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

Sebastian Peitz\textsuperscript{1} and Katharina Bieker\textsuperscript{2}

\textsuperscript{1}Department of Computer Science, Paderborn University, Germany
\textsuperscript{2}Department of Mathematics, Paderborn University, Germany

Abstract

The advances in data science and machine learning have resulted in significant improvements regarding the modeling and simulation of nonlinear dynamical systems. It is nowadays possible to make accurate predictions of complex systems such as the weather, disease models or the stock market. Predictive methods are often advertised to be useful for control, but the specifics are frequently left unanswered due to the higher system complexity, the requirement of larger data sets and an increased modeling effort. In other words, surrogate modeling for autonomous systems is much easier than for control systems. In this paper we present the framework QuaSiModO (Quantization-Simulation-Modeling-Optimization) to transform arbitrary predictive models into control systems and thus render the tremendous advances in data-driven surrogate modeling accessible for control. Our main contribution is that we trade control efficiency by autonomizing the dynamics – which yields mixed-integer control problems – to gain access to arbitrary, ready-to-use autonomous surrogate modeling techniques. We then recover the complexity of the original problem by leveraging recent results from mixed-integer optimization. The advantages of QuaSiModO 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 in use, and little prior knowledge requirements in control theory to solve complex control problems.

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 \cite{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 \cite{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 \cite{3}, the flow of blood within arteries \cite{4} or of cerebrospinal fluid within the brain \cite{5}, and – more recently – the dynamics of pandemics such as COVID-19 \cite{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 \cite{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 \cite{8,9} or Reservoir Computers / Echo State Networks (ESN)
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 model predictive control (MPC) methods [23] using either intrusive approaches (i.e., the equations need to be known) such as POD [24] or black-box methods, for instance specific neural network architectures [25]. 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 [26]. In pure black-box methods, a straight-forward approach to avoid this tedious effort is to transform control inputs via state augmentation [27], 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 [28]. Recently, an alternative was presented in [29], 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 [30]) 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) **Quantization** of the the admissible control \( U \) (for instance by replacing the interval \( U = [u^{\text{min}}, u^{\text{max}}] \) by the bounds \( V = \{u^{\text{min}}, u^{\text{max}}\} \));

(ii) **Simulation** of the individual autonomous systems (e.g., \( \Phi_{u^{\text{min}}}(y) = \Phi(y, u^{\text{min}}) \) and \( \Phi_{u^{\text{max}}}(y) = \Phi(y, u^{\text{max}}) \)); We may also use experimental data;

(iii) **Modeling** 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;

(iv) **Optimization** using the 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.

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 $\Phi$ with constant time step $\Delta t = t_{i+1} - t_i$, $i = 0, 1, \ldots$, 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: $U = [u_{\text{min}}, u_{\text{max}}]$. 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 of MPC can be formalized in an optimal control problem of the following form:

$$\min_{u \in U} J(y) = \min_{u \in U} \sum_{i=0}^{p-1} P(y_{i+1})$$

subject to $y_{i+1} = \Phi(y_i, u_i), \quad i = 0, 1, 2, \ldots,$

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 $P(y_i) = \|y_i - y_{\text{ref}}\|^2$. Note that no explicit dependency of $P$ on $u$ is assumed, as this will be favorable for the error analysis. Furthermore, we assume that there exists a control-to-state mapping that uniquely defines $y$ when choosing $u$, such that the state itself is not a control variable.

1.1. Quantization:

In the first step of QuaSiModO, we quantize the control set $U$ such that only a finite subset $V = \{u^1, \ldots, u^m\} \subseteq U$ 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 problem:

$$\min_{u \in V} J(y) = \min_{u \in V} \sum_{i=0}^{p-1} P(y_{i+1})$$

subject to $y_{i+1} = \Phi_{u_i}(y_i), \quad i = 0, 1, 2, \ldots,$

Remark 1.1. 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 that the optimal value of $J$ in Problem (II) is arbitrarily close to the optimal value of Problem (I), cf. Lemma 2.2. In the control-affine case, this is ensured by simply choosing the control bounds.

1.2. Simulation & Modeling:

Problem (II) allows for the straightforward introduction of surrogate models, as we can simply replace the individual autonomous systems $\Phi_{u_i}(y)$ by respective reduced systems $\Phi_{u_i}^r(z)$ (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 collect data from one long time series $Z = [z_1, \ldots, z_N]$ with random actuation $u_1, \ldots, u_N \in V$ (which we then have to split into data sets for the individual systems).

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:

$$
\min_{u \in V} J^r(z) = \min_{u \in V} \sum_{i=0}^{p-1} P^r(z_{i+1}) \quad \text{(III)}
$$

$$
s.t. \quad z_{i+1} = \Phi_{u_i}^r(z_i), \quad i = 0, 1, 2, \ldots.
$$

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 or the observables $f$ are chosen such that the objective can be calculated from the observed quantities. This is particularly the case for real systems, where the measurements cannot be chosen arbitrarily, but need to be technically realized via sensors. Formally, we assume

