Learning Stable Evolutionary PDE Dynamics
A Scalable System Identification Approach

Liu, Diyou; Khosravi, Mohammad

DOI
10.1109/CCTA60707.2024.10666557

Publication date
2024

Document Version
Final published version

Published in
Proceedings of the IEEE Conference on Control Technology and Applications, CCTA 2024

2024 IEEE Conference on Control Technology and Applications (CCTA)
August 21-23, 2024. Newcastle upon Tyne, UK

2024 IEEE Conference on Control Technology and Applications (CCTA) | 979-8-3503-7094-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/CCTA60707.2024.10666557

Learning Stable Evolutionary PDE Dynamics: A Scalable System Identification Approach

Diyou Liu and Mohammad Khosravi

Abstract—**In this paper, we discuss the learning and discovery problem for the dynamical systems described through stable evolutionary Partial Differential Equations (PDEs). The main idea is to employ a suitable learning approach for creating a map from boundary conditions to the corresponding output. More precisely, in order to accurately uncover the evolutionary PDE dynamics, we propose a scheme that employs large-scale system identification to construct such a map using sufficiently informative measurements. Accordingly, we first develop a scalable implementation for the subspace identification method, enforcing stability on the identified system. To this end, numerical optimization techniques such as coordinate descent, randomized singular value decomposition, and large-scale semidefinite programming are employed. The performance and complexity of the resulting scheme are discussed and demonstrated through numerical experiments on generic identification examples. Following this, we validate the effectiveness of the proposed approach on an example of a stable evolutionary partial differential equation. The numerical results confirm the efficacy of the proposed learning scheme.**

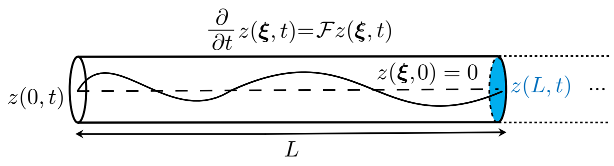
I. Introduction

Partial Differential Equations (PDEs) are ubiquitous in science and technology [1], serving as fundamental tools for understanding, analyzing, and predicting complicated real-world phenomena, such as fluid dynamics [2], thermodynamics [3], and electrodynamics [4]. Accordingly, their theory and techniques have been extensively studied, and various methods have been developed for the related *forward problems*, i.e., deriving the corresponding solution given the PDE dynamics and other required conditions [1]. While traditional methods focus on solving the forward problems analytically [5], due to the diverse classes of partial differential equations, establishing a general analytical approach encompassing all PDEs is challenging [6]. Additionally, the complicated nature of PDEs describing complex physical systems presents significant limitations in obtaining analytical solutions, prompting the use of numerical methods [7], including finite difference [7], finite element [8], and finite volume [9] methods. Nonetheless, these methods are prone to major issues such as numerical instability, slow computational efficiency, and sensitivity to grid dependence, requiring ongoing improvements.

Due to the complexity of deriving PDE dynamics physically, the *inverse problems* associated with PDEs, i.e., utilizing data-driven modeling techniques to recover either the PDE itself, partially or entirely, or its solution from synthetic or measurement data, have received significant attention in recent years [10–12]. To this end, various methodologies are proposed, e.g., using kernel-based learning [13, 14], Koopman operator theory [15, 16], and deep learning [11]. Some of the research focus on recovering PDE directly. For example, PDE-FIND, proposed in [10], employs a dictionary of possible derivative terms and represents the PDE parametrically in terms of dictionary elements, where the parameters are subsequently determined through a regression problem minimizing the L_2-norm error of fitting data to the estimated forward model result. Moreover, in [12], the problem is formulated through an adjoint method, where a constraint is introduced to enforce the estimation results as a solution to a parameterized PDE. However, these methods have several limitations. First, one needs knowledge of the underlying PDE and a suitable parameterization to obtain an accurate solution, which may not be available. Furthermore, when the initial condition, boundary condition, or external forcing change, the PDE solution needs to be recomputed from the obtained PDE dynamics, which can be computationally intensive. Instead of recovering PDE, some research papers focus on recovering the PDE solution. For instance, in [11], physics-informed neural networks (PINN) are proposed, which uses a fully connected neural network to train supervised learning tasks. The loss function is built based on physical laws (PDE dynamics, initial and boundary conditions). Furthermore, in [17], based on PINN, the author proposed to use forward neural network and residual neural network to improve the performance. However, in order to build a loss function, these methods require the knowledge of structure of PDE dynamics, which may not be available in some scenarios. Additionally, the neural network needs to be retrained if the initial or boundary conditions change.

This paper focuses on recovering and learning a stable evolutionary PDE. The goal is to learn a mapping between the boundary conditions and the PDE solution at the specified region. Thus, we can ensure that no additional training is required when the boundary conditions change. To this end, we treat PDE as a system to be identified, and subsequently, we employ a system identification scheme suitable for handling vast amount of data. To this end, we propose a scalable implementation of a subspace state space system identification method with enforced stability [18–20], which belongs to the general domain of identification techniques incorporating side-information [21–24]. Note that,

Delft Center for Systems and Control, Delft University of Technology, Delft, Netherlands. Email: {d.liu-9, mohammad.khosravi}@tudelft.nl



$\frac{\partial}{\partial t}z(\boldsymbol{\xi},t)=\mathcal{F}z(\boldsymbol{\xi},t)$
$z(\boldsymbol{\xi},0)=0$
$z(0,t)$
$z(L,t)$
$\cdots$
L

Figure 1. One dimensional heat conduction in rod with initial and boundary conditions.

given sufficient measurement data, we first estimate impulse response by solving a least squares problem. The resulting estimates can be used to establish the extended observability and extended controllability matrices using singular value decomposition (SVD). Subsequently, from the observability matrix, an optimization problem is formulated to find an estimate of system matrices, where a linear matrix inequality constraint is introduced to ensure stability of the estimated system. We utilize the proposed scheme to learn stable linear evolutionary PDEs and solve them numerically. Given a linear stable evolutionary PDE, the boundary conditions can be seen as the input and the PDE solution at the specified region as the output. To identify the system, we first sample the boundary conditions and the solution over the region of interest to collect data. Then, the proposed system identification method is used to learn the mapping as a system. Having found the estimated system matrix, one can easily obtain the output for any given input.

