 [1] suggests that – assuming current policies remain unchanged and that the efficiency can be increased as expected – the total demand in energy will increase by 25% until 2040. Consequently, a further increase in efficiency of conventional processes for producing electrical energy (e.g., turbomachinery, combustion, wind energy and tidal energy) is required as well as the development of new concepts – nuclear fusion being a prominent and very promising example [2]. Furthermore, the efficiency of complex systems such as aircraft needs to be increased in order to limit the energy requirements. All the above-mentioned applications are governed by high-dimensional nonlinear dynamics, which exhibit complex multi-scale phenomena and are thus extremely difficult to control. Similar challenges arise in other areas such as the health sector, where complex dynamics govern our breathing [3], the flow of blood within arteries [4] or of cerebrospinal fluid within the brain [5], and – more recently – the dynamics of pandemics such as COVID-19 [6].

The efficient prediction of complex systems such as the ones mentioned above is often hindered by the fact that the system dynamics are either very expensive to simulate or even unknown. Researchers have been investigating ways to accelerate the solution by using data for decades, the *Proper Orthogonal Decomposition (POD)* being an early and very prominent example [7]. Therein, the original system is projected onto a linear subspace spanned by modes that have been computed from simulation data of the full system state. More recently, the major advances in data science and machine learning have lead to a plethora of new possibilities to overcome the shortcomings of projection-based methods like POD, such as the required knowledge of the system dynamics and the quickly growing model dimension with increasing complexity. Important examples are different artificial neural network architectures such as *Long Short-Term Memory (LSTM)* Networks [8, 9] or *Reservoir Computers / Echo State Networks*

[10, 11], regression-based frameworks for the identification of nonlinear dynamics [12, 13], or numerical approximations of the *Koopman operator* [14, 15, 16], which describes the linear dynamics of observable functions. These methods facilitate the efficient simulation and prediction of high-dimensional spatio-temporal dynamics using measurement data, without requiring prior system knowledge. One important application is, among many others, the prediction of rare events [17], for instance in nuclear fusion reactors [18].

The large success of data-driven modeling has also attracted the attention of the control community, where many approaches have been presented over the past decades. They can be categorized into the direct learning of feedback signals using, e.g., feed-forward neural networks [19] or reinforcement learning [20, 21, 22], and into MPC methods using either intrusive approaches (i.e., the equations need to be known) such as POD [23] or black-box methods, for instance specific neural network architectures [24]. MPC allows for a particularly easy implementation of both control and state constraints. However, a drawback is that the construction of surrogate models with inputs is often very tedious and in addition highly problem-specific. For instance, in the case of POD, multiple modeling steps and several simulations with different boundary conditions are required [25]. In pure black-box methods, a straight-forward approach to avoid this tedious effort is to transform control inputs via state augmentation [26], by which the system is *autonomized*, and then use "off-the-shelf" methods for autonomous systems. However, this results in significantly increased data requirements due to (a) the increased dimension of the augmented state space and (b) the fact that the dynamics of the control system is not necessarily restricted to a low-dimensional manifold any longer, which is a key enabler for efficient data-driven modeling [27]. Recently, an alternative was presented in [28], where the key observation was that instead of adapting the surrogate model according to the control problem requirements, it can be advantageous to modify the control problem and use a finite set of autonomous systems. This results in a mixed-integer optimal control problem which is considerably harder to solve, and additional favorable properties of the surrogate model (such as the linearity of the Koopman operator [29]) are required to facilitate an efficient solution.

The framework we present in this article – *QuaSiModO* – makes use of the above-mentioned benefits of modifying the control problem. It consists of four main steps (cf. also Figure 1). In these, Problem (I) (which is an optimal control problem with continuous inputs) is successively transformed into related control problems that – as long as the predictive surrogate model is sufficiently accurate – yield optimal trajectories y^* that are close to one another.

(i) **Qua**ntization of the the admissible control U (for instance by replacing the interval $U = [u^{\mathsf{min}}, u^{\mathsf{max}}]$ by the bounds $V = \{u^{\mathsf{min}}, u^{\mathsf{max}}\}$);

(ii) **Si**mulation of the individual autonomous systems (e.g., $\Phi_{u^{\mathsf{min}}}(y) = \Phi(y, u^{\mathsf{min}})$ and $\Phi_{u^{\mathsf{max}}}(y) = \Phi(y, u^{\mathsf{max}})$); Note that this step can also be replaced by collecting measurement data from experiments;

(iii) **Mod**eling of the individual systems – using either the full state y or some observable $z = f(y)$ – via an arbitrary "off-the-shelf" surrogate modeling technique (POD, neural network, Koopman operator, etc.);

(iv) **O**ptimization using the resulting set of autonomous surrogate models and relaxation techniques.

This complex interplay between continuous and integer control modeling as well as between the full system state and observed quantities (e.g., measurements) allows us to utilize the best of both worlds:

- integer controls for efficient data-driven modeling using arbitrary predictive models,
- continuous control inputs for real-time control, and
- existing error bounds for predictive models.

After introducing the respective steps in detail in the next section, we give a detailed derivation as well as numerical verification of the derived performance guarantees in Section 2. We then present control results on a large variety of complex systems, surrogate modeling techniques and observable types in Section 3.

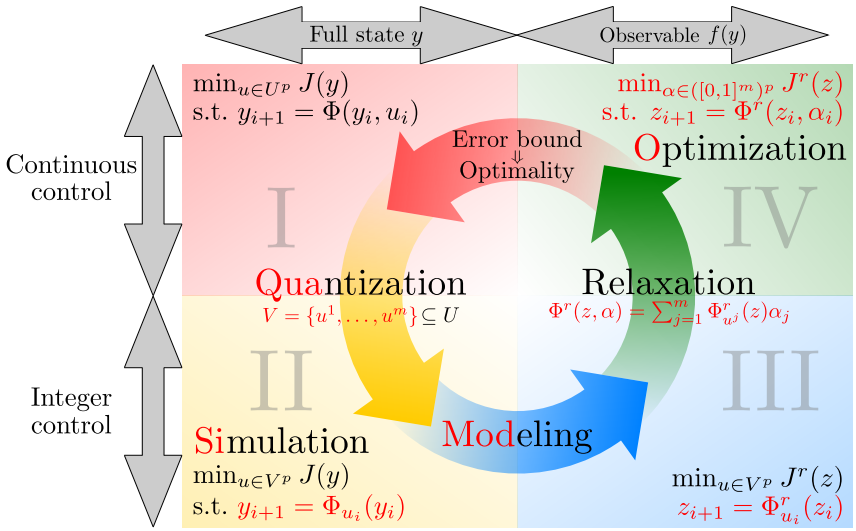


Figure 1: The QuaSiModO framework consisting of the four steps Quantization, Simulation, Modeling and Optimization.

1. QuaSiModO steps

The overarching goal of QuaSiModO is real-time control of complex systems using "off-the shelf" data-driven surrogate modeling techniques and their respective error bounds. In the problems we consider, the state of such a complex dynamical system, denoted by y, is a function of time t and – in the case of partial differential equations (PDEs) – of space x. For ordinary differential equations (ODEs) and PDEs, the dynamics is described by the right-hand side g, e.g., $\dot{y} = g(y(t), u(t))$ in the ODE case, with u being the *control input*. As most surrogate modeling techniques yield discrete-time systems, we directly introduce a time discretization using the time-T-map Φ with constant time step $\Delta t = t_{i+1} - t_i$, $i = 0, 1, \dots$, i.e.,

$$\Phi(y_i, u_i) = y_i + \int_{t_i}^{t_{i+1}} g(y(t), u_i)\, dt = y_{i+1},$$

where $u(t) = u_i \in U$ is constant over the interval $[t_i, t_{i+1})$, and U is the admissible set for the control. A popular example for U are box constraints: $U = [u^{\mathsf{min}}, u^{\mathsf{max}}]$. In the case of PDEs, we use a spatial discretization (such as finite elements), which then yields high-dimensional ODEs. Thus, we will only consider ODEs from now on. Using the above considerations, the overall goal can be formalized in an optimal control problem of the following form:

$$\begin{aligned} &\min_{u\in U^p} J(y) = \min_{u\in U^p} \sum_{i=0}^{p-1} P(y_{i+1}) \\ &\text{s.t.} \quad y_{i+1} = \Phi(y_i, u_i), \qquad i = 0, 1, 2, \dots, \end{aligned} \tag{I}$$

where J is the overall objective function over the time horizon $T = p\Delta t$, and P is the objective function at a particular time instant, e.g., a tracking term $P(y_i) = \|y_i - y_i^{\mathsf{ref}}\|^2$. Note that P does not explicitly depend on u, as this is favorable for the error bounds that we will derive. However, such penalty terms could be included as well.

1.1. Quantization:

In the first step of QuaSiModO, we quantize the control set U such that only a finite subset $V = \{u^1, \dots, u^m\} \subseteq U$ is feasible. This allows us to replace the control system $\Phi(y, u)$ by a finite set of autonomous systems $\Phi_{u^j}(y)$, which yields the following mixed-integer optimal control problem:

$$\begin{aligned} &\min_{u\in V^p} J(y) = \min_{u\in V^p} \sum_{i=0}^{p-1} P(y_{i+1}) \\ &\text{s.t.} \quad y_{i+1} = \Phi_{u_i}(y_i), \qquad i = 0, 1, 2, \dots. \end{aligned} \tag{II}$$

The choice of the entries of V is problem-specific and has an influence on the control performance and on the numerical effort. In addition, as we will see later, the reachable set corresponding to U has to be contained in the convex hull of the reachable set corresponding to V if we want to guarantee similarity between the optimal values of Problems (I) and (II).

1.2. Simulation & Modeling:

Problem (II) allows for the straightforward introduction of surrogate models, as we can simply replace the individual autonomous systems $\Phi_{u^j}(y)$ by respective reduced systems $\Phi^r_{u^j}(z)$ (or – in continuous time – replace g by g^r) that predict the dynamics of a reduced quantity $z = f(y)$. Here, f is called an *observable* that takes arbitrary measurements from the full state. These range from full-state observation (i.e., $z = y$) over partial observations or point-wise measurements to arbitrary nonlinear functions of the state. In many applications – in particular in experimental setups – the choice of f is determined by which measurements are feasible or accessible. For instance, in the case of fluid dynamics, it is highly unrealistic to measure the full state or even take measurements from the interior of the flow domain.

The quantization also simplifies the process of simulation (or, more generally, data collection). Here, we can either collect data from one long time series $Z = [z_1, \dots, z_N]$ with random actuation $u_1, \dots, u_N \in V$ (which we than have to split into data sets for the individual systems), or we can use individual time series for each system:

$$Z^j = [z_1^j, \dots, z_N^j] \quad \text{with} \quad u_1^j = \dots = u_N^j = u^j \in V, \quad j = 1, \dots, m.$$

The fact that we need to construct surrogate models for a set of autonomous systems allows us to use arbitrary predictive models for z, thus giving us straightforward access to a highly active area of research with new techniques being proposed very frequently. These surrogate models then simply replace the full system in Problem (II), yielding:

$$\begin{aligned} \min_{u \in V^p} J^r(z) &= \min_{u \in V^p} \sum_{i=0}^{p-1} P^r(z_{i+1}) \\ \text{s.t.} \quad z_{i+1} &= \Phi^r_{u_i}(z_i), \qquad i = 0, 1, 2, \dots . \end{aligned} \tag{III}$$

Note that the objective function has to be altered as well, as it is now a function of z and not of the full state y. However, this is not a strong limitation, as objective function evaluations are usually based on observable quantities. That is, we assume

$$P^r(f(y)) = P(y) \qquad \text{for all } y \in Y. \tag{1}$$

1.2.1. Surrogate model error

Regarding the *quantification of the modeling error*, researchers have made significant advances over the past decades, starting with *Proper Orthogonal Decomposition* [7]. With the increasing interest in data-driven methods, the effort invested in deriving bounds for entirely data-based models has significantly increased in recent years. For instance, convergence of Extended Dynamic Mode Decomposition towards the Koopman operator in the infinite data limit was shown in [30], bounds for the prediction error of models based on Dynamic Mode Decomposition using finite data were presented in [31, 32], and probabilistic error bounds for the approximation of linear systems via finite data in [33]. Seeing the ever increasing effort, it is likely that more progress will be made in the near future, and one of the main goals of QuaSiModO is to exploit these advances for control in a systematic way. To this end, we assume in the following that an error bound for the deviation over a time step Δt of the following form is known:

$$\left\| f(\Phi_{u^j}(y_0)) - \Phi^r_{u^j}(\bar{z}_0) \right\|_\infty \leq E(\left\| f(y_0) - \bar{z}_0 \right\|, \Delta t).$$