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

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 [31], bounds for the prediction error with finite training data were presented in [32, 33, 34, 35], and probabilistic error bounds for the approximation of linear systems via finite data in [36]. Seeing the ever increasing effort, it is likely that more progress will be made in the near future. One of the goals of QuaSiModO is to exploit these advances in a systematic way for control, by deriving error bounds that exclusively rely on existing bounds for autonomous dynamics. To this end, we assume in the following that an error bound for the deviation over a time step $\Delta t$ of the following form is known:

$$
\|f(\Phi_{u_0}(y_0)) - \Phi_{u_i}^r(z_0)\|_\infty \leq E(\|f(y_0) - z_0\|, \Delta t),
$$

where $y_0$ denotes the initial value of the full state of the real system and $z_0$ denotes the value the surrogate model is initialized with. In the first time step, $z_0 = f(y_0)$ does hold, but this is usually not valid in subsequent steps due to the model error. Thus, we obtain the time-dependent model error $E_{\text{model}}$:

$$
E_{\text{model}}(t_0) = \|f(y_0) - z_0\|, \quad E_{\text{model}}(t_i) \leq E(E_{\text{model}}(t_{i-1}), \Delta t). \quad (2)
$$
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 [37]. 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 (SUR) algorithm from [38]. 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., [25].

As a first step towards the relaxation of (III), we introduce a new (binary) control variable \( \omega \in \{0,1\}^m \) that yields a formulation equivalent to (III):

\[
\begin{align*}
\min_{\omega \in \{0,1\}^m} J^\alpha(z) &= \min_{\omega \in \{0,1\}^m} \sum_{i=0}^{p-1} P^\alpha(z_{i+1}) \\
\text{s.t.} \quad z_{i+1} &= \Phi^\alpha(z_i, \omega_i) = \sum_{j=1}^m \omega_{i,j} \Phi^\alpha_{u,j}(z_i), \tag{III-\omega}
\end{align*}
\]

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 \( i = 0, \ldots, p-1 \).

In order to obtain a continuous optimization problem which can be solved more efficiently than (III) or (III-\omega), where the control inputs are discrete, Problem (III-\omega) can be relaxed by replacing \( \omega_i \in \{0,1\} \) with \( \alpha_i \in [0,1] \).

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

The main differences to Problem (I) are the additional condition on \( \alpha \) and the fact that the dimension of the control variable is changed by a factor of \( (m-1)/\dim(U) \) (the last entry of \( \alpha_i \) 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 [37] can be used.

The final step of the optimization is now to construct the control \( u^* \) from the optimal solution \( \alpha^* \). To this end, we proceed according to the SUR procedure for mixed integer problems as proposed in [38], where \( \alpha^* \) is transformed back to \( \omega^* \) while taking past rounding decisions into account:

\[
\begin{align*}
\hat{\omega}_{i,j} &= \sum_{k=0}^{i} \alpha^*_{k,j} - \sum_{k=0}^{i-1} \omega^*_{k,j} \tag{3a} \\
\omega^*_{i,j} &= \begin{cases} 
1, & \hat{\omega}_{i,j} \geq \hat{\omega}_{i,l} \ \forall \ l \neq j \ \text{and} \\
j < l \ \forall \ l : \hat{\omega}_{i,j} = \hat{\omega}_{i,l}, & \text{else.} 
\end{cases} \tag{3b} \\
u_i^* &= \sum_{j=1}^m \omega^*_{i,j} u^j, \quad i = 0, 1, 2, \ldots. \tag{3c}
\end{align*}
\]
For control-affine systems and convex sets $U$, an alternative, straightforward approach is to simply calculate the convex combination of the individual $u^i \in V$:
\[
  u^*_i = \sum_{j=1}^{m} \alpha^*_i, u^j, \quad i = 0, 1, 2, \ldots
\]

In the control-affine case, the two solutions are very close to one another, as experiments for the Lorenz system as well as the cylinder flow show, cf. Figure 3 in Section 3. Note that in terms of the objective function value, we can prove convergence for both the rounded and non-rounded approaches. However, an error bound on the control signal $u$ is not achievable in the rounded case.

For the numerical solution of the resulting optimization problems, we simply use the standard optimizer from the SciPy library, i.e., “SLSQP” [39] or “trust-constr” [40] 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 $\Phi^t$ 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 performance of the QuaSiModO approach in terms of the objective function value, 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 $E_{\text{model}}$ (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 [38]. 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 $J^*_1$ of (I) is always at least as small as the optimal value $J^*_2$ 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 [41] 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 $\Delta t$ between the control inputs 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 [38, 42] to derive error bounds for the SUR algorithm. Unless otherwise specified, we denote by $\|\cdot\|$ the maximum norm $\|\cdot\|_{\infty}$ in the following, and $L^1((0,T),Y)$ is the space of Lebesgue integrable functions mapping from the interval $(0,T)$ to the state space $Y$.

**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
\[
  y(t) = y_0 + \int_0^t g(y(\tau), u(\tau)) \, d\tau \quad \text{and} \quad \bar{y}(t) = \bar{y}_0 + \int_0^t \bar{g}((\bar{y}(\tau), \bar{u}(\tau)) \, d\tau,
\]
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
\[
  \sup_{t \in [0,T]} \left\| \int_0^t g(y(\tau), u(\tau)) - \bar{g}(\bar{y}(\tau), \bar{u}(\tau)) \, d\tau \right\| \leq M
\]
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. The claim mainly follows from Grönwall’s lemma.

In the MPC context, \( y_0 = \bar{y}_0 \) is the current state and we obtain a bound for the distance between two trajectories related 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 optimal 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. Problems (III-\( \omega \)) and (IV).