II. Discovery Problem for Stable Evolutionary PDE Dynamics

Consider a stable dynamical system described through an evolutionary PDE dynamics as

$$\frac{\partial}{\partial t}z(\boldsymbol{\xi},t) = \mathcal{F}\big(z(\boldsymbol{\xi},t)\big), \qquad \forall\, (\boldsymbol{\xi},t) \in \Omega \times [0,\infty), \tag{1}$$

where Ω is a simply connected closed subset of $\mathbb{R}^n$ with boundary $\partial\Omega$, and $\mathcal{F}$ is a linear differential operator characterizing the spatial evolution in (1). For the ease of discussion, assume the system is initially at rest, i.e., $z(\boldsymbol{\xi},0)=0$, for all $\boldsymbol{\xi}\in\Omega$. Let $\nu : \partial\Omega\times[0,\infty)\to\mathbb{R}$ be a function characterizing an *external actuation* applied to the boundary of system $\partial\Omega$, and we have no additional input on $\Omega\backslash\partial\Omega$. Moreover, let $\zeta : \Xi\times[0,\infty)\to\mathbb{R}$ be the respective *response* of the system on $\Xi\subseteq\Omega\backslash\partial\Omega$. In this paper, our motivating example is the one dimensional heat conduction over a thin rod shown in Figure 1.

When system dynamics (1) is completely known and ν is given, one may obtain ζ through numerical methods solving the above-mentioned PDE. However, introducing a new external actuation requires repeating the numerical procedure to determine the updated response, which can be computationally intensive. Furthermore, in various situations, the PDE dynamics (1) may not be known exactly. Following the above discussion and considering the dependency between the external actuation and response, we pose the problem of *Learning Stable Evolutionary PDE Dynamics* as below.

Problem (Learning Stable Evolutionary PDE Dynamics)**.** *Let $\nu : \partial\Omega\times[0,\infty)\to\mathbb{R}$ and $\zeta : \Xi\times[0,\infty)\to\mathbb{R}$ be a given pair of external actuation on the boundary and response with respect to PDE (1), which is possibly unknown. Find a suitable operator mapping the external actuations to the corresponding response.*

III. Discrete Dynamic Model for Stable Evolutionary PDE

In order to solve the problem introduced in Section II, we treat PDE as a dynamic system. Assume $\{\boldsymbol{\omega}_i | i=1,\ldots,n_{\mathrm{u}}\}$ and $\{\boldsymbol{\xi}_i | i=1,\ldots,n_{\mathrm{y}}\}$ are respectively sets of grid points in $\partial\Omega$ and Ξ. Accordingly, we define discrete-time signals $u : [0,\infty)\to\mathbb{R}^n_{\mathrm{u}}$ and $y : [0,\infty)\to\mathbb{R}^n_{\mathrm{y}}$ respectively as $\mathrm{u}_k = [z(\boldsymbol{\omega}_i,k\Delta T)]_{i=1}^{n_{\mathrm{u}}}$ and $\mathrm{y}_k = [z(\boldsymbol{\xi}_i,k\Delta T)]_{i=1}^{n_{\mathrm{y}}}$, for $k\in\mathbb{Z}_+$, where $\Delta T>0$ is a small sampling time. Accordingly, the introduced problem boils down to find a stable linear dynamical system $\mathcal{S}$, which accurately estimates the dynamic from input to output, and described as

$$\begin{aligned} \mathrm{x}_{k+1} &= \mathrm{A}\mathrm{x}_k + \mathrm{B}\mathrm{u}_k, \\ \mathrm{y}_k &= \mathrm{C}\mathrm{x}_k + \mathrm{D}\mathrm{u}_k + \mathrm{w}_k, \end{aligned} \qquad \forall\, k\in\mathbb{Z}, \tag{2}$$

where vectors $\mathrm{x}_k\in\mathbb{R}^{n_{\mathrm{x}}}$, $\mathrm{u}_k\in\mathbb{R}^{n_{\mathrm{u}}}$, and $\mathrm{y}_k\in\mathbb{R}^{n_{\mathrm{y}}}$, respectively correspond to the state, input and output of the system at time $k\in\mathbb{Z}$, matrices $\mathrm{A}\in\mathbb{R}^{n_{\mathrm{x}}\times n_{\mathrm{x}}}$, $\mathrm{B}\in\mathbb{R}^{n_{\mathrm{x}}\times n_{\mathrm{u}}}$, $\mathrm{C}\in\mathbb{R}^{n_{\mathrm{y}}\times n_{\mathrm{x}}}$, and $\mathrm{D}\in\mathbb{R}^{n_{\mathrm{y}}\times n_{\mathrm{u}}}$ characterize the system, and, $(\mathrm{w}_k)_{k\in\mathbb{Z}}$ is a zero-mean white noise process affecting the output measurements of the system.

To find unknown system $\mathcal{S}$, sufficiently informative measurement data is required. Accordingly, we assume that the system is actuated with a persistently exciting input signal of suitable length, and the output of the system is then measured at time instants $t=0,1,\ldots,n_{\mathcal{D}}-1$. Thus, we are provided with the following set of data

$$\mathcal{D} := \big\{(\mathrm{u}_k,\mathrm{y}_k)\,\big|\, k=0,1,\ldots,n_{\mathcal{D}}-1\big\}, \tag{3}$$

which contains $n_{\mathcal{D}}\in\mathbb{Z}_+$ input-output measurement pairs. For accurate estimation, a dense mesh is required for discretization. Thus, $\mathcal{S}$ is a large-scale system. More precisely, the dimensions of the state, input, or output vectors can be considerably large, i.e., $n_{\mathrm{x}}, n_{\mathrm{u}}, n_{\mathrm{y}} \gg 1$. Therefore, the sufficient informativity of the measurements data leads to employing a substantially small ΔT, which leads to considerably large $n_{\mathcal{D}}$ and dealing with a massive set of measurement data. Furthermore, since $\mathcal{S}$ is a stable system, the stability constraint needs to be imposed on to be identified model.

IV. Scalable Stable Subspace Identification

In this section, we introduce a scalable algorithm for subspace state space system identification (N4SID) to identify (2). The algorithm is summarized in Algorithm 1.

