

Inference of Continuous Linear Systems from Data with Guaranteed Stability

Pawan Goyal¹ Igor Pontes Duff² Peter Benner³

¹*Max Planck Institute of Dynamics of Complex Technical Systems*

Email: goyalp@mpi-magdeburg.mpg.de, ORCID: [0000-0003-3072-7780](https://orcid.org/0000-0003-3072-7780)

²*Max Planck Institute of Dynamics of Complex Technical Systems*

Email: pontes@mpi-magdeburg.mpg.de, ORCID: [0000-0001-6433-6142](https://orcid.org/0000-0001-6433-6142)

³*Max Planck Institute of Dynamics of Complex Technical Systems, Otto-von-Guericke University Magdeburg*

Email: benner@mpi-magdeburg.mpg.de, ORCID: [0000-0003-3362-4103](https://orcid.org/0000-0003-3362-4103)

Abstract: Machine-learning technologies for learning dynamical systems from data play an important role in engineering design. This research focuses on learning continuous linear models from data. *Stability*, a key feature of dynamic systems, is especially important in design tasks such as prediction and control. Thus, there is a need to develop methodologies that provide stability guarantees. To that end, we leverage the parameterization of stable matrices proposed in [Gillis/Sharma, Automatica, 2017] to realize the desired models. Furthermore, to avoid the estimation of derivative information to learn continuous systems, we formulate the inference problem in an integral form. We also discuss a few extensions, including those related to control systems. Numerical experiments show that the combination of a stable matrix parameterization and an integral form of differential equations allows us to learn stable systems without requiring derivative information, which can be challenging to obtain in situations with noisy or limited data.

Keywords: Continuous systems, linear dynamical models, stability, Lyapunov function, Runge-Kutta scheme.

Novelty statement:

- Learning stability-guaranteed continuous linear systems.
- Utilizing a stable matrix parameterization.
- Employing an integral form of differential equations to learn continuous systems, hence does not require estimating derivative information for data.
- Several numerical examples demonstrate the stability-guarantee of the learned models, which otherwise yield unstable models despite good data fit.

1 Introduction

Linear dynamical systems (LDS) are a widely used class of dynamical systems for describing underlying physical processes in science and engineering. These models are also utilized to understand the dynamic behavior of more complex nonlinear dynamical systems around operating points or equilibria. Consequently, LDS can be employed for optimization, and the design of feedback control laws. However, for complex physical processes, mathematical models are often hard to build from first-principles and physical parameters. Thus, there is a growing interest in developing mathematical models using experimental

data, which is widely available due to advances in measurement technology. This work aims at learning continuous-time LDS from data while ensuring that the learned models are stable, which is necessary for goals such as long-term predictions.

Significant research has been dedicated to learning discrete-time linear dynamic systems (LDS) from time-domain data in the literature. One method that has gained widespread attention is dynamic mode decomposition (DMD), due to its accuracy in predicting dynamic systems, its connection to the Koopman operator [1], and its ease of implementation [2]. DMD was originally developed in the field of fluid dynamics [3], and several variations have been proposed since, such as compressed DMD [4], extended DMD [5], and DMD with control [6]. In the field of systems and control, methods for learning LDS from input-output data include the eigensystem realization algorithm [7], and subspace methods [8, 9]. There are also several methods for learning LDS from frequency data measurements, such as the Loewner framework [10], vector fitting [11, 12], and the AAA algorithm [13]. In addition, the operator inference method [14] was proposed for learning linear or polynomial dynamics from state-space data in continuous-time systems. This method can be viewed as a continuous-time version of DMD, at least for linear systems. Indeed, both involve solving a least squares problem from data. However, a key difference between both is that operator inference utilizes the derivative data, while DMD uses the shift state data.

Many physical processes are typically stable, meaning their state variables are well-behaved and globally bounded. These processes are often described by a set of stable differential equations, which is necessary for an accurate representation of the physics, as well as for numerical computations. However, stability is often neglected in many frameworks for learning dynamical systems. Consequently, unstable learned dynamical systems may be inferred even if the original dynamics are stable, leading to failure in predicting long-term behavior [15]. Stability constraints based on eigenvalues can be applied to an optimization problem, but this results in a non-convex constraint optimization problem that does not scale well with complexity. In the literature, the identification of linear stable discrete-time systems using matrix inequality constraints has been addressed in [16, 17] in the context of subspace identification methods. In DMD perspectives, a nonsmooth optimization method was suggested in [18] to enforce stability. Very recently, using parametrization for discrete stable matrices [19], the authors guarantee the stability of discrete linear dynamical systems, see [20]. More recently, operator inference with matrix inequality constraints was discussed in [21] for a particular structured case where the linear matrix or operator is symmetric and negative definite.

In this study, we propose a framework for learning continuous-time LDS enforcing stability by construction. To achieve this, we exploit the stable matrix parameterization proposed in [22], which guarantees stability for continuous-time LDS. This allows us to formulate the identification problem as an unconstrained optimization loss function. Hence, this methodology does not require any stability constraints, e.g., by means of eigenvalues or matrix inequalities, and therefore it scales to more significant complexity problems. Additionally, we use an integral form of differential equations to avoid the need to estimate derivative information, which can be challenging to obtain for noisy or scarce data. This idea is highly inspired by the recent advances in Neural ODEs [23] and the use of integrating schemes in learning of dynamical systems [24–26]. Through numerical experiments, we illustrate that the new methodology (sLSI) yields systems that are guaranteed to be stable. In contrast, the classical methods—that do not enforce this property—can produce unstable models. Moreover, we show that the inference methodology is suitable for high-dimensional data by combining it with a compression step to obtain a low-dimensional data representation.