Thus, we obtain the time-dependent model error E_{model}:

$$E_{\mathsf{model}}(t_0) = \left\| f(y_0) - \bar{z}_0 \right\|, \qquad E_{\mathsf{model}}(t_i) \leq E(E_{\mathsf{model}}(t_{i-1}), \Delta t). \tag{2}$$

Remark 1.1. *In the numerical experiments regarding the error bound (Sec. 2.4), we will give a more explicit formula for the specific case of full state observables.*

1.3. Optimization:

We now have three opportunities to calculate the solution of the surrogate-based Problem (III). First, we can directly solve it (for instance, using Dynamic Programming) which – due to the combinatorial nature – is only viable for a small number of control parameters. For larger problems, it is much more advisable to use a relaxation approach, which yields another continuous problem that can be solved efficiently using methods from nonlinear constrained optimization [35]. The second and third approach thus rely on relaxation, and we can either directly apply the obtained solution to the real system or – as this is only viable for control affine systems – we can use the sum up rounding algorithm from [34]. This way, a control corresponding to one of the quantized inputs is applied to the real system. On the one hand, this limits the control freedom (which – as we show in Section 2 – is acceptable for a wide range of problems). On the other hand, the real system is exclusively actuated by inputs that are already contained in V, which allows for a very easy implementation of online learning of the individual surrogate models. This can be extremely beneficial for the control performance, cf., e.g., [24].

In the relaxation approach, we first introduce a new control variable ω that yields a formulation equivalent to (III):

$$\begin{aligned} \min_{\omega\in(\{0,1\}^m)^p} J^r(z) &= \min_{\omega\in(\{0,1\}^m)^p} \sum_{i=0}^{p-1} P^r(z_{i+1}) \\ \text{s.t.} \quad z_{i+1} = \Phi^r(z_i,\omega_i) = \sum_{j=1}^{m} \omega_{i,j}\Phi^r_{u^j}(z_i), \quad &\text{and} \quad \sum_{j=1}^{m} \omega_{i,j} = 1, \qquad i = 0,1,2,\dots. \end{aligned} \tag{III-ω}$$

The last condition ensures that exactly one of the right-hand sides is applied in each time step, i.e., $\sum_{j=1}^{m} \omega_{i,j}u^j = u_i \in V$ for $i = 0,\dots,p-1$.

Problem (III-ω) can now be relaxed by replacing $\omega_i \in \{0,1\}$ with $\alpha_i \in [0,1]$, which is again a continuous optimal control problem:

$$\begin{aligned} \min_{\alpha\in([0,1]^m)^p} J^r(z) &= \min_{\alpha\in([0,1]^m)^p} \sum_{i=0}^{p-1} P^r(z_{i+1}) \\ \text{s.t.} \quad z_{i+1} = \Phi^r(z_i,\alpha_i) = \sum_{j=1}^{m} \alpha_{i,j}\Phi^r_{u^j}(z_i) \quad &\text{and} \quad \sum_{j=1}^{m} \alpha_{i,j} = 1, \qquad i = 0,1,2,\dots. \end{aligned} \tag{IV}$$

The main differences to Problem (I) are the additional condition on α and the fact that the dimension of the control variable is changed by a factor of $(m-1)/\mathsf{dim}(U)$ (the last entry of α_i can always be computed from the condition $\sum_{j=1}^{m} \alpha_{i,j} = 1$). To solve (IV), efficient methods from nonlinear optimization such as gradient descent or BFGS methods [35] can be used.

The final step of the optimization is now to construct the control u^* from the optimal solution α^*. To this end, we proceed according to the SUR procedure for mixed integer problems as proposed in [34], where α^* is transformed back to ω^* while taking past rounding decisions into account:

$$\hat{\omega}_{i,j} = \sum_{k=0}^{i} \alpha^*_{k,j} - \sum_{k=0}^{i-1} \omega^*_{k,j} \tag{3a}$$

$$\omega^*_{i,j} = \begin{cases} 1, & \hat{\omega}_{i,j} \geq \hat{\omega}_{i,l} \;\forall\, l \neq j \quad \text{and} \quad j < l \;\forall\, l \text{ with } \hat{\omega}_{i,j} = \hat{\omega}_{i,l}, \\ 0, & \text{else.} \end{cases} \tag{3b}$$

$$u^*_i = \sum_{j=1}^{m} \omega^*_{i,j} u^j, \qquad i = 0,1,2,\dots. \tag{3c}$$

For control-affine systems and convex sets U, an alternative, straightforward approach is to simply calculate the convex combination of the individual $u^j \in V$:

$$u^*_i = \sum_{j=1}^{m} \alpha^*_{i,j} u^j, \qquad i = 0,1,2,\dots.$$

A comparison between both approaches is made for the Lorenz system as well as for the cylinder flow. Not surprisingly – and as we will discuss in depth in the following section – the two solutions are very close to one another in the control-affine case.

As the focus of this article is not to efficiently solve the resulting optimization problems, we simply use the standard optimizer from the *SciPy* library, i.e., "SLSQP" [36] or "trust-constr" [37] with finite difference approximations of the derivatives. We note that significant speedups are very likely possible by using more efficient solvers tailored to the problems, as well as by providing explicit derivative information. However, the latter requires the derivative of the flow map Φ^r and thus, model-specific knowledge. For neural networks, for instance, these gradients are often easily accessible via algorithmic differentiation.

2. Performance guarantees

In this section, we derive an error bound for the QuaSiModO approach which is composed of the quantization error, the model error and the error caused by the SUR procedure. Since the model error depends on the chosen model, we will assume that E_{model} (cf. Eq. (2)) is known and concentrate on the quantization error which needs to be combined with the model error afterwards. Note that even though we have introduced discrete-time systems, we will derive the bounds for continuous-time systems for ease of notation and in accordance with [34]. The discrete-time version follows as a special case.

2.1. Similarity of trajectories

We begin with the observation that the optimal solutions of (I) and (II) can get arbitrarily close. Obviously, the optimal value $J^*_{(\mathrm{I})}$ of (I) is always at least as small as the optimal value $J^*_{(\mathrm{II})}$ of (II). For the other direction, we will show that for each continuous control u, $u(t) \in U$, we can construct a sequence of discrete controls v_n, $v_n(t) \in V$, which leads to trajectories converging to the trajectory induced by u. In [38] it was already stated that this holds if the set of points in the state space that can be reached with control inputs from U is a subset of the convex hull of the points reachable with control inputs in V. We will prove the same, but in a constructive way in order to obtain concrete error bounds if this condition is not satisfied. Additionally – and more importantly – we are not in the limit case, i.e., a discretization error is introduced which depends on the switching time Δt in (II).

To estimate the distance between the trajectories given by different time-T-maps and different controls, the following result will be useful. A similar idea is used, for instance, in [34, 39] to derive error bounds for the SUR algorithm. Unless otherwise specified, we denote by $\|\cdot\|$ the maximum norm $\|\cdot\|_\infty$ in the following.

Lemma 2.1. *Let $g, \bar{g} : Y \times U \to Y$ and $u, \bar{u} : [0,T] \to U$ be measurable functions with $Y \subseteq \mathbb{R}^{n_y}$ and $U \subseteq \mathbb{R}^{n_u}$. Furthermore, $g(y(\cdot), u(\cdot))$ and $\bar{g}(\bar{y}(\cdot), \bar{u}(\cdot)) \in L^1((0,T), Y)$, where $y(\cdot)$ and $\bar{y}(\cdot)$ are given by*

$$y(t) = y_0 + \int_0^t g(y(\tau), u(\tau))\, d\tau \qquad \text{and} \qquad \bar{y}(t) = \bar{y}_0 + \int_0^t \bar{g}(\bar{y}(\tau), \bar{u}(\tau))\, d\tau,$$

with $y_0, \bar{y}_0 \in Y$. Assume that $\bar{g}$ is Lipschitz continuous in the first argument with Lipschitz constant $L_{\bar{g}}$. If

$$\sup_{t\in[0,T]} \left\| \int_0^t g(y(\tau), u(\tau)) - \bar{g}(y(\tau), \bar{u}(\tau))\, d\tau \right\| \leq M$$

then

$$\|y(t) - \bar{y}(t)\| \leq (M + \|y_0 - \bar{y}_0\|) e^{L_{\bar{g}} t} \quad \forall t \in [0,T].$$

Proof. Let t be in $[0,T]$. Then,

$$\begin{aligned}
\|y(t) - \bar{y}(t)\| &= \left\| y_0 - \bar{y}_0 + \int_0^t g(y(\tau), u(\tau)) - \bar{g}(\bar{y}(\tau), \bar{u}(\tau))\, d\tau \right\| \\
&\leq \|y_0 - \bar{y}_0\| + \left\| \int_0^t g(y(\tau), u(\tau)) - \bar{g}(y(\tau), \bar{u}(\tau))\, d\tau \right\| \\
&\quad + \left\| \int_0^t \bar{g}(y(\tau), \bar{u}(\tau)) - \bar{g}(\bar{y}(\tau), \bar{u}(\tau))\, d\tau \right\| \\
&\leq \|y_0 - \bar{y}_0\| + M + \int_0^t \|\bar{g}(y(\tau), \bar{u}(\tau)) - \bar{g}(\bar{y}(\tau), \bar{u}(\tau))\|\, d\tau \\
&\leq \|y_0 - \bar{y}_0\| + M + L_{\bar{g}} \int_0^t \|y(\tau) - \bar{y}(\tau)\|\, d\tau.
\end{aligned}$$

We can now apply Grönwall's lemma and obtain

$$\|y(t) - \bar{y}(t)\| \leq (M + \|y_0 - \bar{y}_0\|) \cdot e^{L_{\bar{g}} t} \quad \forall t \in [0, T].$$

□

In the MPC context, $y_0 = \bar{y}_0$ is the current state and we obtain a bound for the distance between two trajectories corresponding to the two MPC problems (I) and (II). Note, that the given bound M only has to hold for the optimal trajectory of the original MPC problem (I), not over the entire space.

In order to prove an error bound between the solutions of (I) and (II), according to Lemma 2.1, we need to construct a control v with $v(t) \in V$ for the optimal solution u^* of (I) and derive the following bound:

$$\sup_{t \in [0,T]} \left\| \int_0^t g(y^*(\tau), u^*(\tau)) - g(y^*(\tau), v(\tau))\, d\tau \right\| \leq M.$$

To this end, we will use the idea of relaxation and SUR mentioned before, cf. Eqs. (III-ω) and (IV).

2.2. Bound without model error

We begin without using a surrogate model, i.e., $z = f(y) = y$ and $\Phi^r = \Phi$, and the error bound is derived in two steps. First, we prove that for every feasible solution of our original problem (I), a feasible solution of the relaxed convexification problem (IV) exists that leads to the same trajectory under some assumptions on the chosen subset V. If these assumptions do not hold, we can give an error bound instead. Second, we use results from [34] to show that we obtain a trajectory of the convexified binary problem (III-ω) (which is equivalent to (II) in this setting) that is arbitrarily close to the solution of (IV) by using SUR.