2.2. Bound without model error

We begin without using a surrogate model, i.e., \( z = f(y) = y \) and \( \Phi^c = \Phi \), 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 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 [38] to show that we obtain a trajectory of the binary control problem (III-\( \omega \)) (which is equivalent to (II) in this setting) that is arbitrarily close (for arbitrary small switching times) to the solution of (IV) by using SUR.

Lemma 2.2. Let \( U \subseteq \mathbb{R}^n \) be bounded and \( V = \{u^1, \ldots, u^m\} \subseteq U \) 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 \( V \), more precisely,

\[
D = \sup_{t \in [0,T]} \sup_{g \in g(y(t),U)} d(g, \text{Conv}(g(y(t),V)))
\]

with \( d(a, B) = \inf_{b \in B} \|a - b\| \) for \( a \in \mathbb{R}^n \) and \( B \subseteq \mathbb{R}^n \).

Proof. For every \( t \in [0, T] \), let \( y_c(t) \) be the element in \( \text{Conv}(g(y(t), V)) \) which is closest to \( g(y(t), u(t)) \), i.e.,

\[
\|g(y(t), u(t)) - y_c(t)\| = d(g(y(t), u(t)), \text{Conv}(g(y(t), V))).
\]

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 \( t \in [0, T] \)

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

\[
\leq T \cdot \sup_{t \in [0,T]} \|g(y(t), u(t)) - y_c(t)\| = T \cdot \sup_{t \in [0,T]} d(\bar{y}, \text{Conv}(g(y(t), V))) = M_1.
\]

\[= D < \infty, \text{ (continuity of } g, y \text{ and boundedness of } U)\]
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 $w^j \in V$, 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 $g : Y \times U \rightarrow Y$, $y : [0, T] \rightarrow Y$ and $\alpha : [0, T] \rightarrow [0, 1]^m$ be measurable functions and assume that $\omega : [0, T] \rightarrow \{0, 1\}^m$ is constructed from $\alpha$ via SUR (cf. Eq. (3) or [38]) with time discretization $\Delta t$. Then,

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

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

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

Then,

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

**Proof.** The proof can be found in [38].

**Remark 2.4.** A similar result can be found in [42] but with weaker assumptions on $g$. There, the authors prove that for $g(y(\cdot), v) \in L^1((0, T), Y)$,

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

if

$$\lim_{\Delta t \to 0} \left\| \int_0^t \alpha(\tau) - \omega(\tau) d\tau \right\| = 0.$$

Furthermore, they show that the statement holds not only for ODEs, but for semilinear PDEs as well. Here, we will use the result from [38], 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 $U \subseteq \mathbb{R}^n_+$ be bounded and $V = \{w^1, \ldots, w^m\} \subseteq U$ be a finite subset. Assume $g : Y \times U \rightarrow Y$ is continuous, $u : [0, T] \rightarrow U$ is measurable and $y : [0, T] \rightarrow Y$ is defined by

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

Furthermore, let $g(y(t), w^j)$ 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 $w^j \in V$ such that for almost all $t \in [0, T]$:

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

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

$$\bar{y}(t) = y_0 + \int_0^t g(\bar{y}(\tau), \bar{u}(\tau)) d\tau, \quad \bar{y}_0 \in Y,$$

it holds

$$\|y(t) - \bar{y}(t)\| \leq (M_1 + \varepsilon + \|y_0 - y_0\|) \cdot e^{L_g t} \quad \forall t \in [0, T].$$
Proof. Lemma 2.2 ensures the existence of a measurable function $\alpha : [0, T] \to [0, 1]^m$ with $\sum_{j=1}^m \alpha_j(t) = 1 \forall t \in [0, T]$ and $M_1 > 0$, such that

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

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

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

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

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

Since $\bar{u}(t) := \sum_{j=1}^m \omega_j(t) u^j \in V$ for all $t \in [0, T]$, choosing $\Delta t$ 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. $\square$

Remark 2.6. The $m$ in $M_2(\Delta t)$ in Theorem 2.3 and 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 $T$.

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 $(\bar{y}^*, u^*)$ 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 $g : Y \times U \to \mathbb{R}$ 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

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

Proof. The claim follows from Theorem 2.5 and the Lipschitz continuity of $P$. $\square$

Remark 2.8. The errors

$$E_V(V) = L_P M_1 \frac{e^{L_P \Delta t} (e^{L_P \Delta t} - 1)}{e^{L_P \Delta t} - 1}$$

and

$$E_0(\Delta t) = L_P M_2(\Delta t) \frac{e^{L_P \Delta t} (e^{L_P \Delta t} - 1)}{e^{L_P \Delta t} - 1}$$

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 $M_1 = 0$, cf. Lemma 2.2. Moreover, we have

$$\lim_{\Delta t \to 0} \frac{e^{L_P \Delta t} (e^{L_P \Delta t} - 1)}{e^{L_P \Delta t} - 1} = p \quad \text{and} \quad \lim_{\Delta t \to 0} M_2(\Delta t) = 0,$$

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 $U$. 

