A Novel Schur Decomposition-Based Weight Projection Method for Stable State-Space Neural-Network Architectures

Sergio Vanegas
Computational Engineering
LUT University
Lappeenranta, Finland
sergio.vanegas.arias@lut.fi

Lasse Lensu
Computational Engineering
LUT University
Lappeenranta, Finland
lasse.lensu@lut.fi

Fredy Ruiz
DEIB
Politecnico di Milano
Milan, Italy
fredy.ruiz@polimi.it

Abstract

Building black-box models for dynamical systems from data is a challenging problem in machine learning, especially when asymptotic stability guarantees are required. In this paper, we introduce a novel stability-ensuring and backpropagation-compatible projection scheme based on the Schur decomposition for the state matrix of linear discrete-time state-space layers, as well as an alternative pre-factorized formulation of the methodology. The proposed methods dynamically project the quasi-triangular factor of the state matrix's real Schur decomposition onto its nearest stable peer, ensuring stable dynamics with minimal overparameterization. Experiments on synthetic linear systems demonstrate that, despite a marginal increase in computational complexity, the method achieves accuracy and convergence rates comparable to those of state-of-the-art stable-system identification techniques. Furthermore, the lower weight count facilitates convergence during training without sacrificing accuracy in stacked neural-network architectures with static nonlinearities targeting real-world datasets. These results suggest that the Schur-based projection provides a numerically robust framework for identifying complex dynamics on par with the state of the art while satisfying strict asymptotic-stability requirements. All code is available at `https://codeberg.org/sergiovaneg/SchurSS`.

1 Introduction

Recent developments in State-Space (SS) System Identification (SysID) theory have sparked renewed interest within the Machine Learning (ML) community as an alternative sequence-modelling paradigm to Transformers [1]. The SS framework models the sampled output $\boldsymbol{y}[k] \in \mathbb{R}^{n_y}, k \in \mathbb{N}_0$ of a real Linear Time-Invariant (LTI) system with exogenous input $\boldsymbol{u}[k] \in \mathbb{R}^{n_u}$ as a difference-equation system of the form

$$\begin{cases} \boldsymbol{x}[k+1] &= \boldsymbol{A}\boldsymbol{x}[k] + \boldsymbol{B}\boldsymbol{u}[k], \\ \boldsymbol{y}[k] &= \boldsymbol{C}\boldsymbol{x}[k] + \boldsymbol{D}\boldsymbol{u}[k], \end{cases} \quad s.t. \quad \begin{matrix} \boldsymbol{A} \in \mathbb{R}^{n_x \times n_x}, \ \boldsymbol{B} \in \mathbb{R}^{n_x \times n_u}, \\ \boldsymbol{C} \in \mathbb{R}^{n_y \times n_x}, \ \boldsymbol{D} \in \mathbb{R}^{n_y \times n_u}, \end{matrix} \tag{1}$$

which can be used to represent the sampled dynamics of a continuous-time system by means of a discretization scheme [2]. Even if incapable of modelling nonlinear dynamics on their own, the versatility and expressiveness of this approach have made SS subsystems the building block of choice for some of the most noteworthy sequence-modelling Neural Networks (NNs) in recent years [3–5].

The revitalization of SS theory for NNs was catalyzed by the introduction of the High-order Polynomial Projection Operators (HiPPO) framework [6], which optimizes the state vector $\boldsymbol{x}$ to maximize the representation of the exogenous-input history. This is achieved through a prescribed upper-triangular matrix $\boldsymbol{A}$ that maximizes the information compressed in the states with respect to a given

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

measure function, allowing SS models to capture long- and short-term dependencies with efficiency comparable to Transformer-based architectures. Beyond its memory capabilities, the HiPPO matrix ensures inherent stability: it is Hurwitz-stable by construction in continuous time, and its discretized counterpart remains Schur-stable regardless of the step size. To overcome the limitations of fixed state dynamics, subsequent works such as Structured State-Space Sequence (S4) and Simplified Structured Selective State-Space Sequence (*Mamba*) have utilized this matrix (and its diagonalized variant) primarily as a robust initialization strategy, rather than a static parameter [3, 5].

While high-capacity architectures are designed for the billion-parameter scale characteristic of Large Language Models (LLMs), their computational overhead is often prohibitive for time-sensitive applications such as real-time control. To address this gap, backpropagation-ready matrix parameterizations [7] and weight regularization techniques [8] targeting smaller-scale nonlinear systems have emerged. Despite these advances, methodologies for low-dimensional models remain less mature than their large-scale counterparts, requiring significant over-parameterization [7] or careful hyperparameter tuning [8] to ensure stable convergence and performance.