Lemma 2.2. *Let $U \subseteq \mathbb{R}^{n_u}$ be bounded and $V = \{u^1, \dots, u^m\} \subseteq U$ a finite subset. Furthermore, let $g : Y \times U \to Y$ and $y : [0,T] \to Y$ be continuous and $u : [0,T] \to U$ be measurable. Then, there exists a measurable function $\alpha : [0,T] \to [0,1]^m$ such that $\sum_{j=1}^m \alpha_j(t) = 1$ $\forall t \in [0,T]$ and*

$$\sup_{t \in [0,T]} \left\| \int_0^t g(y(\tau), u(\tau)) - \sum_{j=1}^m g(y(\tau), u^j)\alpha_j(\tau)\ d\tau \right\| \leq T \cdot D =: M_1,$$

where D is the maximal distance between the reachable set corresponding to U and the convex hull of the reachable set corresponding to V, i.e.,

$$D = \sup_{t \in [0,T]} \operatorname{dist}(g(y(t), U), \operatorname{Conv}(g(y(t), V)).$$

Proof. For every $t \in [0,T]$, let $y_c(t)$ be the element in $\operatorname{Conv}(g(y(t), V))$ which is closest to $g(y(t), u(t))$. We can write $y_c(t)$ as a convex combination, i.e.,

$$y_c(t) = \sum_{j=1}^m g(y(t), u^j)\alpha_j(t),$$

with $\alpha(t) \in [0,1]^m$ and $\sum_{j=1}^m \alpha_j(t) = 1$. Note, that it is possible to choose $y_c(t)$ and $\alpha(t)$ such that $t \mapsto \alpha(t)$ is measurable. Therefore, it holds for every $t \in [0,T]$

$$\begin{aligned}
&\sup_{t \in [0,T]} \left\| \int_0^t g(y(\tau), u(\tau)) - \sum_{j=1}^m g(y(\tau), u^j)\alpha_j(\tau)\ d\tau \right\| \\
\leq\ &\sup_{t \in [0,T]} \int_0^t \left\| g(y(\tau), u(\tau)) - \sum_{j=1}^m g(y(\tau), u^j)\alpha_j(\tau) \right\| d\tau \\
\leq\ &T \cdot \sup_{t \in [0,T]} \left\| g(y(t), u(t)) - \sum_{j=1}^m g(y(t), u^j)\alpha_j(t) \right\| \\
\leq\ &T \cdot \underbrace{\sup_{t \in [0,T]} \operatorname{dist}(g(y(t), U), \operatorname{Conv}(g(y(t), V))}_{<\infty, \text{ due to continuity of } g \text{ and } y \text{ and the boundedness of } U} = M_1.
\end{aligned}$$

□

This means that if V is chosen such that we can reach the extreme points of the reachable set corresponding to U with the controls $u^j \in V$, we obtain $M_1 = 0$. Next, we can use the idea of the SUR algorithm to estimate the error between the relaxed and the discrete control. Therefore, we will use the following result.

Theorem 2.3. *Let $g : Y \times U \to Y$, $y : [0,T] \to Y$ and $u : [0,T] \to U$ be measurable functions and assume that $\omega : [0,T] \to [0,1]^m$ is constructed from α via SUR (cf. Eq. (3) or [34] with the time discretization Δt, i.e., it holds*

$$\left\| \int_0^t \alpha(\tau) - \omega(\tau) \right\| \leq (m-1)\Delta t \quad \forall t \in [0,T].$$

Furthermore, assume that $g(y(\cdot), u^j)$ is differentiable for almost all $t \in [0,T]$ and that constants C_1 and $C_2 \in \mathbb{R}$ exist for all $u^j \in V$ such that for almost all $t \in [0,T]$:

$$\left\| \frac{d}{dt} g(y(t), u^j) \right\| \leq C_1 \qquad \text{and} \qquad \left\| g(y(t), u^j) \right\| \leq C_2.$$

Then

$$\sup_{t \in [0,T]} \left\| \int_0^t \sum_{j=1}^m g(y(\tau), u^j)(\alpha_j(\tau) - \omega_j(\tau))\, d\tau \right\| \leq (C_2 + C_1 T)(m-1)\Delta t =: M_2(\Delta t).$$

Proof. The proof can be found in [34]. □

Remark 2.4. *A similar result can be found in [39] but with weaker assumptions on g. There, the authors prove that for $g(y(\cdot), v) \in L^1((0,T), Y)$,*

$$\lim_{\Delta t \to 0} \sup_{t \in [0,T]} \left\| \int_0^t \sum_{j=1}^m g(y(\tau), u^j)(\alpha_j(t) - \omega_j(\tau))\, d\tau \right\| = 0 \qquad \text{if} \qquad \lim_{\Delta t \to 0} \left\| \int_0^t \alpha(\tau) - \omega(\tau)\, d\tau \right\| = 0.$$

Furthermore, they show that the statement holds not only for ODEs, but for semilinear PDEs as well. Here, we will use the result from [34], as we want to use the concrete error bound.

Using the above results, we can now ensure that continuous and discrete inputs yield trajectories that are close.

Theorem 2.5. *Let $U \subseteq \mathbb{R}^{n_u}$ be bounded and $V = \{u^1, \dots, u^m\} \subseteq U$ be a finite subset. Assume $g : Y \times U \to Y$ is continuous, $u : [0,T] \to U$ is measurable and $y : [0,T] \to Y$ is defined by*

$$y(t) = y_0 + \int_0^t g(y(\tau), u(\tau))\, d\tau, \quad y_0 \in Y.$$

Furthermore, let $g(y(t), u^j)$ be differentiable with respect to time for almost all $t \in [0,T]$ and assume that constants C_1 and $C_2 \in \mathbb{R}$ exist for all $u^j \in V$ such that for almost all $t \in [0,T]$:

$$\left\| \frac{d}{dt} g(y(t), u^j) \right\| \leq C_1 \qquad \text{and} \qquad \left\| g(y(t), u^j) \right\| \leq C_2.$$

In addition, assume that g is Lipschitz continuous in the second argument with Lipschitz constant L_g. Then, for every $\varepsilon > 0$, there exists a discrete control function $\bar{u} : [0,T] \to V$, such that for $\bar{y}$ given by

$$\bar{y}(t) = \bar{y}_0 + \int_0^t g(\bar{y}(\tau), \bar{u}(\tau))\, d\tau, \quad \bar{y}_0 \in Y,$$

it holds

$$\|y(t) - \bar{y}(t)\| \leq (M_1 + \varepsilon + \|y_0 - \bar{y}_0\|) \cdot e^{L_g t} \quad \forall t \in [0,T].$$

Proof. Lemma 2.2 ensures the existence of a measurable function $\alpha : [0,T] \to [0,1]$ with $\sum_{j=1}^{m} \alpha_j(t) = 1 \ \forall t \in [0,T]$ and

$$\sup_{t\in[0,T]} \left\| \int_0^t g(y(\tau),u(\tau)) - \sum_{j=1}^{m} g(y(\tau),u^j)\alpha_j(\tau)\, d\tau \right\| \leq M_1, \quad \forall t \in [0,T].$$

Using SUR, we can construct (according to Theorem 2.3) a function $\omega : [0,T] \to \{0,1\}$ with $\sum_{j=1}^{m} \omega_j(t) = 1 \ \forall t \in [0,T]$ from α, such that

$$\sup_{t\in[0,T]} \left\| \int_0^t \sum_{j=1}^{m} g(y(\tau),u^j)(\alpha_j(\tau) - \omega_j(\tau))\, d\tau \right\| \leq M_2(\Delta t),$$

with $M_2(\Delta t) = (C_2 + C_1 T)(m-1)\Delta t$. Therefore, we get

$$\sup_{t\in[0,T]} \left\| \int_0^t g(y(\tau),u(\tau)) - \sum_{j=1}^{m} g(y(\tau),u^j)\omega_j(\tau)\, d\tau \right\| \leq M_1 + M_2(\Delta t).$$

Since $\bar{u}(t) := \sum_{j=1}^{m} \omega_j(\tau) u^j \in V$ for all $t \in [0,T]$, choosing Δt sufficiently small yields:

$$\sup_{t\in[0,T]} \left\| \int_0^t g(y(\tau),u(\tau)) - g(y(\tau),\bar{u}(t))\, d\tau \right\| \leq M_1 + \varepsilon,$$

and using Lemma 2.1, we obtain the desired result. □

Remark 2.6. *The m in $M_2(t)$ in Theorem 2.5 can be reduced to the number of elements in V that are actually required in the convex combinations to represent/approximate $u(t)$ over the prediction horizon T, i.e., m is at most $n_y + 1$.*

Finally, with respect to the control problems (I) and (II), we can derive a relation between the optimal values of Problems (I) and (II).

Corollary 2.7. *Let (y^*, u^*) be an optimal solution of (I) where $U \subseteq \mathbb{R}^{n_u}$ is bounded and $P : Y \to \mathbb{R}$ Lipschitz continuous with Lipschitz constant L_P for a fixed initial value $y_0 \in Y$. Assume $g : Y \times U \to Y$ and $V \subseteq U$ are of a form such that the requirements of Theorem 2.5 are satisfied. Then, there exists a tuple $(\bar{y}, \bar{u})$ with $\bar{u}(t) \in V$ which is feasible for (II) with the same initial value y_0 such that*

$$|J(y^*) - J(\bar{y})| \leq L_P(M_1 + M_2(\Delta t)) \frac{e^{L_g \Delta t}\left(e^{p L_g \Delta t} - 1\right)}{e^{L_g \Delta t} - 1}.$$

Proof. First, we construct $\bar{u}$. Therefore, α is chosen as in Lemma 2.2 and ω is constructed via SUR from α, i.e.,

$$y^*_{i+1} = \Phi(y^*_i, u^*_i) = \sum_{j=1}^{m} \alpha_{i,j} \Phi_{u^j}(y^*_i) \text{ and } \bar{u}_i := \sum_{j=1}^{m} \omega_{i,j} u^j.$$

We obtain directly from Theorem 2.5 and the Lipschitz continuity of P:

$$\begin{aligned} |J(y^*) - J(\bar{y})| &\leq \sum_{i=0}^{p-1} L_P \left\| y^*_{i+1} - \bar{y}_{i+1} \right\| \leq \sum_{i=0}^{p-1} L_P(M_1 + M_2(\Delta t)) e^{L_g t_{i+1}} \\ &= \underbrace{L_P(M_1 + M_2(\Delta t)) \frac{e^{L_g \Delta t}\left(e^{p L_g \Delta t} - 1\right)}{e^{L_g \Delta t} - 1}}_{E_{\mathsf{V}}(V) + E_{\mathsf{MI}}(\Delta t)}. \end{aligned}$$

□

Remark 2.8. *The errors*

$$E_{\mathsf{V}}(V) = L_P M_1 \frac{e^{L_g \Delta t}\left(e^{pL_g\Delta t}-1\right)}{e^{L_g\Delta t}-1} \quad \text{and} \quad E_{\mathsf{MI}}(\Delta t) = L_P M_2(\Delta t)\frac{e^{L_g \Delta t}\left(e^{pL_g\Delta t}-1\right)}{e^{L_g\Delta t}-1} \tag{4}$$

account for the distance between the reachable sets of V and U and the transformation into a mixed integer problem and the corresponding relaxation technique, respectively. For an appropriate choice of V, we have $M_1 = 0$. Moreover, we have

$$\lim_{\Delta t\to 0} \frac{e^{L_g \Delta t}\left(e^{pL_g\Delta t}-1\right)}{e^{L_g\Delta t}-1} = p \qquad \text{and} \qquad \lim_{\Delta t\to 0} M_2(\Delta t) = 0,$$

such that both errors can become arbitrarily small using the appropriate numerical setup.

Remark 2.9. *An error bound can also be obtained if P (and J) explicitly depend on the control u. In this case, however, an additional term needs to be added that pessimistically bounds the distance between the optimal controls for the full and surrogate-based problems. This bound also depends on the size of the control set U.*

2.3. Combination with model error

We now consider the additional errors resulting from surrogate models (Eq. (2)), i.e., we solve Problem (III) with the assumption that J and J^r are equivalent, cf. Eq. (1). To do so, there are several ways in practice, and depending on the solution strategy, we obtain different error bounds. One can immediately see (via the triangle inequality) that the different error sources are additive in all cases. For ease of notation, we introduce the *control-to-state operators* $S : U^p \to Y^p$ and $S^r : U^p \to f(Y)^p$ which – for a fixed $y_0 \in Y$ – map the control inputs $(u_0, \dots, u_{p-1})$ to the corresponding states $(y_1, \dots, y_p)$ and $(z_1, \dots, z_p)$, respectively.