---

9
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 $J^*$ 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 $S : U^p \rightarrow Y^p$ and $S^r : U^p \rightarrow f(Y)^p$ which – for a fixed $y_0 \in Y$ – map the control inputs $(u_0, \ldots, u_{p-1})$ to the corresponding states $(y_1, \ldots, y_p)$ and $(z_1, \ldots, z_p)$, 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 $|J(S(u^*_{(I)})) - J(S(u^*_{(III)}))|$, where $u^*_{(I)}$ and $u^*_{(III)}$ are the optimal solutions of (I) and (III), respectively. Since we already have a bound for $|J(S(u^*_{(I)})) - J(S(u^*_{(III)}))|$, it is sufficient to determine the error $|J(S(u^*_{(III)})) - J(S(u^*_{(III)}))|$. Therefore, we first consider the error between $J(S(u))$ and $J^*(S^r(u))$ for arbitrary $u \in V^p$:

$$|J(S(u)) - J^*(S^r(u))| = |J^*(f(S(u))) - J^*(S^r(u))|$$

$$= \sum_{i=0}^{p-1} |P^r(f((S(u))_i)) - P^r((S^r(u))_i)|$$

$$\leq \sum_{i=0}^{p-1} L_P \|J((S(u))_i) - (S^r(u))_i\|$$

$$\leq \sum_{i=0}^{p-1} L_P \|J((S(u))_i) - (S^r(u))_i\|$$

$$\leq L_P \sum_{i=1}^{p} E_{\text{model}}(t_i),$$

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 the maximum of the two as $L_P$. Now, it holds

$$|J(S(u^*_{(III)})) - J(S(u^*_{(III)}))| = J(S(u^*_{(III)})) - J(S(u^*_{(III)}))$$

$$\leq |J(S(u^*_{(III)})) - J(S(u^*_{(III)}))|$$

$$\leq |J(S(u^*_{(III)})) - J(S(u^*_{(III)}))|$$

$$\leq \sum_{i=0}^{p} E_{\text{model}}(t_i),$$

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

$$|J(S(u^*_{(III)})) - J(S(u^*_{(III)}))|$$

$$\leq E_Q(V) + E_M(\Delta t) + 2L_P \sum_{i=0}^{p} E_{\text{model}}(t_i),$$

(E1)
see Eq. (2) for $E_{\text{model}}$ and Eq. (4) for $E_V(V)$ and $E_M(\Delta t)$.

In many practical applications, we will have a finite set $V = \{u^1, \ldots, u^{m}\} \subseteq U$ 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 $u^*_{(IV)-\text{SUR}}$. The error is composed of the error (E1) and $\left|J(S(u^*_{(III)})) - J(S(u^*_{(IV)-\text{SUR}}))\right|$. Analog to the derivation before, we obtain an error $E_{M^r}(\Delta t)$ for the surrogate-based mixed-integer transformation (with variables $M^r_I(\Delta t)$ 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

\[
|J(S(u^*_{(I)})) - J(S(u^*_{(IV)-\text{SUR}}))| \\
\leq E_V(V) + E_M(\Delta t) + 2E_r(E) + E_{M^r}(\Delta t),
\]

(E2.a)

where the additional second model error $E_r(E)$ is due to the fact that we need to estimate the difference in terms of the true objective.

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

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

\[
|J(S(u^*_{(I)})) - J(S(u^*_{(IV)-\text{SUR}}))| \\
\leq E_V(V) + E_M(\Delta t) + E_r(E). 
\]

(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 $\text{Conv}(V) \subseteq U$. 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

\[
|J(S(u^*_{(I)})) - J\left(S\left(\sum_{j=1}^{m} \alpha^*_{(IV),j} u^j\right)\right)| \leq E_V(V) + E_r(E). 
\]

(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 $E_V(V) = 0$ if the convex hull of the reachable set corresponding to $V$ is a subset of the one corresponding to $U$, and both $E_M(\Delta t)$ and $E_{M^r}(\Delta t)$ vanish as the switching time $\Delta t$ tends to zero.

### 2.4. Example

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

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

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., we set $\varepsilon = 0$ and $\varepsilon = 10^{-1}$ for the true and the surrogate model, respectively. As the finite set of controls we choose $V = \{-4, 4\}$. To determine the
model error $E_{\text{model}}(t_i)$ 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
\[
\sup_{t \in [0, \Delta t]} \left\| \int_0^t g(y(\tau), u(\tau)) - g'(y(\tau), u(\tau)) \right\| \leq \varepsilon \Delta t
\]
and according to Lemma 2.1, we can estimate the model error via
\[
E_{\text{model}}(t_{i+1}) = \|y_{i+1} - y'_{i+1}\| \leq (\varepsilon \Delta t + \|y_i - y'_i\|) e^{L_y \Delta t}
\]
where $y_i$ and $y'_i$ are the discrete trajectories defined by $g$ and $g'$, 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 $y_{\text{ref}} = (0, 0)^T$, 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 $\Phi$ and $\Phi'$, as well as for the SUR.

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 $u_{\text{min}}$ and $u_{\text{max}}$, 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 $\varepsilon$ 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, even though the two errors come closer with decreasing $\Delta t$. 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 dynamics of blood cells (modeled by the Mackey-Glass delay
Figure 3.: MPC using QuaSiMoDo applied to various combinations of systems and surrogate models. In the last row, the results of the MPC are shown for the settings explained above. For each system, the first row shows the state space, the second one the tracking error and the last one the obtained control over time. For the Burgers equation and the Mackey-Glass system, the control task was to stabilize the system at 0 and 1, respectively. For the other systems, the reference trajectory is shown by the dashed lines in the $y$ plot. In all examples, problem (IV) was solved. For the Lorenz and Navier–Stokes example, both the continuous (orange) and the control computed via SUR (blue) are shown. In the remaining examples, we have applied the continuous solution of Problem (IV) to the real system. Higher-dimensional inputs are considered for the Burgers equation ($n_u = 5$) and the fluidic pinball ($n_u = 3$).