The present work adds to this line of research by:

1. Introducing an innovative Schur-stable state-matrix weight-projection scheme, exploiting recent results on Ω-stable nearest-matrix identification [9];
2. Proposing an alternative stable state-matrix parameterization using the above weight-projection algorithm, which trades parameter count in favour of higher computational efficiency during model training;
3. Carrying out a comprehensive benchmark of the proposed methodologies against the State of the Art (SotA) in Schur-stable state-matrix identification, targeting both synthetic and real-world datasets.

The paper is divided as follows. Section 2 contains a brief historical overview of the literature on nonlinear SS SysID. Section 3 introduces the proposed Schur-stable projection method, as well as the Ω-stable projection theory behind it. Section 4 presents a series of experiments to determine the performance of the developed method compared to the SotA, comprising both LTI SysID synthetic-data benchmarks and a performance comparison between alternative SS-block formulations within a nonlinear model on real-world datasets. Finally, Section 5 lays out some final remarks based on the experimental results and potential extensions to the showcased methods.

All code associated to this manuscript is made available at `https://codeberg.org/sergiovaneg/SchurSS` and released under the BSD-3-Clause license.

2 Related Work

While S4 [3] and *Mamba* [5] are built using the continuous-time SS formulation, and even though methods for stable identification in continuous time can be found in the literature, the discrete formulation from Equation (1) is more commonly studied as it directly maps to data-driven methodologies. Thus, unless otherwise stated, any mention of *stability* will refer to Schur stability; i.e.,

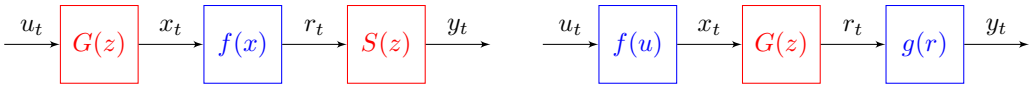
$$|\lambda_i| \leq 1 \quad \forall \lambda_i \in \Lambda_{\boldsymbol{A}},$$

where $\Lambda_{\boldsymbol{A}} \in \mathbb{C}^{n_x}$ denotes the spectrum of $\boldsymbol{A}$.

2.1 Classical Identification Methods

Although SS SysID theory was a well-established field already in the 1970s [10], modern deterministic methods can, for the most part, be traced back to the unifying theory of Subspace Identification Methods (SIMs) proposed by Van Overschee and De Moor [11], under which identified models are guaranteed to be stable provided that the source system is stable. These techniques operate by organizing available input and output data into a mathematical structure known as a Hankel matrix [12], which is shaped by the estimated system order and the available number of samples. As this matrix is constructed directly from recorded measurements, specific SIM approaches differ from each other based on how they recover the individual matrices [13].

SS blocks of the form in Equation (1) are incapable of modelling nonlinear dynamics on their own, so they usually operate as subunits of larger model structures. This concept appears in nonlinear



(a) Wiener-Hammerstein (b) Hammerstein-Wiener

Figure 1: Single-branch structures considered in Schoukens and Tiels [14], obtained by chaining LTI blocks $G(z)$ and $S(z)$, and static nonlinear blocks $f(\cdot)$ and $g(\cdot)$.

stacks like those of Figure 1 [14] and in dimensional-elevation techniques like the Koopman operator [15], which have been extensively studied in the control literature, relying on SIMs to calibrate their parameters. These frameworks are favoured in the literature not only for the simplicity of their formulation, but also because they decouple the nonlinear effects from the SS feedback loop, preserving its asymptotic-stability guarantees. However, the invertibility constraints imposed on the nonlinear blocks by conventional model-fitting algorithms (see Schoukens and Tiels [14]) limit the structure and stack dimensionality of these models.

In practice, this limitation is overcome by applying multi-level optimization methods [16]. Under this optic, it is clear how backpropagation fits the use case: not only does it benefit from the same assumptions made by previous data-driven methods (large data availability, model fitness, etc.), but it is also indifferent to the model's invertibility. In exchange, the stability guarantees given by SIMs are lost [17], necessitating a new strategy to ensure asymptotic stability.