2.3.1. Nonlinear observable functions

The obvious approach is to solve problem (III) directly, for instance using Dynamic Programming or a total evaluation of all possible combinations of control inputs. If we have an appropriate (i.e., sufficiently small) set V, this can be very efficient. In this case, we need to bound the difference $\left|J(S(u^*_{(\mathrm{I})})) - J(S(u^*_{(\mathrm{III})}))\right|$, where $u^*_{(\mathrm{I})}$ and $u^*_{(\mathrm{III})}$ are the optimal solutions of (I) and (III), respectively. Since we already have a bound for $\left|J(S(u^*_{(\mathrm{I})})) - J(S(u^*_{(\mathrm{II})}))\right|$, it is sufficient to determine the error $\left|J(S(u^*_{(\mathrm{II})})) - J(S(u^*_{(\mathrm{III})}))\right|$. Therefore, we first consider the error between $J(S(u))$ and $J^r(S^r(u))$ for arbitrary $u \in V^p$:

$$\begin{aligned}
|J(S(u)) - J^r(S^r(u))| &= |J^r(f(S(u))) - J^r(S^r(u))| = \left|\sum_{i=0}^{p-1} P^r(f((S(u))_i)) - P^r((S^r(u))_i)\right| \\
&\le \sum_{i=0}^{p-1} L_P \left\|f((S(u))_i) - (S^r(u))_i\right\| = \sum_{i=0}^{p-1} L_P \left\|f\left(\Phi_{u_i}(y_i)\right) - \Phi^r_{u_i}(y^r_i)\right\| \\
&\le L_P \sum_{i=1}^{p} E_{\mathsf{model}}(t_i),
\end{aligned} \tag{5}$$

where the first equality is due to (1). For ease of notation, we do not distinguish between the Lipschitz constants of P and P^r and just use maximum of the two as L_P. Now, it holds

$$\begin{aligned}
&\left|J(S(u^*_{(\mathrm{III})})) - J(S(u^*_{(\mathrm{II})}))\right| = J(S(u^*_{(\mathrm{III})})) - J(S(u^*_{(\mathrm{II})})) \\
&= \underbrace{J(S(u^*_{(\mathrm{III})})) - J^r(S^r(u^*_{(\mathrm{III})}))}_{\le \left|J(S(u^*_{(\mathrm{III})})) - J^r(S^r(u^*_{(\mathrm{III})}))\right|} + \underbrace{\left(J^r(S^r(u^*_{(\mathrm{III})})) - J^r(S^r(u^*_{(\mathrm{II})}))\right)}_{\le 0, \text{ since } u^*_{(\mathrm{III})} \text{ is a global Min. of } J^r(S^r(\cdot))} + \underbrace{J^r(S^r(u^*_{(\mathrm{II})})) - J(S(u^*_{(\mathrm{II})}))}_{\le \left|J^r(S^r(u^*_{(\mathrm{II})})) - J(S(u^*_{(\mathrm{II})}))\right|} \\
&\le 2L_P \sum_{i=0}^{p} E_{\mathsf{model}}(t_i),
\end{aligned}$$

where we can neglect the absolute value because $u^*_{(\mathrm{II})}$ is optimal with respect to J. In summary, the error bound is thus given by

$$\left|J(S(u^*_{(\mathrm{I})})) - J(S(u^*_{(\mathrm{III})}))\right| \leq E_{\mathsf{V}}(V) + E_{\mathsf{MI}}(\Delta t) + \underbrace{2L_P \sum_{i=0}^{p} E_{\mathsf{model}}(t_i)}_{E_{\mathsf{r}}(E)}, \tag{E1}$$

see Eq. (2) for E_{model} and Eq. (4) for $E_{\mathsf{V}}(V)$ and $E_{\mathsf{MI}}(\Delta t)$.

In many cases, we will have a finite set $V = \{u^1, \dots, u^m\} \subseteq U$ which is too large to solve the combinatorial problem (III) directly. In this case, we can solve the relaxation (IV) in combination with SUR instead, where the resulting control is denoted by $u^*_{(\mathrm{IV})-\mathsf{SUR}}$. The error is composed of the error (E1) and $\left|J(S(u^*_{(\mathrm{III})})) - J(S(u^*_{(\mathrm{IV})-\mathsf{SUR}}))\right|$. Analog to the derivation before, we obtain an error $E_{\mathsf{MI}^r}(\Delta t)$ for the surrogate-based mixed-integer transformation (with variables $M_2^r(\Delta t)$ and L_{g^r} if the surrogate model is given by $\Phi^r(y_i, u_i) = y_i + \int_{t_i}^{t_{i+1}} g^r(y(t), u_i)\, dt$ and g^r is Lipschitz continuous with Lipschitz constant L_{g^r}). Therefore, we get

$$\left|J(S(u^*_{(\mathrm{I})})) - J(S(u^*_{(\mathrm{IV})-\mathsf{SUR}}))\right| \leq E_{\mathsf{V}}(V) + E_{\mathsf{MI}}(\Delta t) + 2E_{\mathsf{r}}(E) + E_{\mathsf{MI}^r}(\Delta t), \tag{E2.a}$$

where the additional second model error $E_{\mathsf{r}}(E)$ is due to the fact that we need to estimate the difference in terms of the true objective.

2.3.2. Linear observable functions

If f is linear (which is in particular the case when considering the full-state observable), an error bound can be obtained in a similar way to Eq. (E1). Therefore, we do a similar computation as in (5) for the relaxed systems, i.e., for the control-to-state operators $\bar{S} : [0,1]^m \to Y$ and $\bar{S}^r : [0,1]^m \to f(Y)$, and get for an arbitrary control α

$$\left|J(\bar{S}(\alpha)) - J^r(\bar{S}^r(\alpha))\right| \leq L_P \sum_{i=1}^{p} E_{\mathsf{model}}(t_i).$$

In this case, we obtain the same bound as for (E1):

$$\left|J(S(u^*_{(\mathrm{I})})) - J(S(u^*_{(\mathrm{IV})-\mathsf{SUR}}))\right| \leq E_{\mathsf{V}}(V) + E_{\mathsf{MI}}(\Delta t) + E_{\mathsf{r}}(E). \tag{E2.b}$$

Finally, the third option is to solve (IV) and directly apply the relaxed solution to the original system. Obviously, this is only feasible if $\mathrm{Conv}(V) \subseteq U$. This way, we introduce an additional error caused by the linear interpolation if the system is not control affine. Nevertheless, many systems are control affine and in this case, we have

$$\left|J(S(u^*_{(\mathrm{I})})) - J\left(S\left(\sum_{j=1}^{m} \alpha^*_{(\mathrm{IV}),j} u^j\right)\right)\right| \leq E_{\mathsf{V}}(V) + E_{\mathsf{r}}(E). \tag{E3}$$

The bounds are summarized in Table 1, together with the additional requirements for the control problem. It should be noted that all errors besides the one for the surrogate model can be made arbitrarily small. We have $E_{\mathsf{V}}(V) = 0$ if the convex hull of the reachable set corresponding to V is a subset of the one corresponding to U, and both $E_{\mathsf{MI}}(\Delta t)$ and $E_{\mathsf{MI}^r}(\Delta t)$ vanish as the switching time Δt tends to zero.

Table 1: Different transformation procedures and corresponding error bounds.

Transformation approach	Error bound	control affine	f linear	Type of optimization
(I) → (II) → (III)	Eq. (E1)	—	—	Combinatorial
(I) → (II) → (III) → (IV) → SUR	Eq. (E2.a)	—	—	Continuous
	Eq. (E2.b)	—	✓	Continuous
(I) → (II) → (III) → (IV)	Eq. (E3)	✓	✓	Continuous

2.4. Example

To study the error bounds numerically, we consider the well-known Duffing oscillator:

$$\dot{y} = g(y,u) = \begin{pmatrix} y_2 \\ -\delta y_2 - \alpha y_1 - \beta y_1^3 \end{pmatrix} + \begin{pmatrix} 0 \\ u \end{pmatrix},$$

with constants $\alpha = -1$, $\beta = 1$, $\delta = 0$, and $u(t) \in U = [-4, 4]$. To introduce a model error, we add a constant perturbation in the second equation, i.e., as the surrogate model we use

$$\dot{y} = g^r(y,u) = \begin{pmatrix} y_2 \\ -\delta y_2 - \alpha y_1 - \beta y_1^3 + \varepsilon \end{pmatrix} + \begin{pmatrix} 0 \\ u \end{pmatrix},$$

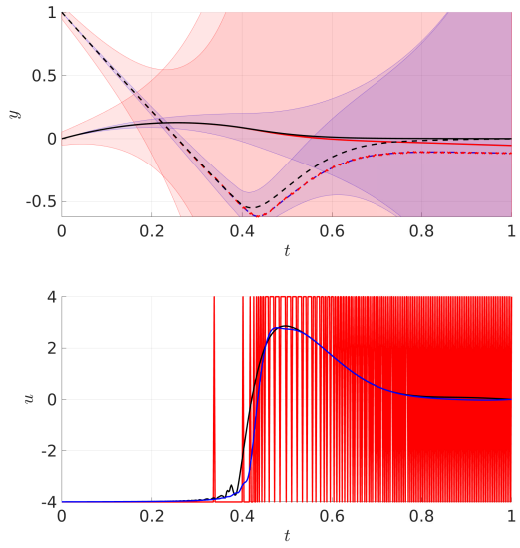
with a fixed $\varepsilon = 10^{-1}$. As the finite set of controls we choose $V = \{-4, 4\}$. To determine the model error $E_{\mathsf{model}}(t_i)$ in (2) we can use Lemma 2.1 since we use the full-state observable. For an arbitrary control $u(t)$ and a trajectory $y(t)$, it holds

$$\sup_{t\in[0,\Delta t]} \left\| \int_0^t g(y(\tau),u(\tau)) - g^r(y(\tau),u(\tau)) \right\| \leq \varepsilon \Delta t$$

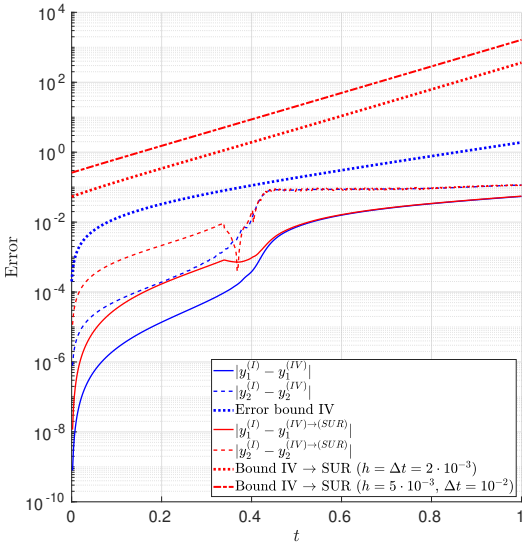
and according to Lemma 2.1, we can estimate the model error via

$$\begin{aligned} E_{\mathsf{model}}(t_{i+1}) &= \left\| y_{i+1} - y^r_{i+1} \right\| \leq (\varepsilon \Delta t + \| y_i - y^r_i \|) e^{L_g \Delta t} \\ &= (\varepsilon \Delta t + E_{\mathsf{model}}(t_i)) e^{L_g \Delta t}, \end{aligned}$$

where y_i and y^r_i are the discrete trajectories defined by g and g^r, respectively, for a given starting point $y_0 \in Y$ and a control sequence $u_i \in V$. The goal is to stabilize the system at $y^{\mathsf{ref}} = (0,0)^\top$, i.e., $J(y) = \sum_{i=0}^{p-1} \|y_{i+1}\|$, and we solve Problems (I), (IV) and (III) (the latter using SUR) to investigate the error bounds (E3) and (E2.b). The control horizon is $[0,1]$, and we use $\Delta t = 2 \cdot 10^{-3}$ for the time-T-maps Φ and Φ^r, respectively, as well as for the sum up rounding.



(a) Optimal state y^* and control u^* of Problem (I) (black). The error margins around the solution and the optimal surrogate-based solutions are shown in blue and red, respectively.



(b) The error between the optimal surrogate-based solutions and the solution to (I). The influence of different values for Δt on (E2.a) is emphasized by the yellow and orange lines, respectively.

Figure 2: Error bounds (E3) (blue) and (E2.a) (red). The solid lines denote the component y_1 and the dashed lines y_2.