The remainder of this paper is organized as follows. In Section 2, we provide a brief overview of a method for learning continuous-time LDS through a DMD-like procedure. The learned models are obtained through the solution of an unconstrained least-squares problem using the available data. In Section 3, we recall the stability concept for continuous-time LDS and present the characterization for stable matrices, proposed in [22], which forms the basis for our proposed methodology. We then formulate a suitable optimization problem for learning LDS, enforcing stability. We further show how to rewrite the optimization problem as an unconstrained one. In Section 4, we discuss incorporating integration schemes in learning continuous systems to avoid the necessity of derivative information, which yields a robust performance of sLSI under noisy and limited data conditions. In Section 5, we show how the methodology can be adapted for high-dimensional data and discuss a few possible extensions. Finally,

Section 6 presents numerical experiments illustrating that sLSI guarantees to produce stable models, and Section 7 provide a summary of our findings.

2 Continuous-time Linear Systems and its Inference

Throughout this work, we consider continuous-time linear dynamical systems as follows:

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{A}\mathbf{x}(t), \quad \mathbf{x}(0) = \mathbf{x}_0, \quad (1)$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ is the state variable, $\mathbf{x}_0 \in \mathbb{R}^n$ is the initial condition, and $\mathbf{A} \in \mathbb{R}^{n \times n}$ is a matrix. The inclusion of control variables in such systems is discussed in Section 5. Our goal is to learn a stable linear dynamical system from state measurements. The DMD algorithm finds the linear operator $\mathbf{A} \in \mathbb{R}^{n \times n}$ that best fits the available data in the desired norm. In this work, we will focus on the analog to the DMD algorithm for continuous-time systems as (1). This methodology is also referred to in the literature as *operator inference* with only linear term [14]. For continuous-time systems, we assume that snapshots of the state $\mathbf{x}(t)$ and the state derivative $\frac{d}{dt}\mathbf{x}(t)$ are available for time instances $\{t_1, \dots, t_N\}$. These snapshots are collected in the snapshot matrices as follows:

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}(t_0) & \mathbf{x}(t_1) & \dots & \mathbf{x}(t_N) \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times N}, \quad \text{and} \quad \dot{\mathbf{X}} = \begin{bmatrix} | & | & & | \\ \frac{d}{dt}\mathbf{x}(t_0) & \frac{d}{dt}\mathbf{x}(t_1) & \dots & \frac{d}{dt}\mathbf{x}(t_N) \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times N}. \quad (2)$$

In terms of these matrices, the operator \mathbf{A} can be inferred via a least squares fitting. More precisely, the matrix \mathbf{A} is the solution to the optimization problem

$$\mathbf{A} = \arg \min_{\tilde{\mathbf{A}} \in \mathbb{R}^{n \times n}} \left\| \dot{\mathbf{X}} - \tilde{\mathbf{A}}\mathbf{X} \right\|_F, \quad (3)$$

where $\|\cdot\|_F$ is the Frobenius norm. Additionally, the minimal-norm solution for the previous optimization problem can be given as follows:

$$\mathbf{A} = \mathbf{X}^\dagger \dot{\mathbf{X}},$$

where \mathbf{X}^\dagger denotes the pseudo-inverse of the snapshot matrix \mathbf{X} .

It is worth noticing that the learned linear model using (3) is not guaranteed to be stable, even if the available data was obtained from an underlying stable system. Another drawback of the linear regression (3) is that it requires knowledge of the derivative of the state. Indeed, it can be difficult to accurately estimate this derivative information when dealing with noisy or limited data. To circumvent this issue, the authors e.g., in [23–26] have suggested reformulating the problem in an integral form, which has been shown to provide robust performance. In this work, we also employ a similar concept to avoid computation of the derivative information which shall be discussed more in Section 4.

3 Stability-guaranteed Learning

We begin by describing the characterization of stability for linear dynamical systems (1) which will be used on this work. It is known that a linear system (1) is stable if and only if all of the eigenvalues of the matrix \mathbf{A} are in the closed left half complex plane and all eigenvalues on the imaginary axis are semi-simple. Additionally, the linear system (1) is asymptotically stable if and only if all of the eigenvalues of the matrix \mathbf{A} are in the closed left half complex plane.

Parametrization of stable matrices: A recent characterization of stable matrices is discussed in [22, Lemma 1]. In this work, the authors show that every stable matrix \mathbf{A} can be written as:

$$\mathbf{A} = (\mathbf{J} - \mathbf{R})\mathbf{Q} \quad (4)$$

where $\mathbf{J} = -\mathbf{J}^\top$ is a skew symmetric matrix, $\mathbf{R} = \mathbf{R}^\top \geq 0$ is symmetric positive semi-definite, and $\mathbf{Q} = \mathbf{Q}^\top > 0$ is symmetric positive definite. The authors discussed such parametrization to determine the distance to stability of any given matrix.

It is worthwhile mentioning that the parametrization (4) encodes a Lyapunov function for the underlying linear dynamical system, which is stated in the following lemma.

Lemma 1. *Consider a linear dynamical system as in (1) where \mathbf{A} takes the form given in (4). Then, the quadratic function*

$$\mathbf{V}(\mathbf{x}(t)) = \frac{1}{2} \mathbf{x}(t)^\top \mathbf{Q} \mathbf{x}(t)$$

is a Lyapunov function of the system.

Proof. Note that

$$\begin{aligned} \frac{d}{dt} \mathbf{V}(\mathbf{x}(t)) &= \frac{1}{2} \left(\frac{d}{dt} \mathbf{x}(t) \right)^\top \mathbf{Q} \mathbf{x}(t) + \frac{1}{2} \mathbf{x}(t)^\top \mathbf{Q} \left(\frac{d}{dt} \mathbf{x}(t) \right) \\ &= \mathbf{x}(t)^\top \mathbf{Q} \left(\frac{d}{dt} \mathbf{x}(t) \right) = \mathbf{x}(t)^\top \mathbf{Q} \mathbf{A} \mathbf{x}(t) \\ &= \mathbf{x}(t)^\top \mathbf{Q} (\mathbf{J} - \mathbf{R}) \mathbf{Q} \mathbf{x}(t) \\ &= \tilde{\mathbf{x}}(t)^\top (\mathbf{J} - \mathbf{R}) \tilde{\mathbf{x}}(t) \quad \text{with } \tilde{\mathbf{x}}(t) = \mathbf{Q} \mathbf{x}(t) \\ &= \tilde{\mathbf{x}}(t)^\top (\mathbf{J} - \mathbf{R}) \tilde{\mathbf{x}}(t) \\ &= \tilde{\mathbf{x}}(t)^\top \mathbf{J} \tilde{\mathbf{x}}(t) - \tilde{\mathbf{x}}(t)^\top \mathbf{R} \tilde{\mathbf{x}}(t) \leq 0, \end{aligned}$$

because $\mathbf{R} \geq 0$. □

Remark 1. *We also highlight that if \mathbf{R} is a positive definite matrix, $\frac{d}{dt} \mathbf{V}(\mathbf{x}(t)) < 0$. As a consequence, the matrix \mathbf{A} is asymptotically stable.*

The parametrization in (4) enables us to characterize every continuous-time stable dynamical system. Instead of relying on constraints involving matrix inequalities or eigenvalues, we will utilize this parametrization to ensure the stability of the learned dynamical system.