2.1.1 Stability by Regularization

Still within the realm of classical model identification theory, but lacking a closed-form solution, it is possible to find regularized-loss identification schemes in the literature [18]. These schemes are least-squares methods penalizing the violation of asymptotic stability through an additive regularization term as

$$\min_{\boldsymbol{\Theta} \triangleq \left[\begin{smallmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{smallmatrix}\right]} \ell(\boldsymbol{\Theta}; \boldsymbol{y}, \boldsymbol{x}[0]) + \rho_r r(\boldsymbol{\Theta}) \quad s.t. \quad \ell(\boldsymbol{\Theta}; \boldsymbol{y}, \boldsymbol{x}[0]) = \frac{1}{N} \sum_{k=0}^{N-1} \mathcal{L}\left(\boldsymbol{y}[k], \hat{\boldsymbol{y}}[k; \boldsymbol{\Theta}, \boldsymbol{x}[0]]\right), \quad (2)$$

where $\mathcal{L} : \mathbb{R}^{n_y} \times \mathbb{R}^{n_y} \to \mathbb{R}$ is a suitable loss function, $\boldsymbol{y} \in \mathbb{R}^{n_y}$ denotes the measured output of the target system totaling $N \in \mathbb{N}$ training samples, $\hat{\boldsymbol{y}} \in \mathbb{R}^{n_y}$ is the output generated by the dynamics in Equation (1) given an initial condition $\boldsymbol{x}[0] \in \mathbb{R}^{n_x}$, $r : \mathbb{R}^{(n_x+n_y)\times(n_x+n_u)} \to \mathbb{R}$ is the aforementioned regularization term, and $\rho_r \geq 0$ denotes the *strength* of the regularization term.

The regularization term can assume a wide range of shapes. Sjöberg et al. [19] proposed a simple regularization term via the vector 1-norm, which equally penalizes large coefficients across all matrices, whereas Van Gestel et al. [20] used a positive semi-definite matrix as a proxy for a weighted sum of the state-matrix spectrum, which penalizes large-magnitude eigenvalues. These functions, however, fail to discriminate between stable and unstable matrices, and they could misidentify marginally stable systems, which is why Bemporad [8] very recently proposed a scheme capable of making such a distinction by means of the piecewise regularization term

$$r(\boldsymbol{\Theta}) = \max\left\{\|\boldsymbol{A}\|_2^2 - 1 + \epsilon, 0\right\}^2, \quad (3)$$

where $\|\cdot\|_2$ denotes the spectral norm and $0 \leq \epsilon \ll 1$ is added for numerical stability. Not only does this method impose a zero penalization for asymptotically stable systems, but it is also proven not to limit the coverage of the Schur-stable space [8, Lemma 1]. Nevertheless, like its peers, this stabilization method is highly dependent on the careful tuning of the parameter ρ_r, as values that are either too small or too large can result in unstable or null state matrices, respectively.

2.2 Modern Structured State-Space Models

The current SotA in SS modelling is the *Mamba* model [5]. *Mamba* models nonlinear irregularly-sampled signals, such as natural language or DNA sequencing, by combining HiPPO initialization, channel-wise variable-step discretization, Multi-Layer Perceptron (MLP) aggregation, and diagonal states. This last property not only facilitates parallelized model execution, but it also simplifies the stabilization of its dynamics.

These design choices are fitting for discrete-output systems where the concept of "regular sampling" does not apply. However, these properties also make the model unnecessarily difficult to train and run for regularly-sampled, continuous-valued systems [21]. Works like Patro and Agneeswaran [22] have tried to lighten this burden by using a single *Mamba* layer followed by a series of frequency-domain Hadamard products (time convolutions) and cross-channel matrix multiplications, at the expense of losing time-dependence locality and requiring a Fourier transform. Furthermore, diagonal real-valued state coefficients are not capable of modelling autonomous oscillations, and complex-valued state matrices require (in general) additional projections to yield a real-valued output.

While constraining the number of complex-conjugate eigenvalues could theoretically enable diagonal models to capture oscillatory dynamics, this approach requires a-priori knowledge of the state matrix's sparsity pattern. Furthermore, the reliance on complex-number arithmetic introduces significant implementation overhead and limits flexibility. Thus, a more intelligent approach to stabilizing dense state matrices is required to reliably model a broader range of dynamical systems without requiring predefined structural tuning.

2.3 SIMBa

The closest competitor in the literature to this paper's approach is System Identification Methods leveraging Backpropagation (SIMBa) [7]. The authors leverage Linear Matrix Inequalities (LMIs) to design constraint-free parameterizations guaranteeing system stability with arbitrary eigenvalue-magnitude upper bounds. Moreover, they prove that their method is capable of capturing *all* Schur-stable matrices, limited only by floating-point precision.