The constants C_1 and C_2 that enter the calculation of M_2 are approximated from data using several simulations with random initial conditions. We do the same for the Lipschitz constants, which we estimate via the derivative. As the system is control affine and V consists of u_{min} and u_{max}, we have $M_1 = 0$.

Figure 2 shows the results for the two approaches. We see that both achieve the control task relatively well with a small error due to the constant offset ε in the second component. Moreover, we see that the error bound (E3) is very well suited for the MPC context (where short prediction horizons are very common), and it is much tighter than the SUR approach. On the other hand, we observe in Figure 2b that the two errors come closer with decreasing Δt, as all errors but E_{model} can be made arbitrarily small. Nevertheless, it can be concluded that solving Problem (IV) without rounding is clearly advantageous for control-affine systems.

3. Results

We have tested the QuaSiModO framework on a variety of dynamical systems, observable functions and surrogate modeling techniques, cf. Figure 3, a detailed description of the numerical setup is given in Appendix A for the surrogate modeling and in Appendix B for the test problems. The systems range from the chaotic Lorenz system over the stochastic dynamics of the ongoing COVID-19 pandemic and the dynamics of blood cells (modeled by the Mackey-Glass delay differential equation) to the Navier–Stokes equations for fluid flows. The observed quantities z range from the full state over partial observations and delay coordinates to point-wise measurements, and we use surrogate models based on POD, the Koopman operator and different neural network architectures. For instance, we can control the lift force acting on a cylinder (determined by the velocity and pressure fields governed by the 2D Navier–Stokes equations) without any knowledge of the flow field using the standard LSTM framework included in *TensorFlow*, and stabilize the Mackey-Glass equation using a sandard echo state network. This highlights the flexibility and broad applicability of the method and the success of the technique in constructing data-driven feedback controllers.

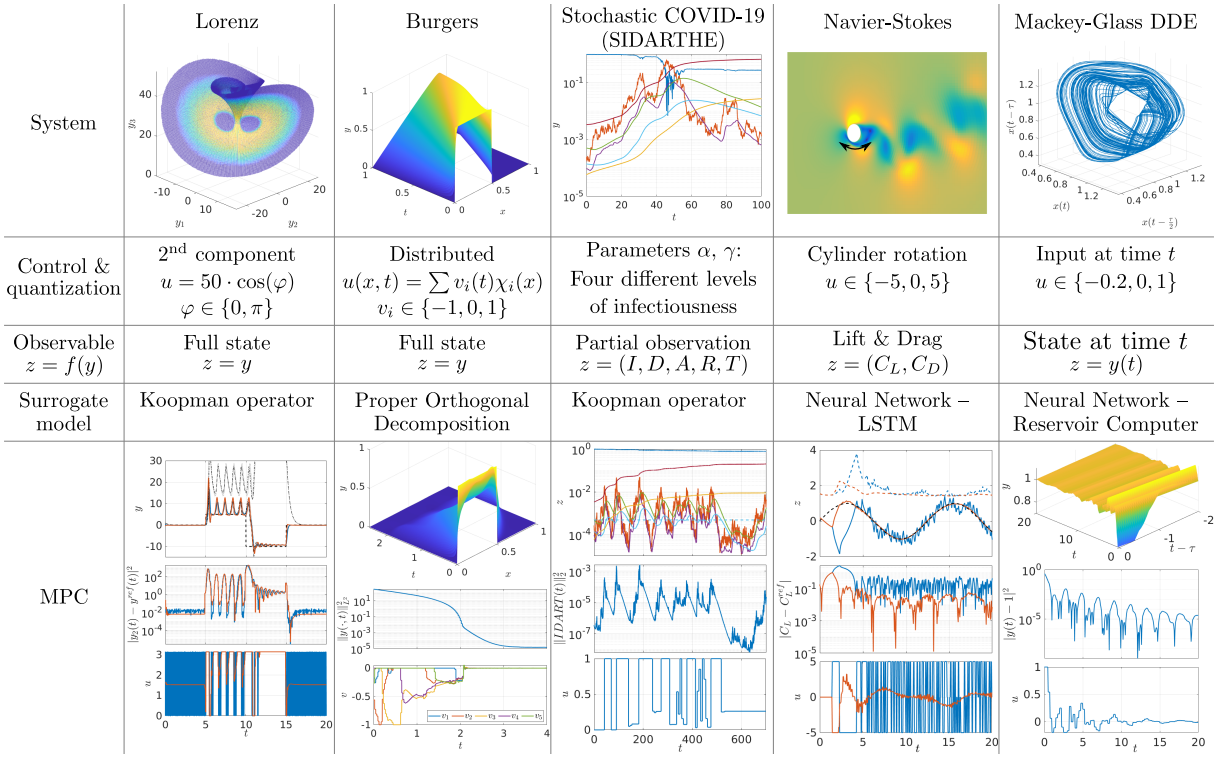


Figure 3: MPC using QuaSiModO applied to various combinations of systems and surrogate models. For the Lorenz and Navier–Stokes example, both the continuous (orange) and the rounded control (blue) are shown. In the remaining examples, we have applied the continuous solution of Problem (IV) to the real system.

4. Conclusion

QuaSiModO is a powerful algorithm for data-driven control of complex systems from many scientific disciplines that does not require knowledge of the underlying dynamics and avoids problem specific modeling efforts. Instead, measurement data corresponding to different fixed inputs can be combined with state-of-the-art surrogate modeling techniques for the prediction of complex dynamical systems in a straightforward manner. We demonstrate excellent control performance on a variety of dynamical systems, using different control inputs, observations, and surrogate modeling techniques, thus showing great flexibility and a wide range of possible applications from problems in engineering, biology, or life sciences. Furthermore, when error bounds are available for the predictive model, these directly translate into error bounds for the corresponding control problem. There is a large number of researchers addressing the issue of error bounds for predictive models constructed from data, see, for instance, [33, 31]. Consequently QuaSiModO will directly benefit from these advancements as well as general improvements in data-driven modeling in the future, and will thus continue to play an important role in the construction of data-driven feedback controllers.

Code

The QuaSiModO toolbox can be obtained freely at https://github.com/SebastianPeitz/QuaSiModO.

Acknowledgments

This research has been supported by the Priority Programme 1962 of the Deutsche Forschungsgemeinschaft (DFG).

References

[1] International Energy Agency. World Energy Outlook 2018. *https://www.iea.org/weo2018*, 2018.

[2] A. Pironti and M. Walker. Fusion, tokamaks, and plasma control: an introduction and tutorial. *IEEE Control Systems Magazine*, 25(5):30–43, 2005.

[3] R. K. Calay, J. Kurujareon, and A. E. Holdø. Numerical simulation of respiratory flow patterns within human lung. *Respiratory Physiology & Neurobiology*, 130(2):201–221, 2002.

[4] P. Crosetto, P. Reymond, S. Deparis, D. Kontaxakis, N. Stergiopulos, and A. Quarteroni. Fluid-structure interaction simulation of aortic blood flow. *Computers and Fluids*, 43(1):46–57, 2011.

[5] H. Mestre, J. Tithof, T. Du, W. Song, W. Peng, A. M. Sweeney, G. Olveda, J. H. Thomas, M. Nedergaard, and D. H. Kelley. Flow of cerebrospinal fluid is driven by arterial pulsations and is reduced in hypertension. *Nature communications*, 9(1):4878, 2018.

[6] G. Giordano, F. Blanchini, R. Bruno, P. Colaneri, A. Di Filippo, A. Di Matteo, and M. Colaneri. Modelling the COVID-19 epidemic and implementation of population-wide interventions in Italy. *Nature Medicine*, 2020.

[7] L. Sirovich. Turbulence and the dynamics of coherent structures part I: coherent structures. *Quarterly of Applied Mathematics*, XLV(3):561–571, 1987.

[8] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[9] P. R. Vlachas, W. Byeon, Z. Y. Wan, T. P. Sapsis, and P. Koumoutsakos. Data-driven forecasting of high-dimensional chaotic systems with long short-Term memory networks. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2213), 2018.

[10] H. Jaeger and H. Haas. Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication. *Science*, 304(5667):78–80, 2004.

[11] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott. Model-Free Prediction of Large Spatiotemporally Chaotic Systems from Data: A Reservoir Computing Approach. *Physical Review Letters*, 120(2):24102, 2018.

[12] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

[13] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3(4), 2017.

[14] C. W. Rowley, I. Mezić, S. Bagheri, P. Schlatter, and D. S. Henningson. Spectral analysis of nonlinear flows. *Journal of Fluid Mechanics*, 641:115–127, 2009.

[15] P. J. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28, 2010.

[16] S. Klus, F. Nüske, S. Peitz, J.-H. Niemann, C. Clementi, and C. Schütte. Data-driven approximation of the Koopman generator: Model reduction, system identification, and control. *Physica D: Nonlinear Phenomena*, 406:132416, 2020.

[17] Z. Y. Wan, P. Vlachas, P. Koumoutsakos, and T. Sapsis. Data-assisted reduced-order modeling of extreme events in complex dynamical systems. *PLoS ONE*, 13(5):1–22, 2018.

[18] J. Kates-Harbeck, A. Svyatkovskiy, and W. Tang. Predicting disruptive instabilities in controlled fusion plasmas through deep learning. *Nature*, 568(7753):526–531, 2019.

[19] C. Sánchez-Sánchez and D. Izzo. Real-time optimal control via deep neural networks: Study on landing problems. *Journal of Guidance, Control, and Dynamics*, 41(5):1122–1135, 2018.

[20] M. P. Deisenroth and C. E. Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 465–472, 2011.

[21] F. L. Lewis, D. Vrabie, and K. G. Vamvoudakis. Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers. *IEEE Control Systems Magazine*, 32(6):76–105, 2012.

[22] M. P. Deisenroth, D. Fox, and C. E. Rasmussen. Gaussian Processes for Data-Efficient Learning in Robotics and Control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):408–423, 2015.

[23] K. Kunisch and S. Volkwein. Control of the Burgers Equation by a Reduced-Order Approach Using Proper Orthogonal Decomposition. *Journal of Optimization Theory and Applications*, 102(2):345–371, 1999.

[24] K. Bieker, S. Peitz, S. L. Brunton, J. N. Kutz, and M. Dellnitz. Deep model predictive flow control with limited sensor data and online learning. *Theoretical and Computational Fluid Dynamics*, 34:577–591, 2020.

[25] M. Bergmann, L. Cordier, and J.-P. Brancher. Optimal rotary control of the cylinder wake using proper orthogonal decomposition reduced-order model. *Physics of Fluids*, 17:1–21, 2005.

[26] J. L. Proctor, S. L. Brunton, and J. N. Kutz. Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems*, 15(1):142–161, 2015.

[27] S. L. Brunton and J. N. Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2019.

[28] S. Peitz and S. Klus. Koopman operator-based model reduction for switched-system control of PDEs. *Automatica*, 106:184 – 191, 2019.

[29] S. Peitz, S. E. Otto, and C. W. Rowley. Data-Driven Model Predictive Control using Interpolated Koopman Generators. *SIAM Journal on Applied Dynamical Systems*, 19(3):2162–2193, 2020.

[30] M. Korda and I. Mezić. On Convergence of Extended Dynamic Mode Decomposition to the Koopman Operator. *Journal of Nonlinear Science*, 28(2):687–710, 2018.

[31] H. Lu and D. M. Tartakovsky. Predictive Accuracy of Dynamic Mode Decomposition. *SIAM Journal on Scientific Computing*, 42(3):1639–1662, 2020.

[32] S. Klus, F. Nüske, and B. Hamzi. Kernel-based approximation of the koopman generator and Schrödinger operator. *Entropy*, 22(7), 2020.

[33] M. Simchowitz, H. Mania, S. Tu, M. I. Jordan, and B. Recht. Learning without mixing: Towards a sharp analysis of linear system identification. In *31st Annual Conference on Learning Theory*, volume 75, pages 1–35, 2018.

[34] S. Sager, H. G. Bock, and M. Diehl. The integer approximation error in mixed-integer optimal control. *Mathematical Programming*, 133(1-2):1–23, 2012.

[35] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer Science & Business Media, 2. edition, 2006.

[36] D. Kraft et al. A software package for sequential quadratic programming. In *https://docs.scipy.org/doc/scipy/reference/optimize.minimize-slsqp.html*. DFVLR Oberpfaffenhofen, Germany, 1988.