For system (2), the output, at time instant $k\in\mathbb{Z}$, is

$$\mathrm{y}_k = \sum_{s=0}^{n_{\mathrm{G}}-1} \mathrm{G}_s\mathrm{u}_{k-s} + \mathrm{C}\mathrm{A}^{n_{\mathrm{G}}}\mathrm{x}_{k-n_{\mathrm{G}}+1} + \mathrm{w}_k, \tag{4}$$

where $\mathrm{G}_0 = \mathrm{D}$ and $\mathrm{G}_s = \mathrm{C}\mathrm{A}^{s-1}\mathrm{B}, s \geq 1$. For stable systems and large enough n_{G}, we can assume $\mathrm{C}\mathrm{A}^{n_{\mathrm{G}}}\mathrm{x}_{k-n_{\mathrm{G}}+1} \approx 0$. This is valid assumption also when the system is initially at rest. Define $\mathrm{Y} = [\mathrm{y}_0^\intercal, \mathrm{y}_1^\intercal, ..., \mathrm{y}_{n_{\mathcal{D}}-1}^\intercal]^\intercal$ and $\mathrm{G} = [\mathrm{G}_0, \mathrm{G}_1, ..., \mathrm{G}_{n_{\mathrm{G}}-1}]$. Thus, equation (4) can be re-written as

$$\mathrm{Y} = \mathrm{T}_u\mathrm{G}^\intercal + \mathrm{W}, \tag{5}$$

where $\mathrm{W} = [\mathrm{w}_0^\intercal, \mathrm{w}_1^\intercal, ..., \mathrm{w}_{n_{\mathcal{D}}-1}^\intercal]^\intercal$ and T_u is a lower block triangular Toeplitz matrix defined as

$$\mathrm{T}_u = \begin{bmatrix} \mathrm{u}_0^\intercal & 0 & \dots & 0 \\ \mathrm{u}_1^\intercal & \mathrm{u}_0^\intercal & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ \mathrm{u}_{n_{\mathcal{D}}-1}^\intercal & \mathrm{u}_{n_{\mathcal{D}}-2}^\intercal & \dots & \mathrm{u}_{n_{\mathcal{D}}-n_{\mathrm{G}}}^\intercal \end{bmatrix}.$$

With sufficiently informative measurements, impulse response $\hat{\mathrm{G}}$ can be estimated by solving a linear least squares problem as

$$\hat{\mathrm{G}} = \mathrm{argmin}_{\mathrm{G}\in\mathbb{R}^{n_{\mathrm{y}}\times n_{\mathrm{G}}n_{\mathrm{u}}}} \|\mathrm{Y} - \mathrm{T}_u\mathrm{G}^\intercal\|^2. \tag{6}$$

To reduce the computational complexity, we split matrix Y and $\mathrm{G}^\intercal$ into n_{batch} blocks, i.e. $\mathrm{Y} = [\mathrm{Y}_1, \mathrm{Y}_2, ..., \mathrm{Y}_{n_{\text{batch}}}]$, $\mathrm{G}^\intercal = [\mathrm{H}_1, \mathrm{H}_2, ..., \mathrm{H}_{n_{\text{batch}}}]$, where $\mathrm{Y}_i \in \mathbb{R}^{n_{\mathcal{D}}\times b_i}$, $\mathrm{H}_i \in \mathbb{R}^{n_{\mathrm{G}}n_{\mathrm{u}}\times b_i}$ and $\sum_{i=1}^{n_{\text{batch}}} b_i = n_{\mathrm{y}}$. With the split matrices, (6) can be solved by n_{batch} linear least squares as

$$\min_{\mathrm{H}_i\in\mathbb{R}^{n_{\mathrm{G}}n_{\mathrm{u}}\times b_i}} \|\mathrm{Y}_i - \mathrm{T}_u\mathrm{H}_i\|^2, \quad \forall i = 1, \dots, n_{\text{batch}}. \tag{7}$$

Note that Y_i gathers the output of the all channels in the i^{th} block across the entire dataset, while H_i represents the impulse response of all input channels to all output channels in the i^{th} block during the initial $n_{\mathcal{D}}$ time steps. Given that $\mathcal{D}$ is a sufficient data set and $n_{\mathcal{D}}$ is large, solving (7) may still lead to memory issues. Accordingly, we implement block coordinate descent method [25].

To realize the extended observability matrix, the Ho-Kalman method and eigensystem realization algorithm (ERA) are used [18]. Specifically, let $n_{\mathrm{H}} = \lfloor n_{\mathrm{G}}/2 \rfloor$ and define block Hankel matrix $\mathrm{H}_{\mathrm{G}} \in \mathbb{R}^{n_{\mathrm{H}}n_{\mathrm{y}}\times n_{\mathrm{H}}n_{\mathrm{u}}}$ as

$$\mathrm{H}_{\mathrm{G}} = \begin{bmatrix} \mathrm{G}_1 & \mathrm{G}_2 & \dots & \mathrm{G}_{n_{\mathrm{H}}} \\ \mathrm{G}_2 & \mathrm{G}_3 & \dots & \mathrm{G}_{n_{\mathrm{H}}+1} \\ \vdots & \vdots & \vdots & \vdots \\ \mathrm{G}_{n_{\mathrm{H}}} & \mathrm{G}_{n_{\mathrm{H}}+1} & \dots & \mathrm{G}_{2n_{\mathrm{H}}-1} \end{bmatrix}. \tag{8}$$

By substituting Markov parameters $\mathrm{G}_s = \mathrm{C}\mathrm{A}^{s-1}\mathrm{B}, s \geq 1$, we can re-write the block Hankel matrix H_{G} as

$$\mathrm{H}_{\mathrm{G}} = \begin{bmatrix} \mathrm{C} \\ \mathrm{CA} \\ \vdots \\ \mathrm{CA}^{n_{\mathrm{H}}-1} \end{bmatrix} \begin{bmatrix} \mathrm{B} & \mathrm{AB} & \dots & \mathrm{A}^{n_{\mathrm{H}}-1}\mathrm{B} \end{bmatrix}. \tag{9}$$

In the above equation, the last line is the product of extended observability matrix and extended controllability matrix. Assume n_{G} is chosen large enough such that we have $n_{\mathrm{x}} \leq n_{\mathrm{H}}n_{\mathrm{u}}, n_{\mathrm{H}}n_{\mathrm{y}}$. Since the system is stable, $\mathrm{rank}(\mathrm{H}_{\mathrm{G}}) = n_{\mathrm{x}}$. we can factorize H_{G} using SVD, i.e. $\mathrm{H}_{\mathrm{G}} = \mathrm{U}\mathrm{S}\mathrm{V}^\intercal$, where $\mathrm{rank}(\mathrm{S}) = n_{\mathrm{x}}$.