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 standard ESN. This highlights the flexibility and broad applicability of the method and the success of the technique in constructing data-driven feedback controllers.

In the following, we describe three experiments in more detail. First, the Lorenz system is analyzed to present the difference between affine linear and nonlinear control systems. We then briefly comment on the fluidic pinball, which has a higher-dimensional input and possesses the most complex dynamics from all our experiments. Finally, the Mackey Glass equation is described to illustrate the potential to exploit certain surrogate model structures. For reconstruction of the results, we refer to the Code and its documentation which is available online, as well as the more detailed description in the appendices.

3.1. Lorenz system & Koopman operator:

As a first example, we consider the Lorenz system. Despite the chaotic dynamics, there are many studies showing that predictions over time horizons of moderate length are possible using different methods such as neural networks [43] or sparse regression [12, 44]. Here, the control task is to track a reference...
trajectory for the second variable, i.e., \( P(y_i) = \| y_{2,i}^{\text{ref}} - y_{2,i} \|_2 ^2 \), using an additive control input:

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

(6)

where \((\sigma, \rho, \beta) = (10, 28, \frac{8}{3})\). As the surrogate model, we use the Koopman operator approximated via EDMD \([45, 46]\), and the full state observable \( z = f(y) = y \). The control sets are \( U = [-50, 50] \) and \( V = \{-50, 50\} \), respectively, and we collect 2000 data points over 100 seconds \((\Delta t = 0.05)\) using piecewise constant inputs from \( V \). As the control input enters linearly, the solution of (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 using both the interpolated and rounded solutions, cf. Figure 4 (a).

In order to emphasize the importance of the effect of nonlinear control inputs, we study a variation of the problem, i.e.,

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

(7)

with \( U = [0, \pi] \), and chose \( V = \{0, \pi\} \) to ensure \( M_1 = 0 \), cf. Lemma 2.2. 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 4 (b).

3.2. Fluidic pinball & Koopman operator:

The fluidic pinball is an unsteady flow around three cylinders arranged on an equidistant triangle, governed by the two-dimensional incompressible Navier–Stokes equations. In [47], it was conjectured that the system exhibits chaotic behavior above the critical Reynolds number \( Re \approx 115 \). In our control setting, we use \( Re = 200 \) as in [25, 30], which likely results in the most complex problem from a dynamics perspective. The goal is to perform tracking of the lift coefficients of the three cylinders, and the control input is the rotation of all cylinders, meaning that we have a three-dimensional input. For the surrogate modeling, we proceed in the same way as for the Lorenz system, and approximate the Koopman operator using EDMD. We do not observe the full state, but only the lift and drag of the three cylinders, which allows for a much more efficient (and more realistic, from an engineering perspective) modeling. As the fluid solver, we use OpenFOAM, and the setup is identical to the references above. Moreover, the example can be found in the accompanying git repository https://github.com/SebastianPeitz/QuaSiModO.
### 3.3. Mackey-Glass equation & Echo State Network

The second example is the control of the Mackey-Glass equation which is a delay differential equation modeling blood cell reproduction:

$$\dot{y}(t) = \beta \frac{y(t - \tau)}{1 + y(t - \tau)^\eta} - \gamma y(t) + u(t).$$  \hspace{1cm} (8)

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 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 $\eta = 9.65$ [48].

Here, we use an **Echo State Network** (ESN) (or **Reservoir Computer**) [49] as the surrogate model (cf. Figure 5), which is a specific type of recurrent neural network (RNN). The basic idea is to create a large RNN (the reservoir) whose weights are initialized randomly and cannot be trained. Instead, a linear readout layer is responsible for extracting the prediction of the next time step, and it can be trained very efficiently using linear regression.

Since only the output layers $W_{out}$ contain information corresponding to the control $u$, we can use a single reservoir for all possible controls. Note, that we do not need to explicitly account for the delay coordinates since the ESN is able to capture past dynamics via the feedback of the reservoir state. Since the control enters linearly – similar to the Lorenz system – the interpolated solution is exact. The results in Figure 3 show that the system can be stabilized at 1.0.

### 4. Discussion of the advantages and limitations

We now briefly discuss the advantages and limitations of our approach to take the effort of reformulating a continuous control problem into a mixed-integer one in order to make easier use of existing surrogate modeling techniques.

**Advantages:** For some surrogate modeling techniques, the task of including control inputs can be tackled via state augmentation, i.e., by introducing $\tilde{y} = (y, u)$. However, this is by no means possible for all approaches. Furthermore, state augmentation results in a higher state space dimension and the loss of a low-dimensional subspace (the attractor) the dynamics is restricted to, such that the collection of data becomes more challenging. Our approach presents a remedy to these issues, which we will now discuss in more detail.
4.1. Straightforward use of existing autonomous models and the corresponding error bounds

For some surrogate models – most importantly projection based approaches such as POD or reduced basis for partial differential equations – the state space cannot simply be extended, as it is typically a Sobolev space and not $\mathbb{R}^n$. It is thus not clear how to autonomize the dynamics, which becomes even more challenging due to the need to satisfy the boundary conditions [26]. Second, if state augmentation is possible, then it may still strongly limit the type of input. For instance, in the case of the Koopman operator, we are restricted to linear inputs [50, 30]. QuaSiModO avoids these challenges altogether, as we can simply use the autonomous systems in our control framework.