Stability informed learning As noted earlier, the matrix decomposition (4) guarantees that the matrix \mathbf{A} is stable. So, we will use the stable matrix parametrization to guarantee that the learned linear systems are stable. To this aim, we formulate our objective function as follows:

$$\begin{aligned} (\mathbf{J}, \mathbf{R}, \mathbf{Q}) &= \arg \min_{\tilde{\mathbf{J}}, \tilde{\mathbf{R}}, \tilde{\mathbf{Q}}} \left\| \dot{\mathbf{X}} - (\tilde{\mathbf{J}} - \tilde{\mathbf{R}}) \tilde{\mathbf{Q}} \mathbf{X} \right\|_F, \\ &\text{subject to } \tilde{\mathbf{J}} = -\tilde{\mathbf{J}}^\top, \tilde{\mathbf{R}} = \tilde{\mathbf{R}}^\top \geq 0, \tilde{\mathbf{Q}} = \tilde{\mathbf{Q}}^\top > 0. \end{aligned} \quad (5)$$

Once we have the optimal value for the tuple $(\mathbf{J}, \mathbf{R}, \mathbf{Q})$, we can construct the stable matrix \mathbf{A} as $(\mathbf{J} - \mathbf{R}) \mathbf{Q}$. Note that the optimization problem (5) includes constraints on the matrices $\tilde{\mathbf{J}}, \tilde{\mathbf{R}}, \tilde{\mathbf{Q}}$. With an appropriate parameterization for these matrices, we can re-formulate the optimization problem (5) as an unconstrained one. To that end, first, note that any skew-symmetric matrix $\tilde{\mathbf{J}}$ can be parameterized as

$$\tilde{\mathbf{J}} = \bar{\mathbf{J}} - \bar{\mathbf{J}}^\top, \quad (6)$$

where $\bar{\mathbf{J}} \in \mathbb{R}^{n \times n}$ is a square matrix. Moreover, to remove the symmetric positive (semi)definite constraints, we parameterize $\tilde{\mathbf{Q}}$ and $\tilde{\mathbf{R}}$ as

$$\tilde{\mathbf{Q}} = \bar{\mathbf{Q}} \bar{\mathbf{Q}}^\top \quad \text{and} \quad \tilde{\mathbf{R}} = \bar{\mathbf{R}} \bar{\mathbf{R}}^\top, \quad (7)$$

for any given matrices $\bar{\mathbf{Q}}$ and $\bar{\mathbf{R}} \in \mathbb{R}^{n \times n}$. Notice that the parametrization (7) only guarantees that $\tilde{\mathbf{Q}}$ is positive semidefinite unless the matrix $\bar{\mathbf{Q}}$ is full rank. In order to avoid the rank constraint, we will relax

the problem, where we allow $\tilde{\mathbf{Q}}$ to be semidefinite. These parametrizations enable us to re-formulate the objective constraint function (5) as the unconstrained objective function:

$$(\bar{\mathbf{J}}, \bar{\mathbf{R}}, \bar{\mathbf{Q}}) = \arg \min_{\bar{\mathbf{J}}, \bar{\mathbf{Q}}, \bar{\mathbf{R}}} \left\| \dot{\mathbf{X}} - (\bar{\mathbf{J}} - \bar{\mathbf{J}}^\top - \bar{\mathbf{R}}\bar{\mathbf{R}}^\top)\bar{\mathbf{Q}}\bar{\mathbf{Q}}^\top \mathbf{X} \right\|_F. \quad (8)$$

This then results in a stable matrix $\mathbf{A} = (\bar{\mathbf{J}} - \bar{\mathbf{J}}^\top - \bar{\mathbf{R}}\bar{\mathbf{R}}^\top) \bar{\mathbf{Q}}\bar{\mathbf{Q}}^\top$.

4 Un-rolling Integrating Schemes to Avoid Derivative Information Requirement

One of the challenges to learning continuous-time LDS (see, e.g., (8)) is the requirement for derivative information, which can be difficult to accurately estimate from noisy and scarce data. As an alternative, researchers have explored the use of integral forms of differential equations and numerical techniques to learn continuous systems [24–26]. With a similar spirit, we focus on embedding a Runge-Kutta fourth-order integration scheme to predict the dependent variable \mathbf{x} at the next time step. Given a stable linear system (1) and the state \mathbf{x} at time t_i , denoted by $\mathbf{x}(t_i)$, we can estimate $\mathbf{x}(t_{i+1})$ as follows:

$$\mathbf{x}(t_{i+1}) = \Phi_{\text{dt}}(\mathbf{A}, \mathbf{x}(t_i)) := \mathbf{x}(t_i) + \frac{\text{dt}}{6}(\mathbf{h}_1 + 2\mathbf{h}_2 + 2\mathbf{h}_3 + \mathbf{h}_4), \quad (9)$$

where

$$\mathbf{h}_1 = \mathbf{A}\mathbf{x}(t_i), \quad \mathbf{h}_2 = \mathbf{A}\left(\mathbf{x}(t_i) + \frac{\text{dt}}{2}\mathbf{h}_1\right), \quad \mathbf{h}_3 = \mathbf{A}\left(\mathbf{x}(t_i) + \frac{\text{dt}}{2}\mathbf{h}_2\right), \quad \mathbf{h}_4 = \mathbf{A}(\mathbf{x}(t_i) + \mathbf{h}_3). \quad (10)$$