The block Hankel matrix H_{G} is unknown, but an estimation $\mathrm{H}_{\hat{\mathrm{G}}}$ can be constructed using the solution $\hat{\mathrm{G}}$ of (6). This estimation matrix can also be factorized and we keep the largest n_{x} singular values with its corresponding left and right singular vectors, i.e. $\mathrm{H}_{\hat{\mathrm{G}}} = \hat{\mathrm{U}}\hat{\mathrm{S}}\hat{\mathrm{V}}^\intercal$, where $\hat{\mathrm{U}} \in \mathbb{R}^{n_{\mathrm{H}}n_{\mathrm{y}}\times n_{\mathrm{x}}}, \hat{\mathrm{S}} \in \mathbb{R}^{n_{\mathrm{x}}\times n_{\mathrm{x}}}$ and $\hat{\mathrm{V}} \in \mathbb{R}^{n_{\mathrm{H}}n_{\mathrm{u}}\times n_{\mathrm{x}}}$. We can write $\mathrm{H}_{\hat{\mathrm{G}}}$ as $\hat{\mathrm{U}}\hat{\mathrm{S}}^{\frac{1}{2}}\hat{\mathrm{S}}^{\frac{1}{2}}\hat{\mathrm{V}}^\intercal$, where $\hat{\mathrm{O}}_{n_{\mathrm{H}}} := \hat{\mathrm{U}}\hat{\mathrm{S}}^{\frac{1}{2}}$ and $\hat{\mathrm{C}}_{n_{\mathrm{H}}} := \hat{\mathrm{S}}^{\frac{1}{2}}\hat{\mathrm{V}}^\intercal$ are the estimation of extended observability matrix and extended controllability matrix, respectively, up to a linear transformation.

Remark 1. Note that for large scale systems, storing $\mathrm{H}_{\hat{\mathrm{G}}}$ is resource intensive. We use Python package DASK [26] to enable our system utilizing disk space as an extension of memory and to perform parallel computation. Additionally, Randomized SVD [27] can be used when $\mathrm{H}_{\hat{\mathrm{G}}}$ is of large size.

From the estimation of extended observability matrix $\hat{\mathrm{O}}_{n_{\mathrm{H}}}$ and extended controllability matrix $\hat{\mathrm{C}}_{n_{\mathrm{H}}}$, one can easily extract the realization of matrices $\hat{\mathrm{B}}$ and $\hat{\mathrm{C}}$ as the first n_{u} columns of $\hat{\mathrm{C}}_{n_{\mathrm{H}}}$ and the first n_{y} rows of $\hat{\mathrm{O}}_{n_{\mathrm{H}}}$, respectively. Thus, we only need to find stable matrix $\hat{\mathrm{A}}$ as an estimation of matrix A. Define $\underline{\Gamma}$ and $\overline{\Gamma}$ respectively as the first and the last $(n_{\mathrm{H}}-1)n_{\mathrm{y}}$ rows of $\hat{\mathrm{O}}_{n_{\mathrm{H}}}$. Therefore, $\underline{\Gamma}$ and $\overline{\Gamma}$ are estimations of $\mathrm{O}_{n_{\mathrm{H}}-1}$ and $\mathrm{O}_{n_{\mathrm{H}}-1}\mathrm{A}$, respectively. Accordingly, given $\hat{\mathrm{O}}_{n_{\mathrm{H}}}$, one can estimate A by solving the following program:

$$\begin{aligned} (\hat{\mathrm{A}}, \hat{\mathrm{P}}) \quad := \quad & \underset{\mathrm{A},\mathrm{P}\in\mathbb{R}^{n_{\mathrm{x}}\times n_{\mathrm{x}}}}{\mathrm{argmin}} && \left\|\mathrm{W}_2(\overline{\Gamma} - \underline{\Gamma}\mathrm{A})\mathrm{W}_1\right\|_{\mathrm{F}}^2, \\ & \quad\text{s.t.} && \mathrm{A}\mathrm{P}\mathrm{A}^\intercal - \mathrm{P} \preceq -\mathbb{I}_{n_{\mathrm{x}}}, \\ & && \mathrm{P} \succeq \mathbb{I}_{n_{\mathrm{x}}}, \end{aligned} \tag{10}$$

where W_1 and W_2 are suitably chosen weight matrices. The constraints ensure stability of matrix $\hat{\mathrm{A}}$.

Note that (10) is a non-convex problem. To solve this problem efficiently and accurately, we convert it to a convex problem [19]. From Schur complements, one can see that if (A, P) satisfies the constraints in (10), then we have

$$\begin{bmatrix} \mathrm{P} - \mathbb{I}_{n_{\mathrm{x}}} & \mathrm{AP} \\ \mathrm{PA}^\intercal & \mathrm{P} \end{bmatrix} \succeq \mathbb{0}_{2n_{\mathrm{x}}}. \tag{11}$$