[37] R. H. Byrd, M. E. Hribar, and J. Nocedal. An interior point algorithm for large-scale nonlinear programming. *SIAM Journal on Optimization*, 9(4):877–900, 1999.

[38] T. Ważewski. On an optimal control problem. *Differential equations and their applications*, pages 229–242, 1963.

[39] P. Manns and C. Kirches. Improved regularity assumptions for partial outer convexification of mixed-integer pde-constrained optimization problems. *ESAIM: COCV*, 26:32, 2020.

[40] P. Holmes, J. L. Lumley, G. Berkooz, and C. W. Rowley. *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*. Cambridge University Press, 2 edition, 2012.

[41] P. Benner, S. Gugercin, and K. Willcox. A Survey of Projection-Based Model Reduction Methods for Parametric Dynamical Systems. *SIAM Review*, 57(4):483–531, 2015.

[42] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz. On Dynamic Mode Decomposition: Theory and Applications. *Journal of Computational Dynamics*, 1(2):391–421, 2014.

[43] S. Klus, P. Gelß, S. Peitz, and C. Schütte. Tensor-based dynamic mode decomposition. *Nonlinearity*, 31(7):3359–3380, 2018.

[44] A. J. Chorin and F. Lu. Discrete approach to stochastic parametrization and dimension reduction in nonlinear dynamics. *Proceedings of the National Academy of Sciences*, 112(32):9804–9809, 2015.

[45] A. J. Chorin, O. H. Hald, and R. Kupferman. Optimal prediction and the mori–zwanzig representation of irreversible processes. *Proceedings of the National Academy of Sciences*, 97(7):2968–2973, 2000.

[46] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel. *Time series analysis: forecasting and control*, volume 734. John Wiley & Sons, 2011.

[47] Z. Tang and P. A. Fishwick. Feedforward neural nets as models for time series forecasting. *ORSA Journal on Computing*, 5(4):374–385, 1993.

[48] I. Mezić. Analysis of Fluid Flows via Spectral Properties of the Koopman Operator. *Annual Review of Fluid Mechanics*, 45:357–378, 2013.

[49] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley. A Data-Driven Approximation of the Koopman Operator: Extending Dynamic Mode Decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, 2015.

[50] A. Lasota and M. C. Mackey. *Chaos, fractals, and noise: Stochastic aspects of dynamics*, volume 97 of *Applied Mathematical Sciences*. Springer, 2nd edition, 1994.

[51] M. Budišić, R. Mohr, and I. Mezić. Applied Koopmanism. *Chaos*, 22, 2012.

[52] S. Klus, P. Koltai, and C. Schütte. On the numerical approximation of the Perron-Frobenius and Koopman operator. *Journal of Computational Dynamics*, 3(1):51–79, 2016.

[53] M. Hinze, R. Pinnau, M. Ulbrich, and S. Ulbrich. *Optimization with PDE Constraints*. Springer Science+Business Media, 2009.

[54] S. Chaturantabut and D. C. Sorensen. Nonlinear Model Reduction via Discrete Empirical Interpolation. *SIAM Journal on Scientific Computing*, 32(5):2737–2764, 2010.

[55] M. Lukosevicius and H. Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127 – 149, 2009.

[56] H. Jaeger. Short term memory in echo state networks. *GMD Technical Report*, 152, 2002.

[57] H. Jaeger. The "echo state" approach to analysing and training recurrent neural networks – with an erratum note. *GMD Technical Report*, 148, 01 2001.
[58] D. Li, M. Han, and J. Wang. Chaotic time series prediction based on a novel robust echo state network. *IEEE Transactions on Neural Networks and Learning Systems*, 23(5):787–799, 2012.
[59] P. L. McDermott and C. K. Wikle. An ensemble quadratic echo state network for nonlinear spatio-temporal forecasting. *arXiv:1708.05094*, 2017.
[60] Ashesh Chattopadhyay, Pedram Hassanzadeh, and Devika Subramanian. Data-driven predictions of a multiscale lorenz 96 chaotic system using machine-learning methods: reservoir computing, artificial neural network, and long short-term memory network. *Nonlinear Processes in Geophysics*, 27(3):373–389, Dec 2019.
[61] J. Pathak, Z. Lu, B. R. Hunt, M. Girvan, and E. Ott. Using machine learning to replicate chaotic attractors and calculate lyapunov exponents from data. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(12):121102, 2017.
[62] Z. Lu, B. R. Hunt, and E. Ott. Attractor reconstruction by machine learning. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 28(6):061104, 2018.
[63] L. B. Armenio, E. Terzi, M. Farina, and R. Scattolini. Model predictive control design for dynamical systems learned by echo state networks. *IEEE Control Systems Letters*, 3(4):1044–1049, 2019.
[64] J. P. Jordanou, E. Camponogara, E. A. Antonelo, and M. A. Schmitz Aguiar. Nonlinear model predictive control of an oil well with echo state networks. *IFAC-PapersOnLine*, 51(8):13–18, 2018. 3rd IFAC Workshop on Automatic Control in Offshore Oil and Gas Production OOGP 2018.
[65] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, 2016.
[66] E. Kaiser, J. N. Kutz, and S. L. Brunton. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2219):20180335, 2018.
[67] J. Köhler, L. Schwenkel, A. Koch, J. Berberich, P. Pauli, and F. Allgöwer. Robust and optimal predictive control of the COVID-19 outbreak. *arXiv:2005.03580*, 2020.
[68] B. R. Noack, K. Afanasiev, M. Morzynski, G. Tadmor, and F. Thiele. A hierarchy of low-dimensional models for the transient and post-transient cylinder wake. *Journal of Fluid Mechanics*, 497:335–363, 2003.
[69] M. C. Mackey and L. Glass. Oscillation and chaos in physiological control systems. *Science*, 197(4300):287–289, 1977.
[70] L. Glass and M. C. Mackey. Pathological conditions resulting from instabilities in physiological control systems. *Annals of the New York Academy of Sciences*, 316(1):214–235, 1979.
[71] J. D. Farmer and J. J. Sidorowich. Predicting chaotic time series. *Physical review letters*, 59(8):845—848, 1987.
[72] M. Farzad, H. Tahersima, and H. Khaloozadeh. Predicting the mackey glass chaotic time series using genetic algorithm. In *2006 SICE-ICASE International Joint Conference*, pages 5460–5463, 2006.
[73] T. M. Martinetz, S. G. Berkovich, and K. J. Schulten. 'neural-gas' network for vector quantization and its application to time-series prediction. *IEEE Transactions on Neural Networks*, 4(4):558–569, 1993.
[74] C. H. López-Caraballo, I. Salfate, J. A. Lazzús, P. Rojas, M. Rivera, and L. Palma-Chilla. Mackey-glass noisy chaotic time series prediction by a swarm-optimized neural network. *Journal of Physics: Conference Series*, 720:012002, 2016.
[75] A. Faqih, A. P. Lianto, and B. Kusumoputro. Mackey-glass chaotic time series prediction using modified rbf neural networks. In *Proceedings of the 2nd International Conference on Software Engineering and Information Management*, ICSIM 2019, page 7–11. Association for Computing Machinery, 2019.
[76] J. Zhao, Y. Li, X. Yu, and X. Zhang. Levenberg-marquardt algorithm for mackey-glass chaotic time series prediction. *Discrete Dynamics in Nature and Society*, 2014:1–6, 11 2014.
[77] A. Goudarzi, A. Shabani, and D. Stefanovic. Product reservoir computing: Time-series computation with multiplicative neurons. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2015.
[78] M. A. Hajnal and A. Lőrincz. Critical echo state networks. In S. D. Kollias, A. Stafylopatis, W. Duch, and E. Oja, editors, *Artificial Neural Networks – ICANN 2006*, pages 658–667. Springer, 2006.
[79] H. Su, X. Ding, and W. Li. Numerical bifurcation control of mackey–glass system. *Applied Mathematical Modelling*, 35(7):3460–3472, 2011.
[80] G. Kiss and G. Röst. Controlling mackey–glass chaos. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(11):114321, 2017.

Appendix

A. Data-driven modeling

The field of data-driven modeling covers a wide range of algorithms and fields of application. On a very general level, the aim is to use data – obtained either via numerical simulations or measurements from experiments – to derive a model that is capable of predicting the future behavior of a system. The list of surrogate models for dynamical systems is extensive, with additional approaches being presented very regularly. These can be divided into different methodologies:

- Projection-based surrogate models [7, 40, 41],
- Dynamic Mode Decomposition / Koopman operator and generator [15, 14, 42, 43, 30, 16, 31],
- (Sparse) regression [44, 12, 13, 33],
- Stochastic modeling approaches [45, 46],

- Feed-forward neural networks [47],
- LSTMs [8, 9],
- Reservoir Computers [10, 11].

We have tested several of the above-mentioned approaches within the QuaSiModO framework. These will be described in more detail below.

A.1. Koopman operator:

Let g be an autonomous dynamical system defined on the state space Y (e.g., $\dot{y}(t) = g(y(t))$), and let $\Phi^t : Y \to Y$ with $\Phi^t(y) = y + \int_0^t g(y(t))\,dt$ be the associated flow map. Furthermore, let $f : Y \to \mathbb{R}^q$ be a real-valued observable of the system. Then the *Koopman semigroup* of operators $\{\mathcal{K}^t\} : \mathcal{F} \to \mathcal{F}$ with $\mathcal{F} = L^2(Y)$, which describes the evolution of the observable f (see [48, 49, 50, 51]), is defined by

$$(\mathcal{K}^t f)(y) = f(\Phi^t(y)),$$

and the Koopman semigroup is generated by the *Koopman generator* $\mathcal{L}$ [16, 29]:

$$\mathcal{L}f = \lim_{t \to 0} \frac{f \circ \Phi^t - f}{t}.$$

The Koopman operator and generator are linear yet infinite-dimensional operators acting on the observable of a system, i.e., on measurements. If we can find a finite-dimensional approximation, then we obtain an entirely data-driven linear system describing the dynamics of the observed quantities $z = f(y)$. One method to compute such a numerical approximation from data is *Extended Dynamic Mode Decomposition (EDMD)* [49, 52] (see [16] for the generator), which is a generalization of DMD [15, 42]. We assume that we have either measurement or simulation data, written in matrix form as

$$Z = \begin{bmatrix} z_1 & z_2 & \cdots & z_N \end{bmatrix} \quad \text{and} \quad \widetilde{Z} = \begin{bmatrix} \widetilde{z}_1 & \widetilde{z}_2 & \cdots & \widetilde{z}_N \end{bmatrix},$$

where $z_i = f(y_i)$ and $\widetilde{z}_i = f(\Phi^{\Delta t}(y_i))$ with a fixed step size Δt. For a given set of basis functions $\{\psi_1, \psi_2, \ldots, \psi_k\}$ (e.g., polynomials, radial basis functions, etc.), the data matrices are embedded into the typically higher-dimensional feature space by

$$\Psi_Z = \begin{bmatrix} \psi(z_1) & \ldots & \psi(z_m) \end{bmatrix} \text{ and } \Psi_{\widetilde{Z}} = \begin{bmatrix} \psi(\widetilde{z}_1) & \ldots & \psi(\widetilde{z}_m) \end{bmatrix},$$

with ψ being the vector of basis functions. With these data matrices, we then compute a finite-dimensional approximation of the Koopman operator $K \in \mathbb{R}^{k \times k}$ by

$$K^\top = \Psi_{\widetilde{Z}} \Psi_Z^+ = \left(\Psi_{\widetilde{Z}} \Psi_Z^\top\right)\left(\Psi_Z \Psi_Z^\top\right)^{-1},$$

where $^+$ denotes the pseudoinverse. The matrix K now allows us to define a discrete-time update for the observable z which approximates the true dynamics, thus yielding a linear system for the lifted observable $\hat{z} = \psi(z)$:

$$\hat{z}_{i+1} \approx \Phi^r(\hat{z}_i) = K^\top \hat{z}_i.$$

A.2. Proper Orthogonal Decomposition:

Consider a partial differential equation of the general form

$$\begin{aligned} \dot{y}(x,t) &= g(y(x,t)), && (x,t) \in \Omega \times (t_0, t_e], \\ a(x,t)\frac{\partial y(x,t)}{\partial n} + b(x,t)y(x,t) &= c(x,t), && (x,t) \in \Gamma \times (t_0, t_e], \\ y(x,t_0) &= y_0(x), && x \in \Omega, \end{aligned} \tag{6}$$

where Ω is the spatial domain of interest with boundary $\Gamma = \partial\Omega$ and corresponding outward normal vector n. The right-hand side g describes the evolution of the system. For details, the reader is referred to [53]. Since the state is space dependent, we have to take boundary conditions (BCs) into account, and the coefficients $a(x,t)$, $b(x,t)$ and $c(x,t)$ are given by the problem definition. Note that this covers both Dirichlet as well as Neumann BCs by neglecting one of the terms on the left hand side, respectively.