We can then modify the objective function (8) as follows:

$$\min_{\bar{\mathbf{J}}, \bar{\mathbf{R}}, \bar{\mathbf{Q}}} \sum_i \|\mathbf{x}(t_{i+1}) - \Phi_{\text{dt}}(\bar{\mathbf{A}}, \mathbf{x}(t_i))\|, \quad \text{with } \bar{\mathbf{A}} = (\bar{\mathbf{J}} - \bar{\mathbf{J}}^\top - \bar{\mathbf{R}}\bar{\mathbf{R}}^\top) \bar{\mathbf{Q}}\bar{\mathbf{Q}}^\top, \quad (11)$$

thus avoiding the necessity of derivative information. Consequently, we have the optimization problem (11) whose solutions are stable by construction. This proposed methodology is called *stable linear system inference* (sLSI) throughout this paper.

Moreover, we highlight that one can use any other numerical integration or higher-order schemes, as well as multi-step unfolding schemes. In fact, the concept of adjoint sensitivity, as proposed in neural ODEs [23], can be employed to perform efficient computation (in term of memory cost) through any numerical integrator, but it often comes with more CPU cost.

5 Further Considerations

In this section, we discuss a few possible considerations and extensions for sLSI.

5.1 Low-dimensional representation and analog to compressed DMD

Collecting data from physical systems often results in high-dimensional data with thousands to millions of degrees of freedom. As a result, building models from such high-dimensional data are computationally expensive. However, mostly these high-dimensional data reside in a low-dimensional subspace, so we can obtain a compact, low-dimensional representation by projecting onto such a subspace. Techniques such as principal component analysis (PCA), proper-orthogonal decomposition (POD), and autoencoders can be used to achieve this goal. Here, we describe the POD approach, which is commonly used in the reduced-order modeling community.

Singular-value decomposition (SVD) is a key element of POD and allows us to determine a suitable low-dimensional subspace. To that aim, let us consider a data matrix $\mathbf{X} \in \mathbb{R}^{n \times N}$, where n is the dimension of

the data, and N is the number of measurements. We can determine the dominant subspace by performing the SVD of \mathbf{X} , i.e.,

$$\mathbf{X} := \mathbf{U}\Sigma\mathbf{V}^\top, \quad (12)$$

where the dominating subspaces are contained in \mathbf{U} , and the singular values stored in Σ determine their significance. $\Sigma := \text{diag}(\sigma_1, \dots, \sigma_n)$ is a diagonal matrix with $\sigma_{i+1} \geq \sigma_i$. By using the subspace spanned by the r most dominant left singular vectors, denoted by \mathbf{U}_r , we can obtain a low-dimensional representation as follows:

$$\mathbf{X}_r = \mathbf{U}_r^\top \mathbf{X}, \quad (13)$$

where $\mathbf{X}_r \in \mathbb{R}^{r \times N}$. Moreover, we can reconstruct our high-dimensional data \mathbf{X} , indicated by $\tilde{\mathbf{X}}$, using \mathbf{X}_r , e.g., $\tilde{\mathbf{X}} = \mathbf{U}_r \mathbf{X}_r$. Additionally, the information loss due to projection and re-projection can be measured as follows:

$$\|\mathbf{X} - \tilde{\mathbf{X}}\|_2 \leq \sum_{r+1}^n \sigma_i. \quad (14)$$

Using the above relation as an indicator, we can choose a suitable r . Finally, once we have a low-dimensional representation of the data \mathbf{X}_r , we can use the method **sLSI** described in the previous sections to learn linear dynamical systems in the low-dimensional space, thus avoiding numerical computations in high dimensions, which may be too expensive.

5.2 Systems controlled by external inputs

Many dynamic processes are controlled by external inputs, which can result in different dynamic behavior when varied. To account for this in our modeling, we consider the following model hypothesis:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t). \quad (15)$$

Given the trajectories $\mathbf{x}(t)$ subject to various control inputs, we aim to learn (15) while maintaining stability constraints. To this aim, we collect snapshots of the input functions as follows

$$\mathbf{U} = \begin{bmatrix} | & | & \dots & | \\ \mathbf{u}(t_0) & \mathbf{u}(t_1) & \dots & \mathbf{u}(t_N) \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{1 \times N}, \quad (16)$$

and tailor the objective function in (8) to include the control input. Hence, the resulting objective for inferring control systems is given by

$$(\bar{\mathbf{J}}, \bar{\mathbf{R}}, \bar{\mathbf{Q}}, \bar{\mathbf{B}}) = \arg \min_{\bar{\mathbf{J}}, \bar{\mathbf{Q}}, \bar{\mathbf{R}}, \bar{\mathbf{B}}} \|\bar{\mathbf{X}} - (\bar{\mathbf{J}} - \bar{\mathbf{J}}^\top - \bar{\mathbf{R}}\bar{\mathbf{R}}^\top)\bar{\mathbf{Q}}\bar{\mathbf{Q}}^\top \bar{\mathbf{X}}\bar{\mathbf{B}}\mathbf{U}\|_F, \quad (17)$$

and the learned dynamical system with control is guaranteed to be stable.