Define matrix $\mathrm{W}_3 = \mathrm{P}^{-1}\mathrm{W}_1$ and $\mathrm{Q} = \mathrm{AP}$. One can re-write the estimation problem (10) as

$$\begin{aligned} (\hat{\mathrm{P}}, \hat{\mathrm{Q}}) \quad := \quad & \underset{\mathrm{P},\mathrm{Q}\in\mathbb{R}^{n_{\mathrm{x}}\times n_{\mathrm{x}}}}{\mathrm{argmin}} && \left\|\mathrm{W}_2(\overline{\Gamma}\mathrm{P} - \underline{\Gamma}\mathrm{Q})\mathrm{W}_3\right\|_{\mathrm{F}}^2, \\ & \quad\text{s.t.} && \begin{bmatrix} \mathrm{P} - \mathbb{I}_{n_{\mathrm{x}}} & \mathrm{Q} \\ \mathrm{Q}^\intercal & \mathrm{P} \end{bmatrix} \succeq \mathbb{0}_{2n_{\mathrm{x}}}, \end{aligned} \tag{12}$$

and estimate A as $\hat{\mathrm{A}} = \mathrm{Q}\mathrm{P}^{-1}$. To further simplify the problem, let W_2 and W_3 be identity matrices. Define matrices $\mathrm{R} = \begin{bmatrix} -\underline{\Gamma} & \overline{\Gamma} \end{bmatrix}$, $\mathrm{S} = \begin{bmatrix} \mathbb{0}_{n_{\mathrm{x}}} \\ \mathbb{I}_{n_{\mathrm{x}}} \end{bmatrix}$, $\mathrm{T} = \begin{bmatrix} \mathbb{I}_{n_{\mathrm{x}}} \\ \mathbb{0}_{n_{\mathrm{x}}} \end{bmatrix}$, and $\mathrm{X} = \begin{bmatrix} \mathrm{P} - \mathbb{I}_{n_{\mathrm{x}}} & \mathrm{Q} \\ \mathrm{Q}^\intercal & \mathrm{P} \end{bmatrix}$. Accordingly, we can re-write (12) as

$$\begin{aligned} \hat{\mathrm{X}} \quad := \quad & \underset{\mathrm{X}\in\mathbb{R}^{2n_{\mathrm{x}}\times 2n_{\mathrm{x}}}}{\mathrm{argmin}} && \left\|\mathrm{R}\mathrm{X}\mathrm{S}\right\|_{\mathrm{F}}^2, \\ & \quad\text{s.t.} && \mathrm{X} \succeq \mathbb{0}_{2n_{\mathrm{x}}}, \\ & && \mathrm{S}^\intercal\mathrm{X}\mathrm{S} - \mathrm{T}^\intercal\mathrm{X}\mathrm{T} = \mathbb{I}_{n_{\mathrm{x}}}. \end{aligned} \tag{13}$$

Remark 2. The introduced optimization problem is convex and can be solved using Semidefinite Programming (SDP) techniques. However, it would become challenging for the large scale systems, as n_{H}, n_{x} and n_{y} can have significantly large values. For high values of n_{x}, the extensive size of X poses memory issues. Furthermore, with large values for n_{y} and n_{H}, the cost function is a summation of numerous terms, resulting in significant complexity and high computation effort.

Regarding the potential memory issues, we use the interior-point method [28] to convert (13) to an unconstrained optimization problem and solve it through Limited-memory BFGS algorithm [29]. Define cost function

$$\begin{aligned} J_\rho(\mathrm{X}) \;:=\; & \frac{1}{2}(\|\mathrm{RXS}\|_{\mathrm{F}}^2 + \|\mathrm{S^\intercal X R^\intercal}\|_{\mathrm{F}}^2) \\ & - \frac{1}{\rho} b(\mathrm{X}) + \rho \|\mathrm{S^\intercal X S} - \mathrm{T^\intercal X T} - \mathbb{I}_{n_{\mathrm{x}}}\|_{\mathrm{F}}^2, \end{aligned} \tag{14}$$

where $\rho > 0$ is a weight parameter, and $b : \mathbb{R}^{2n_{\mathrm{x}} \times 2n_{\mathrm{x}}} \to \mathbb{R} \cup \{\infty\}$ is the logarithmic barrier function defined as

$$b(\mathrm{X}) = \begin{cases} \log(\det(\mathrm{X})), & \text{if } \mathrm{X} \succeq \mathbb{0}_{2n_{\mathrm{x}}}, \\ \infty, & \text{else}. \end{cases} \tag{15}$$

As ρ goes to infinity, the minmum of $J_\rho(\mathrm{X})$ converges to a solution of (13). Moreover, $\|\mathrm{RXS}\|_{\mathrm{F}}^2$ is re-written as $1/2(\|\mathrm{RXS}\|_{\mathrm{F}}^2 + \|\mathrm{S^\intercal X R^\intercal}\|_{\mathrm{F}}^2)$, so that the $\nabla_{\mathrm{X}} J_\rho(\mathrm{X})$ becomes symmetric when X is symmetric and positive definite

$$\begin{aligned} \nabla_{\mathrm{X}} J_\rho(\mathrm{X}) = & \mathrm{R^\intercal R X S S^\intercal} + \mathrm{S S^\intercal X R^\intercal R} - \frac{1}{\rho}\mathrm{X}^{-1} + 2\rho[\mathrm{S}(\mathrm{S^\intercal X S} \\ & - \mathrm{T^\intercal X T})\mathrm{S^\intercal} + \mathrm{T}(\mathrm{T^\intercal X T} - \mathrm{S^\intercal X S})\mathrm{T^\intercal} - \mathrm{S S^\intercal} + \mathrm{T T^\intercal}]. \end{aligned}$$

For each ρ, gradient descent can be used to find the optimal $\hat{\mathrm{X}}$, which can be described by the sequence

$$\mathrm{vec}(\mathrm{X}_{k+1}) = \mathrm{vec}(\mathrm{X}_k) - \alpha_k \mathrm{H}_k \,\mathrm{vec}(\nabla_{\mathrm{X}} J_\rho(\mathrm{X}_k)), \tag{16}$$

where α_k is the stepsize and H_k is the inverse Hessian matrix. To avoid computing inverse matrix of large Hessian matrix, which is computationally complex, the inverse Hessian matrix H_k is recursively updated using L-BFGS method [29, Algorithm 2.1].