This has another important consequence, as we can readily use error bounds from the autonomous modeling world if they are available (such as [51, 52, 36, 32, 33, 34, 35]).

4.2. Easier and more efficient data collection

Besides the simplification regarding the data assimilation task, there may also be an advantage regarding the data requirements, even though this is challenging to assess. Nevertheless, the arguments supporting this assumption are:

- the fact that the dynamics of the controlled system is no longer restricted to a low-dimensional subspace, and
- the increased state space dimension.

To illustrate the latter point, let us consider a setting where we can collect data from arbitrary points in state space.

Example 4.1 (Distance-based error estimates). When thinking about error bounds, some surrogate modeling techniques such as radial basis functions (RBF) rely on a fill distance $h$, which is the maximum distance between an arbitrary point in state space to the closest point from the training data set\(^1\). According to [54], we can assume that $c > 0$ exists such that

$$\|\Phi_u - \Phi_u^r\| \leq \exp\left(-\frac{c}{h}\right)\|\Phi_u\|. \tag{9}$$

Now let us assume that $y(t) \in \mathbb{R}^{n}$ and $u(t) \in \mathbb{R}^{n_u}$, and that we want to obtain different surrogate models which cause nearly the same prediction error, i.e., for which the same error bound holds; a system with an augmented state space $\tilde{y}(t) = (y(t), u(t)) \in \mathbb{R}^{n+n_u}$ (that is hence autonomized), and a set of autonomous models for the QuaSiModO approach. To make the bounds comparable, assume that in Eq. (9), the constant $c$ and the norms of the right-hand side of the dynamics are identical for the autonomous and the control-augmented systems. Then, to obtain training data sets that have the same fill distance $h$, we use an equidistant grid of points with $\pi$ points in each spatial dimension and evaluate $\Phi$ once for each point. We thus require $\pi^n$ and $\pi^{n+n_u}$ data points for the autonomous systems and the autonomized control system, respectively, to achieve the same value for $h$. Assuming that we want to use the control bounds in the QuaSiModO approach, we require $2^n$ autonomous surrogate models (depending on the application, this number might be further reduced by a suitable choice of $V$). Now, comparing the number of required data points to achieve a comparable bound for the approximation error, we obtain

$$\frac{\# \text{ data points augmented}}{\# \text{ data points QuaSiModO}} = \frac{\pi^{n+n_u}}{2^n \pi^n} = \left(\frac{\pi}{2}\right)^n,$$

which grows very quickly for increasing $n_u$.

Remark 4.2. Not all surrogate modeling techniques have an exponential growth of the required data with the state space dimension. (In [34], for instance, the data requirements have a polynomial scaling with respect to the dimension, however under additional assumptions.) Nevertheless, this is a good indicator that the state space dimension can play a crucial role in the data requirements.

To further strengthen our point on data efficiency, we perform a numerical experiment as well, where we use ESNs (cf. Section 3.3), for the Lorenz system (cf. Section B.1) that are trained

---

\(^1\)The related concept of $M$-boundedness is used in error estimates for approximations in Reproducing Kernel Hilbert Space (RKHS) [53, Theorem 3.3].
as autonomous systems corresponding to the entries in \( V \), and

as a single system with an augmented state \( \tilde{y} = (y,u) \).

For a statistically relevant comparison, we repeat the following experiment 100 times:

- Create training data over 750 seconds using random inputs from \( V \) or \( U \), respectively;
- To study the dependency on the size of the training data set, use subsets of different sizes of the collected data (ranging from 100 to 15,000 data points);
- Use these data sets to train the ESNs for the two approaches; both approaches are trained with the identical number of samples, but not with the same ones, as \( U \neq V \);
- Compare the relative \( L^2 \) prediction error over a horizon of 2 seconds, averaged over 100 simulations with random control sequences from either \( U \) or \( V \) for both approaches. For control inputs from \( U \), the prediction by the QuaSiMo model is realized via linear interpolation of the autonomous systems.

The results are shown in Figure 6 for (a) the control affine Lorenz system (6) and (b) the nonlinear system (7). We observe a clear advantage in the control affine case, not surprisingly regardless of whether the control is selected from \( U \) or \( V \). In the nonlinear case, the interpolation error has a negative impact on the QuaSiMo model. However, on the one hand, the performance is still comparable. On the other hand, the sum-up rounding procedure yields controls from \( V \) anyway, such that this is the relevant control set for the comparison. For an LSTM-based surrogate model (cf. Figure 7), we observe a very similar behavior.
Limitations: Besides the above-mentioned advantages, there are also some limitations – or at least additional considerations – that need to be taken into account. Most importantly, we have to mention the task of choosing the finite set $V$. Whereas this is straightforward in the control affine case – and also in cases with a monotonous dependence on $u$ such as the $\cos(u)$ term in system (7) – this can become more challenging for nonlinear control dependencies. We thus face a challenging decision problem, and if we make a sub-optimal choice, the constant $M_1$ in the error bounds is non-zero (cf. Lemma 2.2) which results in a reduced control performance. This is caused by the fact that we have a smaller reachable set using only controls from $V$, which becomes also clear if we repeat the analysis from Fig. 6, but not using the control bounds. The result is demonstrated in Fig. 8, and we see that this has a significant impact on the tracking performance (orange lines). However, we would like to mention that many real systems can be modeled in a control affine manner or allow for an informed selection of the control values needed to satisfy the assumptions of Lemma 2.2.