With some slight modifications, we can easily extend the discussion from Subsection 3 and incorporate numerical schemes to avoid the derivative data. Specifically, we modify (9) as follows:

$$\mathbf{x}(t_{i+1}) \approx \Phi_{\text{dt}}^c(\mathbf{A}, \mathbf{B}, \mathbf{x}(t_i), \mathbf{u}) := \mathbf{x}(t_i) + \frac{\text{dt}}{6} (\mathbf{h}_1 + 2 \cdot \mathbf{h}_2 + 2 \cdot \mathbf{h}_3 + \mathbf{h}_4), \quad \text{with } \text{dt} = t_{i+1} - t_i \quad (18)$$

$$\begin{aligned} \mathbf{h}_1 &= \mathbf{A}\mathbf{x}(t_i) + \mathbf{B}\mathbf{u}(t_i), & \mathbf{h}_2 &= \mathbf{A} \left(\mathbf{x}(t_i) + \frac{\text{dt}}{2}\mathbf{h}_1 \right) + \mathbf{B}\mathbf{u} \left(t_i + \frac{\text{dt}}{2} \right) \\ \mathbf{h}_3 &= \mathbf{A} \left(\mathbf{x}(t_i) + \frac{\text{dt}}{2}\mathbf{h}_2 \right) + \mathbf{B}\mathbf{u} \left(t_i + \frac{\text{dt}}{2} \right), \text{ and } & \mathbf{h}_4 &= \mathbf{A} (\mathbf{x}(t_i) + \mathbf{h}_3) + \mathbf{B}\mathbf{u}(t_i + \text{dt}). \end{aligned} \quad (19)$$

The objective function (17) then changes as follows:

$$\min_{\bar{\mathbf{J}}, \bar{\mathbf{R}}, \bar{\mathbf{Q}}, \bar{\mathbf{B}}} \sum_i \|\mathbf{x}(t_{i+1}) - \Phi_{\text{dt}}^c(\bar{\mathbf{A}}, \bar{\mathbf{B}}, \mathbf{x}(t_i))\|_F, \quad \text{with } \bar{\mathbf{A}} = (\bar{\mathbf{J}} - \bar{\mathbf{J}}^\top - \bar{\mathbf{R}}\bar{\mathbf{R}}^\top) \bar{\mathbf{Q}}\bar{\mathbf{Q}}^\top. \quad (20)$$

This allows us to obtain a stable control system realization by construction and avoid the need to compute derivative information by combining it with a numerical integrator.

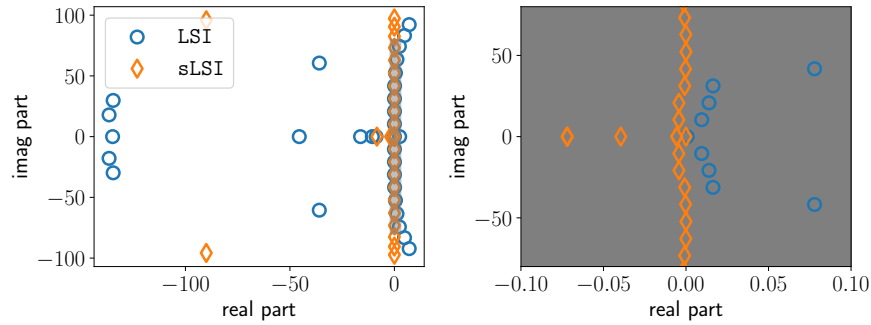


Figure 1: Flow-past cylinder example: A comparison of the eigenvalues of the learned models, which clearly shows **sLSI** guarantees that all of the eigenvalues are in the left half-plane. The right plot shows a zoomed-in version of the left plot centered around the origin.

5.3 Stable linear models for predefined observers

It is well-known that nonlinear dynamic systems can be represented as linear systems in an infinite-dimensional Hilbert space, as described in [27]. However, in order to use these models for engineering purposes, it is necessary to approximate this infinite-dimensional space. The extended DMD method proposed in [5] aims to design observers that can accurately approximate infinite-dimensional operators using finite-dimensional linear operators. However, the methodology to learn stable models is maintained for newly designed observers. Hence, the method for learning stable realizations described in the previous section (**sLSI**) can be readily applied in the context of extended DMD to learn stable linear operators for continuous-time systems.

6 Numerical Experiments

In this section, we evaluate the performance of **sLSI** through three numerical examples. We compare **sLSI** with the recently proposed methodology in [26] which allows inference for LDS using numerical integration un-rolling to avoid the need for derivative computation but without any enforcing of stability of the learned operator. We refer to it as *linear system inference* (**LSI**). All experiments are conducted using PyTorch, with 20,000 updates performed using the Adam optimizer [28] and a triangular cyclic learning rate ranging from 10^{-6} to 10^{-2} . The coefficients of all matrices are randomly generated from a Gaussian distribution with a mean of 0 and standard deviation of 0.1.

6.1 Unsteady flow for cylinder-wake example

In this example, we study the flow past a cylinder, a widely used benchmark problem in the literature (see, for instance, [2]). The Reynolds number is set to $\text{Re} = 100$ and we have 151 vorticity measurements, which exhibits periodic oscillations. The data was collected on a grid of 199×449 . Since the data can be represented very well in a low-dimensional manifold, we compress the data using 31 dominant POD modes, consequently reducing the complexity of the optimization problem and so for the learned model. We learn continuous-time linear models using **sLSI** and **LSI**. To assess the performance, we first compare the eigenvalues of these learned systems in Figure 1, which clearly indicates **sLSI** yields a model with all eigenvalues in the left-half plane. In contrast, **LSI** yields a model, possessing some unstable eigenvalues. Furthermore, we compute the inferred eigenfunctions from the model obtained with **sLSI**. In Figure 2, we plot the real part of the four dominant eigenfunctions showing the dominant flow dynamics as expected.

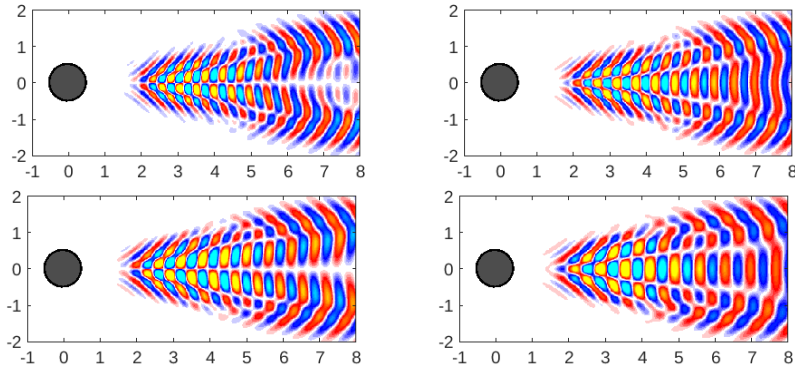


Figure 2: Flow-past cylinder example: The plots show the real part of the dominant eigenfunctions from the learned model via **sLSI**.