As a numerical discretization of such PDE systems (e.g., via finite elements) can in general be very expensive to solve, we want to reduce the model dimension by restricting the dynamics to a finite-dimensional linear subspace. This is realized by representing $y(x,t)$ in terms of basis functions $\{\psi_i(x)\}_{i=1}^{\ell}$:

$$y(x,t) \approx \sum_{i=1}^{\ell} z_i(t)\psi_i(x),$$

see [7, 40] for details. If we now insert this expansion into the weak formulation of the PDE, we obtain a system of nonlinear ODEs in the coefficients z:

$$\dot{z}(t) = g^r(z(t)) \quad \text{and} \quad z_{i+1} = \Phi^r(z_i) = z_i + \int_{t_i}^{t_{i+1}} g^r(z(t))\ dt.$$

Note that this requires knowledge of the equations as well as a careful and often tedious treatment of boundary conditions [25] and nonlinearities [54]. Moreover, in contrast to the other, non-intrusive modeling techniques, projection-based models predict the full state only.

In order to obtain a small yet informative set of basis functions $\{\psi_i(x)\}_{i=1}^{\ell}$, snapshot data is collected from the (spatially discretized) system and stored in a matrix $\hat{Y} = [\hat{y}(t_0), \dots, \hat{y}(t_N)]$, where the hat notation denotes the finite-dimensional spatial discretization of y. The singular value decomposition of $\hat{Y}$ then yields the orthonormal basis (given by the singular vectors) with the smallest L^2 projection error (equal to the sum of the neglected singular values). This procedure is known as *Proper Orthogonal Decomposition* (POD) and has been successfully applied to a wide range of nonlinear systems over the years, fluid dynamics being the most prominent example [40].

Quite a number of extensions have been presented for the use of POD in PDE-constrained control, see, for instance, [23] for the Burgers equation, which we use as one of our examples.

A.3. Reservoir Computing / Echo State Networks:

Reservoir Computers (RC) – also referred to as Echo State Networks (ESN) – have become very popular for time series prediction. This is mainly due to the straightforward and fast training process, which only consists of solving a linear system. Further details on RC and ESN can be found in the survey [55] or in [10, 56, 57].

The basic idea is to create a large recurrent neural network whose weights are initialized randomly and cannot be trained. This is called the reservoir. A linear output layer is then added, which can be trained efficiently via linear regression. The standard equations of an ESN are given by

$$\begin{aligned} r(k+1) &= \sigma(W^{\mathsf{in}} i(k) + W^{\mathsf{res}} r(k) + W^{\mathsf{fb}} o(k)), \\ o(k+1) &= W^{\mathsf{out}} r(k+1), \end{aligned}$$

where σ is a nonlinear activation function, e.g., $\sigma(x) = \tanh(x)$, W^{in}, W^{res} and W^{fb} are randomly generated (sparse) matrices and W^{out} is the trainable output matrix. The reservoir state $r(k+1)$ (for time step t_{k+1}) is computed based on a time-dependent input $i(k)$, the previous reservoir state $r(k)$ and the previous output $o(k)$. In the context of time series prediction the output is the (approximated) state of the system, i.e., $o(k) = y_k$. Since there is no additional time-dependent input in this setting, the term $W^{\mathsf{in}} i(k)$ is omitted.

To train the model, we take – similar to the Koopman operator approach – observed data (from measurements or simulations) to train our model, i.e., $\Phi^t : Y \to Y$ is the flow map which describes the dynamics of the system and $f : Y \to \mathbb{R}^q$ is a real-valued observable of the system. As the reservoir has a "memory" due to the feedback of the reservoir state, the data to train an ESN has to stem from a single time series:

$$Z = \begin{bmatrix} z_{t_0} & z_{t_1} & \cdots & z_{t_N} \end{bmatrix} \text{ with } z_{t_k} = f(\Phi^{t_k}(y_0)). \tag{7}$$

The first time steps are usually used to initialize the reservoir and are not used to train the linear output layer. Furthermore, in the feedback loop the output $o(k)$ is replaced by the true system state from the training data z_{t_k}.

There are many publications on the prediction of chaotic dynamical systems with ESNs, see, e.g., [58, 59, 60, 61, 62, 57]. Furthermore, some papers discuss the use of ESN in an MPC context [63, 64].

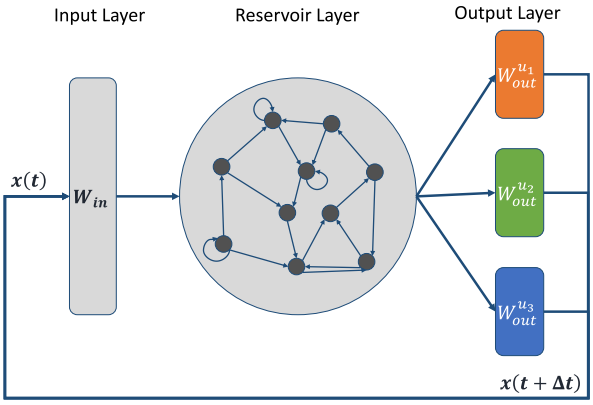


Figure 4: Schematic of a reservoir computer with different readout layers corresponding to different autonomous systems.

Therein, the control input u_k serves as the input $i(k)$. Due to this, the input space dimension increases, and therefore, a larger number of neurons is required in the reservoir to maintain a sufficiently accurate prediction. Moreover, the training will become harder.

By using the QuaSiModO framework, the dimension does no longer increase, since we train an individual ESN for each control $u^j \in V$. Since only the output layer W^{out} contains information corresponding to the control u^j, we can use the same reservoir for every ESN, cf. Fig. 4.

A.4. Long-short-term memory (LSTM) neural networks:

LSTM is a specific architecture for neural networks, more precisely for recurrent neural networks, which is specifically tailored to sequential data [8], e.g., time series prediction. In [9], the authors successfully applied the LSTM-approach to forecasting chaotic systems in a reduced order space. Here, we use the standard *tensorflow* implementation [65] which coincides with the formulation in [8].

Consider the flow map $\Phi^t : Y \to Y$ which describes the dynamics of the system. Furthermore, let $f : Y \to \mathbb{R}^q$ be a real-valued observable. We assume that we have training data where one data point is of the following form:

$$Z_{\text{in}} = \begin{bmatrix} z_{t_{k-d}} & z_{t_{k-d+1}} & \cdots & z_{t_k} \end{bmatrix} \quad \text{and} \quad Z_{\text{out}} = z_{t_{k+1}} \tag{8}$$

with $z_{t_k} = f(\Phi^{t_k}(y_0))$ for a given $y_0 \in Y$, i.e., the LSTM gets the time series Z_{in} as a sequential input and should predict the behavior of the dynamical system for one time step into the future (Z_{out}). Note that in the presented control framework, for training an LSTM model corresponding to control u^j, the delayed time series can be produced by different control inputs, but only the control u_k which maps z_{t_k} to $z_{t_{k+1}}$ has to be equal to u^j.

B. Detailed description of the numerical examples

In this section, we give a detailed description of the numerical setup of the examples presented in Figure 3. In all examples, we solve Problem (IV) with a tracking objective:

$$\begin{aligned} &\min_{\alpha \in ([0,1]^m)^p} \sum_{i=0}^{p-1} (z_{i+1} - z_{i+1}^{\text{ref}})^\top Q (z_{i+1} - z_{i+1}^{\text{ref}}) \\ &\text{s.t.} \quad z_{i+1} = \Phi^r(z_i, \alpha_i) = \sum_{j=1}^{m} \alpha_{i,j} \Phi_{u^j}^r(z_i) \quad \text{and} \quad \sum_{j=1}^{m} \alpha_{i,j} = 1, \qquad i = 0, 1, 2, \ldots, \end{aligned} \tag{$\widehat{\text{IV}}$}$$

where the quadratic, positive semidefinite matrix Q is problem-specific.

B.1. Control of the Lorenz system using the Koopman operator:

As a first example, we consider the Lorenz system which is probably the most studied system when it comes to chaotic dynamics. Despite the chaotic behavior, there are many studies showing that predictions

Table 2: Lorenz system with EDMD surrogate model.

	Parameter	Value
System parameters	(σ, ρ, β)	$(10, 28, \frac{8}{3})$
Quantization	U	$[-50, 50]$ or $[0, \pi]$
	V	$\{-50, 50\}$ or $\{0, \pi\}$
	m	2
Training data	Δt	0.0005
	T_{train}	100.0
	# trajectories	1
	Input	Piecewise constant, random $u_i \in V$
Surrogate model	Δt	0.05
	ψ	Monomials up to degree 3
	observable	$z = f(y) = y$
MPC	T_{MPC}	20.0
	p	3
	Q	$\text{diag}(0, 1, 0)$
	z_i^{ref}	$z_i^{\text{ref}} = 1.5 \cdot \sin(4\pi \cdot t_i / T_{\text{MPC}})$
	Δt_{SUR}	0.0005

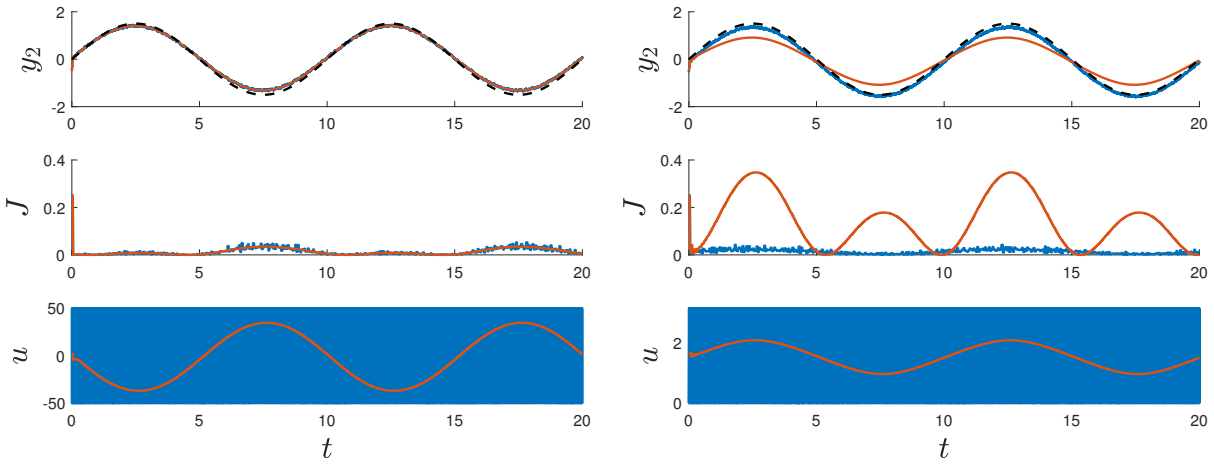


Figure 5: Solutions for the control affine (a) and the adapted Lorenz system (b) with EDMD. The solution of the linearly interpolated problem is shown in orange and the result generated by using SUR after each optimization step in blue. The reference trajectory is given by the dotted black line.

over time horizons of moderate length are possible using different methods such as neural networks [60] or sparse regression [12, 66]. The control task in our example is to track a reference trajectory for the second variable using an additive control input:

$$\frac{d}{dt}\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} \sigma(y_2 - y_1) \\ y_1(\rho - y_3) - y_2 \\ y_1 y_2 - \beta y_3 \end{pmatrix} + \begin{pmatrix} 0 \\ u \\ 0 \end{pmatrix}.$$

We here use EDMD as the surrogate model with $z = f(y) = y$. The detailed setting is described in Table 2. As the control input enters linearly, the solution of ($\widehat{\text{IV}}$) can be directly applied to the real system without an additional error. Therefore, it is not surprising that we achieve a very good control performance by using the interpolated solution, cf. Figure 5a. The SUR algorithm leads to a slightly worse solution which is caused by the discretization Δt_{SUR}.