Algorithm 1 Stable N4SID for Large Scale Systems Identification

```
    Input: $\mathcal{D}$.
    Output: $\hat{\mathrm{A}}$, $\hat{\mathrm{B}}$ and $\hat{\mathrm{C}}$.
    BCD Least Squares:
    Build $\mathrm{T}_u$ from $\mathcal{D}$
 1: for $k \leftarrow 1$ to $n_{\text{batch}}$ do
 2:     $\mathrm{M} \leftarrow \mathrm{T}_u, b \leftarrow \mathrm{Y}_k, z \leftarrow \mathbb{0}_{n_{\mathrm{G}} n_{\mathrm{u}}}$
 3:     Pre Computation:
            $p_i := (\mathrm{M}_i^\intercal \mathrm{M}_i)^{-1} \mathrm{M}_i^\intercal b$
            $Q_{i,j} := (\mathrm{M}_i^\intercal \mathrm{M}_i)^{-1} \mathrm{M}_i^\intercal \mathrm{M}_j, \forall i, j \in \mathbb{R}^n, i \neq j$
 4:     while 1 do
 5:         $z_i \leftarrow p_i - \sum_{j=1, j\neq i}^{n} Q_{i,j} z_j,\; i \leftarrow i+1$
 6:         if z converged then
 7:             break
 8:     $[\hat{\mathrm{G}}^\intercal]_i \leftarrow z$
 9:     Build block Hankel matrix $\mathrm{H}_{\hat{\mathrm{G}}}$
    Randomized SVD:
10: choose $q, l$
11: Generate Gaussian matrix $\Omega$
12: $\mathrm{Y} \leftarrow (\mathrm{H}_{\hat{\mathrm{G}}} \mathrm{H}_{\hat{\mathrm{G}}}^*)^q \mathrm{H}_{\hat{\mathrm{G}}} \Omega$
13: QR factorization $\mathrm{Y} = \mathrm{Q_Y R_Y}$
14: $\mathrm{B} \leftarrow \mathrm{Q_Y^*} \mathrm{H}_{\hat{\mathrm{G}}}$
15: SVD decomposition $\mathrm{B} = \tilde{\mathrm{U}} \hat{\mathrm{S}} \hat{\mathrm{V}}^*$
16: Compute $\hat{\mathrm{U}} = \mathrm{Q_Y} \tilde{\mathrm{U}}$
17: $\hat{\mathrm{O}}_{n_{\mathrm{H}}} \leftarrow \hat{\mathrm{U}} \hat{\mathrm{S}}^{\frac{1}{2}}$ and $\hat{\mathrm{C}}_{n_{\mathrm{H}}} \leftarrow \hat{\mathrm{S}}^{\frac{1}{2}} \hat{\mathrm{V}}^*$
    State Space Realization:
18: Build $\underline{\Gamma}$ and $\overline{\Gamma}$ from $\hat{\mathrm{O}}_{n_{\mathrm{H}}}$
19: $\mathrm{R} \leftarrow \begin{bmatrix} -\underline{\Gamma} & \overline{\Gamma} \end{bmatrix}$, $\mathrm{S} \leftarrow \begin{bmatrix} \mathbb{0}_{n_{\mathrm{x}}} \\ \mathbb{I}_{n_{\mathrm{x}}} \end{bmatrix}$, $\mathrm{T} \leftarrow \begin{bmatrix} \mathbb{I}_{n_{\mathrm{x}}} \\ \mathbb{0}_{n_{\mathrm{x}}} \end{bmatrix}$
20: Define $\mathrm{X} = \begin{bmatrix} \mathrm{P} - \mathbb{I}_{n_{\mathrm{x}}} & \mathrm{Q} \\ \mathrm{Q}^\intercal & \mathrm{P} \end{bmatrix}$
21: for $k \leftarrow 1$ to $K$ do
22:     Define cost function (14)
23:     $\rho \leftarrow \mu\rho, \mu > 1$
24:     while 1 do
25:         Update $\mathrm{H}_k$ through [29, Algorithm 2.1]
26:         $\mathrm{vec}(\mathrm{X}) \leftarrow \mathrm{vec}(\mathrm{X}) - \alpha_k \mathrm{H}_k \,\mathrm{vec}(\nabla_{\mathrm{X}} J_\rho(\mathrm{X}))$
27:         if X converged then
28:             break
29: $\hat{\mathrm{A}} \leftarrow \mathrm{QP}^{-1}$, $\hat{\mathrm{B}}$ as the first $n_{\mathrm{u}}$ columns of $\hat{\mathrm{C}}_{n_{\mathrm{H}}}$ and $\hat{\mathrm{C}}$ as
    the first $n_{\mathrm{y}}$ rows of $\hat{\mathrm{O}}_{n_{\mathrm{H}}}$.
```

V. Procedure and Memory Complexity

A. Subspace Identification

In this section, the performance of the proposed scheme is demonstrated. We consider a system with 30 states, 2 outputs and 3 inputs. The systems are randomly generated with poles having magnitudes less than 0.7786. We generate random inputs for the system with a maximal infinity norm of 2. Three datasets with different sizes are generated. We compare the performance of three different methods, namely MOESP [30], N4SID, and N4SID with stability. The evaluation is based on the average relative impulse response error over $M = 100$ Monte-Carlo trials. All simulations are tested on DelftBlue with a maximum memory limit of 100GB.

Note that systems identified by N4SID and MOESP are not guaranteed to be stable, which might lead to an infinite $\mathcal{H}_2$ norm. In order to compare the performance, we use the impulse response error for the first K time steps, which is defined as

$$\text{error} \;=\; \frac{1}{M} \sum_{i=1}^{M} \frac{\sum_{j=1}^{K} \|\hat{y}_j^i - y_j^*\|_F^2}{\sum_{j=1}^{K} \|y_j^*\|_F^2}, \tag{17}$$

where $\hat{y}_j^i$ is the identified impulse response at time step j for the ith Monte-Carlo trial and y_j^* is the true impulse response at time step j. Table I compares the performance of the mentioned methods for different SNR levels and amounts of data. The table shows that the fitting error for models generated by the N4SID approach is larger than that of both stable N4SID and MOESP for the same SNR and data size,

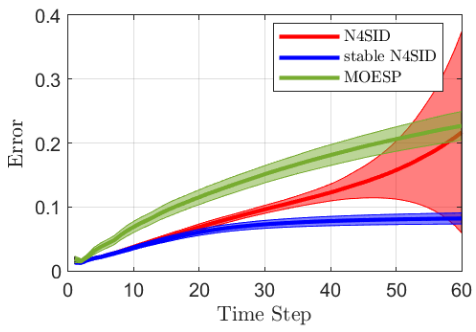


Figure 2. The impulse response error of three different methods with respect to time steps.