6.2 Transporting flow

Next, we consider a two-dimensional transporting flow whose x and y -directions velocities, denoted by u and v , are given by

$$\begin{aligned} u(x, y, t) &= \sin(5(t - x)) \sin(5(t - y)), \\ v(x, y, t) &= \cos(5(t - x)) \cos(5(t - y)). \end{aligned} \quad (21)$$

We consider the spatial domain $[-1.5, 1.5] \times [-1.5, 1.5]$ and 200 equidistant points in both directions to sample the flow. Then, we collect 100 data points in time interval $[0, 5]$, and the flow at time $t = 0$ is shown in Figure 3.

The data in this study has a high dimensionality, with a total of $2 \cdot 40,000$ variables. To address this, we first aim to find a low-dimensional representation of the data using POD. We discover that the data can be described by only three dominant modes (up to the machine precision). Hence, we project the high-dimensional data onto these dominant subspaces and use them to learn linear dynamical models with **sLSI** and **LSI**. The eigenvalues of these models are depicted in Figure 4. The figure clearly demonstrates that the proposed methods ensure stability. In contrast, when stability is not imposed, then the resulting systems can be unstable, which is the case for this example. Moreover, we show the eigenfunctions related to the velocities in the x and y -directions in Figure 5, showing transportation of the flow in a specific direction.

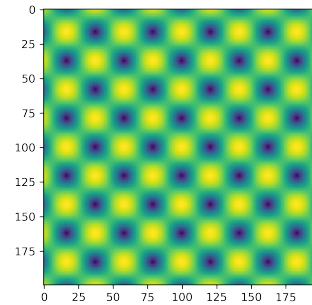


Figure 3: Transporting flow: The plot shows the magnitude of the velocity at time $t = 0$.

6.3 Burgers' equations

In our last example, we consider the one-dimensional Burgers' equation with the following initial and boundary:

$$\begin{aligned} v_t + v v_\zeta &= +\mu v_{\zeta\zeta} \quad \text{in } (0, 1) \times (0, T), \\ v_\zeta(0, \cdot) &= 0, \\ v_\zeta(1, \cdot) &= 0, \\ v(\zeta, 0) &= v_0(\zeta) \quad \text{in } (0, 1), \end{aligned} \quad (22)$$

where v_t and v_ζ denote the derivative of v with respect to time t and space ζ , and $v_{\zeta\zeta}$ denotes the second derivative of v with respect to ζ . The PDE is discretized using 1,000 equidistant grid points. We collect 500 data points for a given initial condition in the time interval $[0, 1]$ s. Furthermore, we assume the initial conditions to be $v_0(\zeta) = 1 + \sin((2f\zeta + 1)\pi)$, where $f = [1, 1.25, \dots, 4.75, 5.0]$. Hence, we have the data

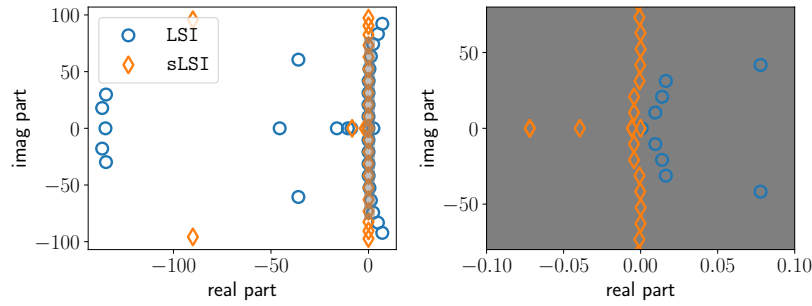


Figure 4: Transport flow example: A comparison of the eigenvalues of the learned models, which clearly shows **sLSI** guarantees to have all eigenvalues in the left half-plane. The right plot shows a zoomed-in version of the left plot centered around the origin.

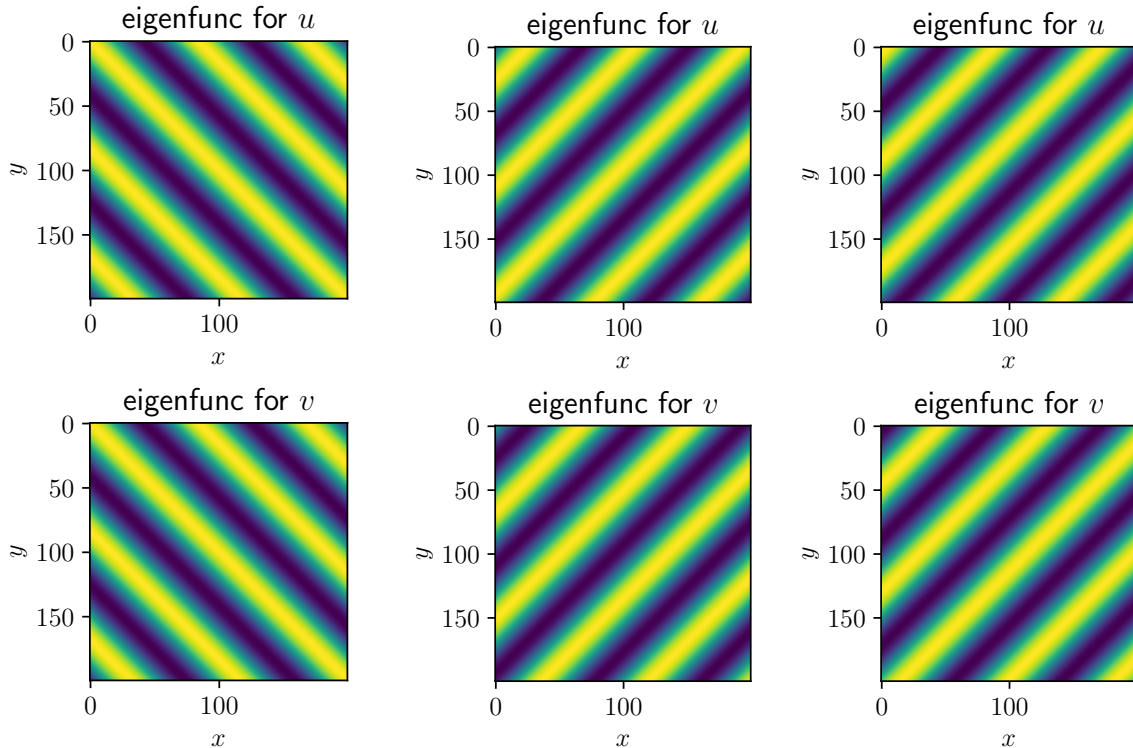


Figure 5: Transport flow example: The plots show the dominant eigenfunctions for u and v , i.e., the velocities in the x and y -directions, respectively.