In order to emphasize the importance of the SUR algorithm, we also study a nonlinear control input, i.e.,

$$\dot{y}_2 = y_1(\rho - y_3) - y_2 + 50 \cdot \cos(u),$$

and choose $U = [0, \pi]$, $V = \{0, \pi\}$. Here, an interpolation error is introduced by applying the linearly

interpolated control to the system and therefore, the SUR algorithm provides a substantially better solution, cf. Figure 5b. Except for the different control spaces U and V, the parameters were not changed with respect to the previous example.

In Figure 3, the results for the same setting but for tracking a piecewise constant trajectory are presented, i.e.,

$$z_i^{\mathsf{ref}} = \begin{cases} 0.0, & \text{if } t_i \leq 5.0 \\ 5.0, & \text{if } 5.0 < t_i \leq 10.0 \\ -10.0, & \text{else.} \end{cases}$$

B.2. Control of the Burgers equation using Proper Orthogonal Decomposition:

Our second example is the one-dimensional viscous Burgers equation:

$$\begin{aligned} \dot{y}(x,t) - \frac{1}{Re}\Delta y(x,t) + y(x,t)\nabla y(x,t) &= \sum_{j=1}^{5} v^j(t)\chi^j(x), && (x,t) \in [0,L] \times (t_0,t_e], \\ y(0,t) = y(L,t) &= 0, && t \in (t_0,t_e], \\ y(x,t_0) &= \begin{cases} 1, & x \in (0, \frac{L}{2}], \\ 0, & x \in (\frac{L}{2}, L), \end{cases} \end{aligned}$$

which was also studied by Kunisch and Volkwein in their seminal work on POD-based control of PDEs [23]. We consider a domain of length $L = 1$ and a distributed control that is realized via indicator functions χ_i with disjoint support:

$$\chi^j(x) = \begin{cases} 1, & \frac{(j-1)L}{5} < x \leq \frac{jL}{5} \\ 0, & \text{else} \end{cases} \quad \Rightarrow \quad \sum_{j=1}^{5} \chi^j(x) = 1 \text{ for } x \in (0,L],$$

with the aim to stabilize the system at $y^{\mathsf{ref}}(\cdot,t) = 0$. For the quantization of the control, we use a difference-star-like set with $u^1 = (0,0,0,0,0)$ and then two additional points per component, where the minimal and maximal value are taken, respectively. This results in a total of 11 autonomous systems. As we use POD as the surrogate model, we have to choose $z = y$. The detailed setup is described in Table 3.

Table 3: 1D Burgers equation with POD surrogate model.

	Parameter	Value
System parameters	Re	100
Quantization	U	$[-1,1]^5$
	V	$\left\{ \begin{pmatrix} -1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \ldots, \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \right\}$
	m	11
Training data	Δt	0.005
	T_{train}	50
	# trajectories	1
	Input	Piecewise constant, random $u_i \in V$
Surrogate model	Δt	0.025
	Basis size ℓ	12
	observable	$z = f(y) = y$
MPC	T_{MPC}	5
	p	5
	Q	Id
	z_i^{ref}	0

B.3. Control of a stochastic COVID-19 model using the Koopman operator:

In order to study the control of stochastic systems, we use the example of a compartment model for the COVID-19 outbreak in Germany in March 2020. The model was developed and validated using data from

Table 4: COVID-19 SIDARTHE model with Koopman operator surrogate model.

	Parameter	Value
System parameters	$(\beta, \epsilon, \zeta, \lambda, \mu, \kappa, \sigma, \tau)$	$(0.0084, 0, 0.079, 0.0566, 0.013, 0.0563, 0.044, 0.0288)$
Quantization	$U = V$	$\{0.0422, 0.1360, 0.1756, 0.3614\}$
	m	4
Training data	Δt	$\frac{1}{12}$ (day)
	T_{train}	200
	# trajectories	40
	Input	10 Sim. per input $u_i \in V$
Surrogate model	Δt	7 (days)
	ψ	Identity ($\rightarrow$ DMD)
	observable	$z = f(y) = (I, D, A, R, T)$
MPC	T_{MPC}	700
	p	4
	d_1	10^7
	d_2	0.03

Italy in [6], and adapted to Germany and used for optimal control in [67]. The model consists of the eight compartments **S**usceptible, **I**nfected (asymptomatic, undetected), **D**iagnosed (asymptomatic, detected), **A**iling (symptomatic, undetected), **R**ecognized (symptomatic, detected), **T**hreatened (symptomatic with life-threatening symptoms, detected), **H**ealed (immune after prior infection, detected or undetected), and **E**xtinct (dead, detected), and the dynamics are described by an eight-dimensional ODE:

$$\frac{d}{dt}\begin{pmatrix} S \\ I \\ D \\ A \\ R \\ T \\ H \\ E \end{pmatrix} = \begin{pmatrix} -S(\alpha I + \beta D + \gamma A + \beta R) \\ S(\alpha I + \beta D + \gamma A + \beta R) - (\epsilon + \zeta + \lambda)I \\ \epsilon I - (\zeta + \lambda)D \\ \zeta I - (\theta + \mu + \kappa)A \\ \zeta D + \theta A - (\mu + \kappa)R \\ \mu A + \mu R - (\sigma + \tau)T \\ \lambda I + \lambda D + \kappa A + \kappa R + \sigma T \\ \tau T \end{pmatrix}.$$

The coefficients $\alpha, \beta, \gamma, \epsilon, \zeta, \lambda, \mu, \kappa, \sigma, \tau$ denote the transfer rates between the various compartments and are fitted using measurement data. In our case, we use the values given in [67] to match the dynamics of the COVID-19 outbreak in Germany in March 2020.

As each individual is contained in exactly one compartment, the sum over all compartments is 1 at all times. To account for uncertainties in the number of infections, we add a normally distributed random variable $N \sim \mathcal{N}(0, 1)$ to the number of infections that is scaled by I:

$$\dot{I} = S(\alpha I + \beta D + \gamma A + \beta R) - (\epsilon + \zeta + \lambda)I + \frac{I}{3}N.$$

We use the standard Euler Maruyama method for the numerical time integration, and in order to maintain a constant number of individuals, we calculate $S = 1 - (I + D + A + R + T + H + E)$ in every time step.

The system can be controlled by adapting the values for α and γ (with $\alpha = \gamma$ for simplicity), which can take four different values corresponding to different counter-measures such as social distancing or the lock-down of restaurants and businesses.

We want to minimize the current number of infectious individuals, i.e., the sum of I, D, A, R and T. Furthermore, the number of threatened individuals should not exceed the number of free beds in intensive care units, which is why we penalize large values of T. Finally, counter-measures can potentially result in economic and social damage, which is why we want to minimize the use of these measures as well. In

summary, we obtain the following mixed-integer optimal control problem:

$$\begin{aligned}
\min_{u\in\{\alpha^1,\dots,\alpha^4\}} \sum_{i=0}^{p-1}(I_i^2+D_i^2+A_i^2+R_i^2+T_i^2)+d_1\max\{0,(T-T^{\mathsf{max}})^2\}\\
+d_2\max\left\{10^{-4},10^{-1}-\sqrt{I_0+D_0+A_0+R_0+T_0}\cdot u_i\right\}\\
\text{s.t.}\quad (I_{i+1},D_{i+1},A_{i+1},R_{i+1},T_{i+1},H_{i+1},E_{i+1})^\top=\Phi(I_i,D_i,A_i,R_i,T_i,H_i,E_i,N_i,u_i),\\
S_{i+1}=1-(I_{i+1}+D_{i+1}+A_{i+1}+R_{i+1}+T_{i+1}+H_{i+1}+E_{i+1}),\\
(S_0,I_0,D_0,A_0,R_0,T_0,H_0,E_0)=(S^0,I^0,D^0,A^0,R^0,T^0,H^0,E^0).
\end{aligned}$$

As the surrogate model, we use again the Koopman operator, which yields a linear system for the expected value [52]. As the objective function does not depend on S, H and E, we exclude these from the observable $z=(I,D,A,R,T)$. The detailed setup is described in Table 4.

B.4. Control of the flow around a cylinder using an LSTM neural network:

Next, we want to control the flow around a cylinder – governed by the two-dimensional incompressible Navier–Stokes equations – which exhibits periodic vortex shedding at $Re=100$ [68]. Our aim is not to control the flow field, but the forces acting on the cylinder (the lift C_L and the drag C_D) without any knowledge of the flow field itself.

As the surrogate model we use an LSTM neural network. The control aim is to force the lift to follow a predefined trajectory. This was already done in [24] but there the control input was continuous and a different RNN architecture was used to build the surrogate model. For detailed parameters of the setting see Table 5.

Table 5: 2D Cylinder flow with LSTM surrogate model.

	Parameter	Value
System parameters	Re	100
Quantization	U	$[-5,5]$
	V	$\{-5,0,5\}$
	m	3
Training data	Δt	0.05
	T_{train}	(2 *) 500
	# trajectories	2
	Input	Piecewise constant, random $u_i\in V$
Surrogate model	Δt	0.1
	neurons per LSTM-cell	500
	observable	$z=f(y)=(C_D,C_L)$
	delay coordinates	15
MPC	T_{MPC}	20.0
	p	5
	Q	$\begin{bmatrix}0&0\\0&1\end{bmatrix}$
	z_i^{ref}	$\sin(\frac{t_i}{2})$
	Δt_{SUR}	0.05

B.5. Control of the Mackey-Glass delay differential equation for blood cell reproduction using Reservoir Computing (ESN):

The last example is the control of the Mackey-Glass equation which is a delay differential equation modeling blood cell reproduction [69, 70]:

$$\dot{y}(t)=\beta\frac{y(t-\tau)}{1+y(t-\tau)^\eta}-\gamma y(t)+u(t)\qquad\text{with }\beta,\gamma,\eta>0. \tag{9}$$

The uncontrolled system (i.e., $u(t)=0$) was studied for different parameters and chaotic behavior was proven for certain parameter values, for instance $\beta=2$, $\gamma=1$ and $\eta=9.65$ [70]. Since this system has

Table 6: Mackey-Glass DDE with ESN surrogate model.

	Parameter	Value
System paramters	$(\beta, \gamma, \eta, \tau)$	$(2, 1, 9.65, 2)$
Quantization	U	$[-0.2, 1.0]$
	V	$\{-0.2, 0.0, 1.0\}$
	m	3
Training data	Δt	0.05
	T_{train}	1000.0
	# trajectories	1
	Input	Piecewise constant, random $u_i \in V$
Surrogate model	Δt	0.25
	size residuum	200
	spectral radius (W^{res})	0.75
	sparsity (W^{res})	0.9
	σ	0.99
	β	0.0001
	observable	$z = f(y) = y$
MPC	T_{MPC}	20.0
	p	5
	Q	1
	z_i^{ref}	1.0

become a benchmark for predicting chaotic delay systems, there are several studies on data-based models, and many different methods were used such as local linear approximation [71], genetic algorithms [72] and different forms of neural networks [73, 74, 75, 76]. Another approach is via Echo State Networks (ESN), which were studied in [57, 77, 78].

The additive term to control the system represents – in the context of blood reproduction – an increase in the number of blood cells caused by, e.g., a transfusion. The same control problem was already studied in [79, 80], where the authors derived different feedback laws in order to stabilize the systems.

Here, we use a ESN as surrogate model. As mentioned in Appendix A.3 we can use a single reservoir for all possible controls, i.e., only the output layer W^{out} differs. For the detailed setting see Table 6. Note, that we do not need delay coordinates although the state depends on $y(t)$ and $y(t-\tau)$ since the ESN is able to capture the past dynamics by the feedback of the reservoir state. Since the control enters linearly – similar to the Lorenz system – the interpolated solution is exact ($E_{\mathsf{V}}(V) = 0$). The results in Figure 3 show that the system can be stabilized at 1.0.