If we consider $V$ to be the set of control bounds, we face another, yet related drawback. In this case, we have an exponential increase in autonomous systems that we need to learn, i.e., we have $2^n_u$ systems that we have to approximate. Even though we have demonstrated superiority in data requirements in some cases (cf. Example 4.1), this may still be computationally very complex. For control affine systems, this problem can be prevented if we allow extrapolation. This way, we can create $n_u + 1$ systems, corresponding to $u = 0$ and $n_u$ Euclidean basis vectors for the control space ($u_1 = (1,0,\ldots,0)$, $u_2 = (0,1,0,\ldots,0)$, etc.), which results in a linear complexity with the control dimension. It was already shown that this is a feasible approach for the Koopman operator framework [30].

5. 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, [36, 32]. 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 was supported in part 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. Pironi 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. Holde. 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. Thithof, 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):241202, 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. Syvatkovskiy, 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. Iazo. 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, 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.
[21] L. Grüne and J. Pannek. Nonlinear Model Predictive Control. Springer International Publishing, 2 edition, 2017.
[22] 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.
[23] K. Bieker, S. Peitz, S. L. Brunton, J. N. Kutz, and M. Dellnitz. Optimal rotary control of the cylinder wake using proper orthogonal decomposition reduced-order model. Physics of Fluids, 17:1–21, 2005.
[24] L. J. Proctor, S. L. Brunton, and J. N. Kutz. Dynamic mode decomposition with control. SIAM Journal on Applied Dynamical Systems, 15(1):142–161, 2015.
[25] S. L. Brunton and J. N. Kutz. Data-driven science and engineering: Machine learning, dynamical systems, and control. Cambridge University Press, 2019.
[26] S. Peitz and S. Klus. Koopman operator-based model reduction for switched-system control of PDEs. Automatica, 106:184 – 191, 2019.
[27] 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.
[28] 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.
[29] H. Lu and D. M. Tartakovsky. Predictive Accuracy of Dynamic Mode Decomposition. SIAM Journal on Scientific Computing, 42(3):1639–1662, 2020.
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, 55, 56],
- Dynamic Mode Decomposition / Koopman operator and generator [15, 14, 57, 58, 31, 16, 32],
- (Sparse) regression [59, 12, 13, 36],
- Stochastic modeling approaches [60, 61],
- Feed-forward neural networks [62],
- 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 operator $K$ of operators $\{K^t\}: \mathcal{F} \to \mathcal{F}$ with $\mathcal{F} = L^2(Y)$, which describes the evolution of the observable $f$ (see [63, 45, 64, 65]), is defined by

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

and the Koopman semigroup is generated by the Koopman generator $\mathcal{L}$ [16, 30]:

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

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 assume that we have either measurement or simulation data, written in matrix form as

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

where $z_i = f(y_i)$ and $\tilde{z}_i = f(\Phi^\Delta t(y_i))$ with a fixed step size $\Delta t$. 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

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

with $\psi$ 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

$$K^\top = \Psi_{\tilde{Z}} \Psi_Z^\top = (\Psi_Z \Psi_Z^\top)^{-1},$$

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

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

A.2. Proper Orthogonal Decomposition:

Consider a partial differential equation of the general form

$$\begin{aligned}
\dot{y}(x, t) &= g(y(x, t)), \quad (x, t) \in \Omega \times (t_0, t_f], \\
\alpha(x, t) \frac{\partial y(x, t)}{\partial n} + b(x, t)y(x, t) &= c(x, t), \quad (x, t) \in \Gamma \times (t_0, t_f], \\
y(x, t_0) &= y_0(x), \quad x \in \Omega,
\end{aligned} \tag{10}$$

where $\Omega$ 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 [66]. Since the state is space dependent, we have to take boundary conditions (BCs) into account, and the coefficients $\alpha(x, t)$, $b(x, t)$ 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 $\{\psi_i(x)\}_{i=1}^\ell$:

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

where $z_i(t)$ are the solution coefficients.
a system of nonlinear ODEs in the coefficients $z$:

$$\dot{z}(t) = g'(z(t)) \quad \text{and} \quad z_{t+1} = \Phi^t(z_t) = z_t + \int_{t_1}^{t_{t+1}} g'(z(t)) \, dt.$$ 

Note that this requires knowledge of the equations as well as a careful and often tedious treatment of boundary conditions [26] and nonlinearities [67]. 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 $\hat{Y} = [\hat{y}(t_0), \ldots, \hat{y}(t_N)]$, 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 [53].

Quite a number of extensions have been presented for the use of POD in PDE-constrained control, see, for instance, [24] 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 [68] or in [10, 69, 49].