corresponding to in-total 17 different initial conditions. Next, we split the data into training and testing, and for the testing, we take the initial conditions with $f = [1.75, 2.75, 3.75]$, and the rest are considered for learning.

First of all, since the data are high-dimensional, we seek to determine a low-dimensional representation of the training data by projecting it using the dominant POD basis. We consider 21 modes, which capture more than 99.9% of the total energy. Next, we employ **sLSI** and **LSI** to obtain linear models. For comparison, we first look at the eigenvalues of both learned models, which is shown in [Figure 6](#).

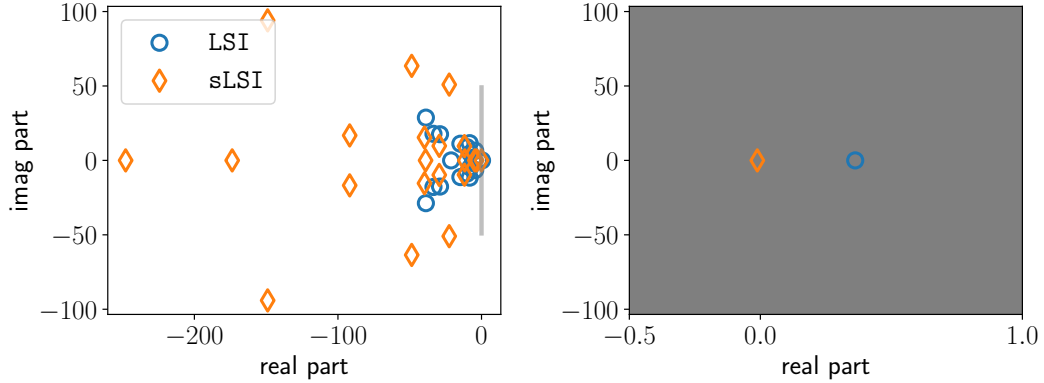


Figure 6: Burgers' equation: A comparison of the eigenvalues of the learned models, which clearly shows that **sLSI** guarantees not to have any eigenvalues in the right half-plane.

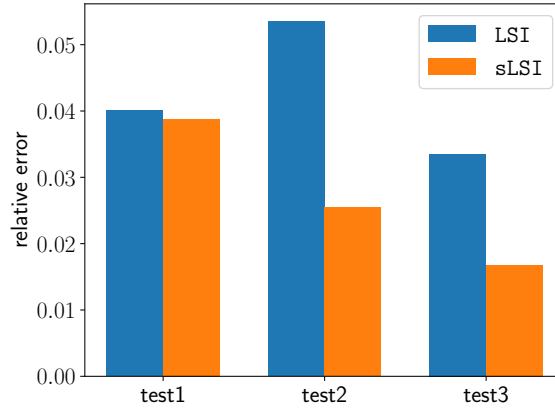


Figure 7: Burgers' equation: A performance of the learned models on the test data.

It indicates the guaranteed stability of **sLSI**, whereas **LSI** has an eigenvalue in the right-half plane, indicating potential instability in long-term prediction.

Moreover, we compare the learned models on the left-out initial conditions for testing. For this, we first project the initial condition onto a 21-dimensional subspace using the same POD basis as used for the training data. We integrate the system for the time-interval $[0, 1]$ and take 500 steps in this interval. For comparison, we examine the relative L_2 -error for all three testing initial conditions, shown in [Figure 7](#). It shows that **sLSI** even performs well, up to a factor of two, in the time-domain simulations despite having stability constraints on the linear operator while inferring. We also compare the time-domain simulations for one of the test cases in [Figure 8](#). It is done by first integrating the low-dimensional model, which is followed by re-projecting onto the higher-dimensional using the POD basis.

7 Conclusions

We discussed a method for learning continuous-time linear dynamical systems (**sLSI**) with two key features. First, the **sLSI** approach makes use of the parameterization of stable matrices, ensuring the stability of the inferred systems through direct encoding. The second feature of the proposed method is the use of an integral form for learning continuous-time systems, thus eliminating the requirement for ex-

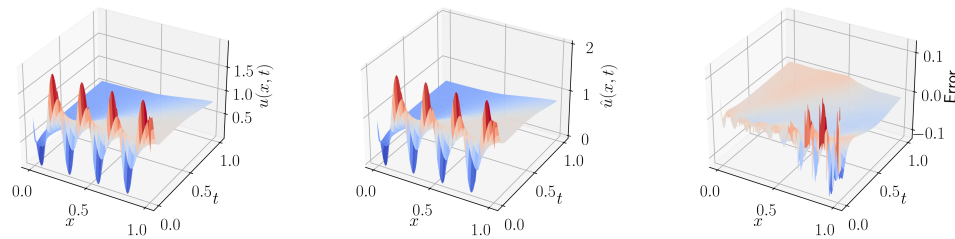


Figure 8: Burgers’ equation: A comparison of the time-domain simulation of **sLSI** on a test data with $f = 3.75$. The left figure is the ground truth; the middle is **sLSI**, and the right one shows the error between them.

PLICIT derivative computation. Several extensions are presented as well. Numerical examples demonstrate the stability features of the learned models. In contrast, this characteristic can be broken if stable-matrix parameterization is not applied. Moreover, the performance of the learned models using **sLSI** can lead to an improvement of up to a factor of two in time-domain simulations.