which is potentially due to the frequent instances of unstable systems identified by N4SID. In those cases, as time index increases, the impulse response tends to infinity, resulting in substantial impulse response error (17) for large values of K. Furthermore, due to potential instability, increasing data size and SNR do not lead to significant improvements in the performance of N4SID. Conversely, for stable N4SID and MOESP, the performance improves as data size and SNR increase. MOESP also identifies instances of unstable systems, albeit less frequently than N4SID. Consequently, the corresponding mean errors fall between those of the other two methods across all trials. Finally, MOESP reaches the memory usage limit for datasets with the size of one million. Therefore, it is computationally more complex than N4SID, which is one of the reasons we do not select it for large-scale systems identification. Figure 2 illustrates the impulse response error over time steps. Let μ and σ denote the mean error and standard deviation across all realizations, respectively. The solid line shows the mean error over time steps, while the shaded region covers region $(\mu-\sigma, \mu+\sigma)$. Notably, N4SID exhibits the largest standard deviation due to unstable identification, leading to a diverging mean error. Across all realizations, the maximal eigenvalue of A obtained by N4SID has a mean of 1.03 and a standard deviation of 0.03. Conversely, MOESP and stable N4SID exhibit relatively lower standard deviations. Finally, the results also indicate that stable N4SID outperforms the other two methods in terms of both mean error and standard deviation.

B. *Memory Complexity*

In this section, we demonstrate the memory complexity of the proposed scheme. In the experiment, we examine the memory usage for various data sizes across five different scenarios. These five scenarios differ in terms of state, input and output sizes. For the sake of a fair comparison, all scenarios have the same maximal eigenvalue 0.8 for the system matrix A. Thus, the truncation parameter n_{G} is set to 45 so that $\|\mathrm{A}^{n_{\mathrm{G}}}\|$ is close to zero and can be ignored.

Figure 3 illustrates the memory complexity with respect to the size of dataset. The figure shows that memory usage exhibits an approximate linear relationship with the amount of data, particularly for large datasets. However, in complex systems with numerous state variables and relatively small datasets, memory usage tends to be more influenced by the number of state variables. Nevertheless, as the dataset size increases, memory usage converges to a more linear relationship with data size.

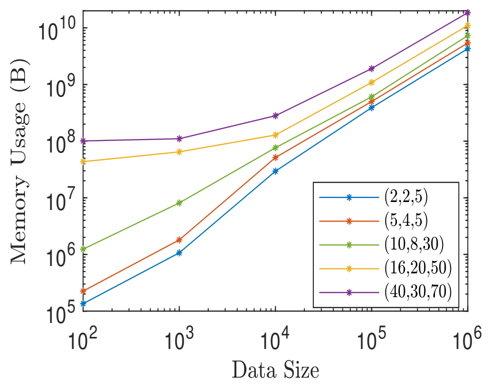


Figure 3. The computational complexity of stable N4SID for different data sizes under five scenarios with different system parameters $(n_{\mathrm{u}}, n_{\mathrm{y}}, n_{\mathrm{x}})$.

VI. Discovery of PDEs

In this section, we show the proposed scheme can be used to discover PDEs. Accordingly, we consider an augmented heat PDE as

$$\frac{\partial}{\partial t}z(\xi,t) = \alpha(\xi)\frac{\partial^2}{\partial \xi^2}z(\xi,t) + \beta(\xi)z(\xi,t), \tag{18}$$

where $\alpha, \beta : \mathbb{R}_+ \to \mathbb{R}$ are nonlinear functions. Note that when $\beta(\xi) = 0$, (18) is the heat equation. Let the PDE be initially at rest and the boundary condition be as $z(0,t) = u(t)$, for $t \in \mathbb{R}_+$. Define output as $y(t) = z(L,t)$, for $t \in \mathbb{R}_+$, where L is a positive real scalar. We employ the proposed scheme to learn a mapping between input u and output y. More precisely, to identify the system described above, we initially consider a uniform grid on the $[0, L]$ with $n_\xi + 1$ points $\{\xi_0, ..., \xi_{n_\xi}\}$, where $\xi_0 = 0$ and $\xi_{n_\xi} = L$. Define $z^i(t) = z(\xi_i, t)$, for $i = 0, \dots, n_\xi$. Then, we have $z^0(t) = u(t)$, $z^{n_\xi}(t) = y(t)$, and the approximate dynamics

$$\dot{z}^i(t) \approx \alpha(\xi_i)\frac{z^{i+1}(t) - 2z^i(t) + z^{i-1}(t)}{(\Delta\xi)^2} + \beta(\xi_i)z^i(t), \tag{19}$$

for $i = 1, \dots, n_\xi - 1$. The resulting dynamics can be regarded as a linear time-invariant (LTI) system. Therefore, we use stable N4SID to identify this LTI system.

For this numerical experiment, we set $\alpha(\xi) = 0.5\xi^2 - L\xi + 1$ and $\beta(\xi) = -0.2\xi^2$. Additionally, we set $L = 1$ and discretize it into 100 points. The discretized system is stable with all poles having magnitudes less than 0.9. Given input $u(t) = \sin(t) + 1$, we solve the system using an ODE solver. We consider $\Delta t = 0.01s$ and collect the first $n_{\mathcal{D}} = 1000$ input and output data points as dataset $\mathcal{D}$. Figure 4 shows the performance comparison of stable N4SID with ODE solver or finite difference method under four different external inputs. In the first three scenarios, the input u are nonlinear functions and (19) can be analytically solved. Therefore, the performance is compared to the analytical solution. However, for the last scenario, random input is applied to the system so that we compare stable N4SID with the finite difference

Table I
Error of three methods with respect to SNR and data size.

data size \ method	N4SID			stable N4SID			MOESP		
	5dB	10dB	20dB	5dB	10dB	20dB	5dB	10dB	20dB
1e4	0.83	0.67	0.60	0.14	0.08	0.05	0.48	0.26	0.13
1e5	2.85	0.92	1.53	0.05	0.03	0.02	0.22	0.11	0.05
1e6	0.73	3.42	0.81	0.02	0.01	0.005	Out of Memory		