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

$$r(k+1) = \sigma(W^{in}i(k) + W^{res}r(k) + W^{fb}o(k)),$$

$$o(k+1) = W^{out}r(k+1),$$

where $\sigma$ 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 $W^{in}i(k)$ 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., $\Phi : Y \to Y$ 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:

$$Z = [z_{t_0}, z_{t_1}, \ldots, z_{t_N}] \quad \text{with} \quad z_{t_k} = f(\Phi^{t_k}(y_0)). \quad (11)$$

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., [70, 71, 43, 72, 73, 49]. Furthermore, some papers discuss the use of ESN in an MPC context [74, 75]. 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.

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 [76] 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:

$$Z_{\text{in}} = [z_{t_k-d}, z_{t_k-d+1}, \ldots, z_{t_k}] \quad \text{and} \quad Z_{\text{out}} = z_{t_k+1},$$

(12)

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 $Z_{\text{in}}$ as a sequential input and should predict the behavior of the dynamical system for one time step into the future ($Z_{\text{out}}$). 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:

$$\min_{\alpha \in (0,1)^m} \sum_{i=0}^{p-1} (z_{i+1} - z_{i+1}^{\text{ref}})^\top Q (z_{i+1} - z_{i+1}^{\text{ref}})$$

(IV)

s.t. $z_{i+1} = \Phi^r(z_i, \alpha_i) = \sum_{j=1}^{m} \alpha_{i,j} \Phi^r_{u^j}(z_i)$ and $\sum_{j=1}^{m} \alpha_{i,j} = 1, \quad i = 0, 1, 2, \ldots,$

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

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

To complete the description in the results section, a detailed setting of the numerical setup is described in Table 2.

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

Table 2.: Lorenz system with EDMD surrogate model.
B.2. Control of the Burgers equation using Proper Orthogonal Decomposition:

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

\[ \dot{y}(x, t) - \frac{1}{Re} \Delta y(x, t) + y(x, t) \nabla y(x, t) = \sum_{j=1}^{5} w^j(t) \chi^j(x), \quad (x, t) \in [0, L] \times (t_0, t_e], \]

\[ y(0, t) = y(L, t) = 0, \quad t \in [t_0, t_e], \]

\[ y(x, t_0) = \begin{cases} 
1, \quad x \in \left(0, \frac{L}{2}\right], \\
0, \quad x \in \left(\frac{L}{2}, L\right), 
\end{cases} \]

which was also studied by Kunisch and Volkwein in their seminal work on POD-based control of PDEs [24]. We consider a domain of length \( L = 1 \) and a distributed control that is realized via indicator functions \( \chi_i \) with disjoint support:

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

with the aim to stabilize the system at \( y_{\text{ref}}(\cdot, t) = 0 \). 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.

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

B.3. 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 \( Re = 100 \) [77]. 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 [25] 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 4.: 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 | $\Delta t$ | 0.05      |
|               | $T_{\text{train}}$ | (2 *) 500 |
|               | $\#$ trajectories | 2         |
| Input         | Piecewise constant, random $u_i \in V$ |
| Surrogate model | $\Delta t$ | 0.1 |
|               | neurons per LSTM-cell | 500 |
|               | observable | $z = f(y) = (C_D, C_L)$ |
|               | delay coordinates | 15       |
| MPC           | $T_{\text{MPC}}$ | 20.0 |
|               | $p$        | 5          |
|               | $Q$        | $\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$ |
|               | $z^r_{\text{ref}}$ | $\sin(\frac{t}{2})$ |
|               | $\Delta t_{\text{SUR}}$ | 0.05 |

B.4. Control of the Fluidic Pinball using the Koopman operator:

As before, our aim is not to control the flow field, but the forces acting on the (this time three) cylinders without any knowledge of the flow field itself.

As the surrogate model we again use the Koopman operator, and the control aim is to force the lift to follow a predefined trajectory abs before. For detailed parameters of the setting see Table 5.

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

| Parameter     | Value       |
|---------------|-------------|
| System parameters | $Re$ | 200 |
| Quantization  | $U$        | $[-2, 2]^3$ |
|               | $V$        | $[-2, 2]^3$ |
|               | $m$        | 8          |
| Training data | $\Delta t$ | 0.05      |
|               | $T_{\text{train}}$ | 100 |
|               | $\#$ trajectories | 1         |
| Input         | Piecewise constant, random $u_i \in V$ |
| Surrogate model | $\Delta t$ | 0.25 |
|               | $\psi$     | Monomials up to degree 1 |
|               | observable | $z = f(y) = (C_{D,1}, C_{L,1}, C_{D,2}, C_{L,2}, C_{D,3}, C_{L,3})$ |
|               | delay coordinates | 1         |
| MPC           | $T_{\text{MPC}}$ | 60.0 |
|               | $p$        | 5          |
|               | $Q$        | $\text{diag}(0, 1, 0, 1, 0, 1)$ |
|               | $z^r_{\text{ref}}$ | piecewise constant |

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 [78, 48]:

$$\dot{y}(t) = \beta \frac{y(t - \tau)}{1 + y(t - \tau)^q} - \gamma y(t) + u(t)$$
with $\beta, \gamma, \eta > 0$. (13)

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 $\eta = 9.65$ [48]. Since this system has
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 [79], genetic algorithms [80] and different forms of neural networks [81, 82, 83, 84]. Another approach is via Echo State Networks (ESN), which were studied in [49, 85, 86].

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 [87, 88], 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 ($E_V(V) = 0$). The results in Figure 3 show that the system can be stabilized at 1.0.

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

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