This work opens up the possibility for further research. As emphasized, having an adequate parameterization, which directly encodes desired properties, guarantees to have those properties by construction. Therefore, it would be intriguing to explore the possibility of enforcing additional physical laws, such as conservation of mass and energy, through the use of appropriate parameterizations. An extension to parametric varying systems could also be promising.

References

- [1] C. W. Rowley, I. Mezic, S. Bagheri, P. Schlatter, D. S. Henningson, Spectral analysis of nonlinear flows, *Journal of Fluid Mechanics* 641 (2009) 115–1127. doi:10.1017/S0022112009992059.
- [2] J. N. Kutz, S. L. Brunton, B. W. Brunton, J. L. Proctor, *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*, Society of Industrial and Applied Mathematics, Philadelphia, USA, 2016. doi:10.1137/1.9781611974508.
- [3] P. J. Schmid, Dynamic mode decomposition of numerical and experimental data, *J. Fluid Mech.* 656 (2010) 5–28. doi:10.1017/S0022112010001217.
- [4] S. L. Brunton, J. L. Proctor, J. H. Tu, J. N. Kutz, Compressed sensing and dynamic mode decomposition, *J. Comput. Dynamics* 2 (2) (2015) 165.
- [5] M. O. Williams, I. G. Kevrekidis, C. W. Rowley, A data-driven approximation of the Koopman operator: Extending dynamic mode decomposition, *J. Nonlinear Science* 25 (6) (2015) 1307–1346.
- [6] J. L. Proctor, S. L. Brunton, J. N. Kutz, Dynamic mode decomposition with control, *SIAM J. Appl. Dyn. Syst.* 15 (1) (2016) 142–161. doi:10.1137/15M1013857.
- [7] J.-N. Juang, R. S. Pappa, An eigensystem realization algorithm for modal parameter identification and model reduction, *J. Guidance, Control, and Dynamics* 8 (5) (1985) 620–627.
- [8] P. Van Overschee, B. De Moor, N4SID: Subspace algorithms for the identification of combined deterministic-stochastic systems, *Automatica* 30 (1) (1994) 75–93.
- [9] M. Viberg, Subspace-based methods for the identification of linear time-invariant systems, *Automatica* 31 (12) (1995) 1835–1851.
- [10] A. J. Mayo, A. C. Antoulas, A framework for the solution of the generalized realization problem, *Linear Algebra Appl.* 425 (2–3) (2007) 634–662. doi:10.1016/j.laa.2007.03.008.

-
- [11] B. Gustavsen, A. Semlyen, Rational approximation of frequency domain responses by vector fitting, *IEEE Trans. Power Del.* 14 (3) (1999) 1052–1061. doi:10.1109/61.772353.
- [12] Z. Drmač, S. Gugercin, C. Beattie, Vector fitting for matrix-valued rational approximation, *SIAM J. Sci. Comput.* 37 (5) (2015) A2346–A2379. doi:10.1137/15M1010774.
- [13] Y. Nakatsukasa, O. Sète, L. N. Trefethen, The AAA algorithm for rational approximation, *SIAM J. Sci. Comput.* 40 (3) (2018) A1494–A1522.
- [14] B. Peherstorfer, K. Willcox, Data-driven operator inference for nonintrusive projection-based model reduction, *Comp. Meth. Appl. Mech. Eng.* 306 (2016) 196–215. doi:10.1016/j.cma.2016.03.025.
- [15] N. L. C. Chui, J. M. Maciejowski, Realization of stable models with subspace methods, *Automatica* 32 (11) (1996) 1587–1595.
- [16] B. Boots, G. J. Gordon, S. Siddiqi, A constraint generation approach to learning stable linear dynamical systems, *Adv. Neural Inform. Processing Sys.* 20 (2007).
- [17] S. L. Lacy, D. S. Bernstein, Subspace identification with guaranteed stability using constrained optimization, *IEEE Trans. Autom. Control* 48 (7) (2003) 1259–1263.
- [18] P. Benner, C. Himpe, T. Mitchell, On reduced input-output dynamic mode decomposition, *Adv. Comput. Math.* 44 (6) (2018) 1751–1768. doi:10.1007/s10444-018-9592-x.
- [19] N. Gillis, M. Karow, P. Sharma, Approximating the nearest stable discrete-time system, *Linear Algebra Appl.* 573 (2019) 37–53.
- [20] G. Mamakoukas, O. Xherija, T. Murphey, Memory-efficient learning of stable linear dynamical systems for prediction and control, *Adv. Neural Inform. Processing Sys.* 33 (2020) 13527–13538.
- [21] N. Sawant, B. Kramer, B. Peherstorfer, Physics-informed regularization and structure preservation for learning stable reduced models from data with operator inference, *arXiv preprint arXiv:2107.02597* (2021).
- [22] N. Gillis, P. Sharma, On computing the distance to stability for matrices using linear dissipative Hamiltonian systems, *Automatica* 85 (2017) 113–121.
- [23] R. T. Chen, Y. Rubanova, J. Bettencourt, D. K. Duvenaud, Neural ordinary differential equations, *Adv. Neural Inform. Processing Sys.* 31 (2018).
- [24] R. González-García, R. Rico-Martínez, I. G. Kevrekidis, Identification of distributed parameter systems: A neural net based approach, *Computers & Chemical Engineering* 22 (1998) S965–S968.
- [25] P. Goyal, P. Benner, Discovery of nonlinear dynamical systems using a Runge-Kutta inspired dictionary-based sparse regression approach, *Proc. Royal Society A: Mathematical, Physical and Engineering Sciences* 478 (2262) (2022) 20210883. doi:10.1098/rspa.2021.0883.
- [26] W. I. T. Uy, D. Hartmann, B. Peherstorfer, Operator inference with roll outs for learning reduced models from scarce and low-quality data, *arXiv preprint arXiv:2212.01418* (2022).
- [27] B. O. Koopman, Hamiltonian systems and transformation in Hilbert space, *Proc. Nat. Acad. Sci. U.S.A.* 17 (5) (1931) 315–318.
- [28] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, *arXiv preprint arXiv:1412.6980* (2014).