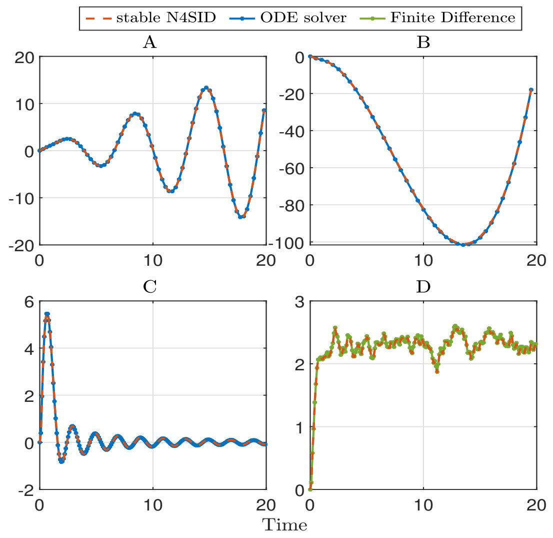


Figure 4. Output of the identified system for the augmented heat equation compared to ODE solver or finite difference method under four distinct input sequences. A: $u(t) = t\sin(t)$, B: $u(t) = 0.1t^3 - 2t^2 + t - 2$, C: $u(t) = 10\,\mathrm{sinc}(t)$, D: random input.

method. Across all scenarios, the output obtained from the identified system closely aligns with the ground truth and the relative error is approximately 1%. The results show that the systems identified by stable N4SID exhibit behaviors that closely match those of the real systems.

VII. Conclusion

In this work, we have proposed a scalable system identification approach applicable to large-scale systems. Through numerical experiments, we have compared the proposed approach with existing methods and verified its memory efficiency. Furthermore, we have demonstrated that the proposed scheme can be utilized for PDE learning and inverse PDE problems. Future work will focus on discovering more complex PDEs and analyzing noisy systems.

References

[1] Lawrence C Evans. *Partial Differential Equations*, volume 19. American Mathematical Society, 2022.

[2] Michael Eckert. *The Dawn of Fluid Dynamics: A Discipline Between Science and Technology*. John Wiley & Sons, 2007.

[3] Elias Cueto and Francisco Chinesta. Thermodynamics of learning physical phenomena. *Archives of Computational Methods in Engineering*, pages 1–14, 2023.

[4] Herbert W Meyer. A History of Electricity and Magnetism. 1971.

[5] Jirair Kevorkian. *Partial Differential Equations: Analytical Solution Techniques*, volume 89871. Springer, 1990.

[6] Sergiu Klainerman. PDE as a unified subject. In *Visions in Mathematics: GAFA 2000 Special Volume, Part I*, pages 279–315. Springer, 2010.

[7] Christian Grossmann. *Numerical Treatment of Partial Differential Equations*. Springer, 2007.

[8] Gouri Dhatt, Emmanuel Lefrançois, and Gilbert Touzot. *Finite Element Method*. John Wiley & Sons, 2012.

[9] Robert Eymard, Thierry Gallouët, and Raphaèle Herbin. Finite Volume Methods. *Handbook of numerical analysis*, 7:713–1018, 2000.

[10] Samuel H Rudy, Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Data-driven discovery of Partial Differential Equations. *Science advances*, 3(4):e1602614, 2017.

[11] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear Partial Differential Equations. *Journal of Computational physics*, 378:686–707, 2019.

[12] Mohsen Sadr, Tony Tohme, and Kamal Youcef-Toumi. Data-driven discovery of PDEs via the adjoint method. *arXiv preprint arXiv:2401.17177*, 2024.

[13] George Stepaniants. Learning partial differential equations in reproducing kernel Hilbert spaces. *Journal of Machine Learning Research*, 24(86):1–72, 2023.

[14] Michael Griebel and Christian Rieger. Reproducing kernel Hilbert spaces for parametric partial differential equations. *SIAM/ASA Journal on Uncertainty Quantification*, 5(1):111–137, 2017.

[15] Mohammad Khosravi. Representer theorem for learning Koopman operators. *IEEE Transactions on Automatic Control*, 2023.

[16] Mohammad Khosravi. Learning finite-dimensional representations for Koopman operators. In *Learning for Dynamics and Control*. PMLR, 2021.

[17] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. DeepXDE: A deep learning library for solving differential equations. *SIAM review*, 63(1):208–228, 2021.

[18] BL Ho and Rudolf E Kálmán. Effective construction of linear state-variable models from input/output functions: Die konstruktion von linearen modeilen in der darstellung durch zustandsvariable aus den beziehungen für ein-und ausgangsgrößen. *at-Automatisierungstechnik*, 14(1-12):545–548, 1966.

[19] Seth L Lacy and Dennis S Bernstein. Subspace identification with guaranteed stability using constrained optimization. *IEEE Transactions on automatic control*, 48(7):1259–1263, 2003.

[20] Mohammad Khosravi and Roy S Smith. The existence and uniqueness of solutions for kernel-based system identification. *Automatica*, 148:110728, 2023.

[21] Mohammad Khosravi and Roy S Smith. Kernel-based identification with frequency domain side-information. *Automatica*, 150:110813, 2023.

[22] Mohammad Khosravi and Roy S Smith. Nonlinear system identification with prior knowledge on the region of attraction. *IEEE Control Systems Letters*, 5(3):1091–1096, 2021.

[23] Mohammad Khosravi and Roy S Smith. Kernel-based impulse response identification with side-information on steady-state gain. *IEEE Transactions on Automatic Control*, 2023.

[24] Mohammad Khosravi and Roy S Smith. Convex nonparametric formulation for identification of gradient flows. *IEEE Control Systems Letters*, 5(3):1097–1102, 2021.

[25] Paul Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, 109:475–494, 2001.

[26] Matthew Rocklin et al. Dask: Parallel computation with blocked algorithms and task scheduling. In *Proceedings of the 14th python in science conference*, volume 130, page 136. SciPy Austin, TX, 2015.

[27] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions. 2009.

[28] II Dikin. Iterative solution of problems of linear and quadratic programming. In *Doklady Akademii Nauk*, volume 174, pages 747–748. Russian Academy of Sciences, 1967.

[29] Dong C Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

[30] M Verahegen and Patrick Dewilde. Subspace model identification. Part I: The output-error state-space model identification class of algorithm. *Int. J. Control*, 56:1187–1210, 1992.