The only concern raised by the authors of SIMBa is regarding its training time compared to traditional identification methods. Since the projection-based methods proposed in this paper also rely on backpropagation, this metric is expected to be in the same order of magnitude. However, it is worth pointing out that SIMBa requires tuning $5n_x^2$ weights to parameterize an n_x^2-sized matrix, which marginally increases the model's memory requirements and, more importantly, induces a gradient fan-out effect that could potentially impact the model-training's convergence rate.

3 State-Matrix Stabilization via Schur Factorization

3.1 Nearest Ω-Stable Matrix Identification

Not unique to the realm of SysID, but with clear applicability, is the problem of the nearest Ω-stable matrix identification, formally posed in Definition 3.1. In the case of real discrete-time systems (i.e., as in Equation (1)), $\Omega = \Omega_S \triangleq \{\lambda \in \mathbb{C} : |\lambda| \leq 1\}$.

Definition 3.1 (Noferini and Poloni [9])**.** Let Ω be a non-empty closed subset of $\mathbb{C}$, and let

$$S(\Omega, n, \mathbb{F}) \triangleq \left\{ \boldsymbol{X} \in \mathbb{F}^{n\times n} : \Lambda_{\boldsymbol{X}} \subseteq \Omega \right\} \subseteq \mathbb{F}^{n\times n} \tag{4}$$

be the set of $n \times n$ matrices (with either $\mathbb{F} = \mathbb{C}$ or $\mathbb{F} = \mathbb{R}$) whose eigenvalues all belong to Ω.

Given $\boldsymbol{A} \in \mathbb{F}^{n\times n}$, the nearest Ω-stable matrix $\hat{\boldsymbol{A}}$ is defined as

$$\hat{\boldsymbol{A}} \triangleq \underset{\boldsymbol{X}\in S(\Omega,n,\mathbb{F})}{\arg\min} \|\boldsymbol{A} - \boldsymbol{X}\|_F^2 , \tag{5}$$

where $\|\boldsymbol{X}\|_F^2 \triangleq \sum_{i,j=1}^n |X_{ij}|^2 = \mathrm{tr}\,(\boldsymbol{X}^*\boldsymbol{X})$ denotes the *Frobenius* norm.

This problem shows up repeatedly in the systems-theory literature [23], but the non-convexity of the Schur-stable matrix space makes it notoriously hard to guarantee a global minimum. It was not until recent years that Noferini and Poloni [9] provided a solution with such a guarantee by proposing a bi-level optimization over the real Schur decomposition of $\hat{\boldsymbol{A}}$, defined as in Theorem 3.2. At every iteration, the 2×2 block-diagonal elements of $\boldsymbol{T}$ are projected to the nearest element[1] in the set $\hat{\mathcal{S}}(\boldsymbol{T}_{\boldsymbol{i},\boldsymbol{i}}) \subset \mathbb{R}^{2\times 2}$ of 15 elements at most (see Appendix A), followed by the optimization of the orthogonal factor $\boldsymbol{Z}$ over the Riemannian manifold $SO(2) \triangleq \{\boldsymbol{X} \in O(2) : \det(\boldsymbol{X}) = 1\}$.

[1] For $\boldsymbol{T}_{\boldsymbol{i},\boldsymbol{i}} \in \mathbb{R}^{1\times 1}$, $\hat{\boldsymbol{T}}_{\boldsymbol{i},\boldsymbol{i}} = \boldsymbol{T}_{\boldsymbol{i},\boldsymbol{i}}/\max\{1,|\boldsymbol{T}_{\boldsymbol{i},\boldsymbol{i}}|\}$.

Algorithm 1: Quasi-triangular Schur factor projection.

Input: Schur factor $\boldsymbol{T} \in \mathbb{R}^{n\times n}$ (quasi-triangular)
Output: Stable Schur factor $\hat{\boldsymbol{T}} \in \mathbb{R}^{n\times n}$

begin
  Initialize $i \leftarrow 1, \hat{\boldsymbol{T}} \leftarrow \boldsymbol{T}$;
  for $i \leftarrow 1$ *to* k **do** `// k from Theorem 3.2`
    if $\boldsymbol{T_{i,i}} \in \mathbb{R}^{1\times 1}$ **then**
      $\hat{\boldsymbol{T}}_{\boldsymbol{i,i}} \leftarrow \boldsymbol{T_{i,i}} / \max\{\mathbf{1}, |\boldsymbol{T_{i,i}}|\}$;
    else `// 2 × 2 block`
      Initialize $\boldsymbol{X} \leftarrow \mathbf{0_{2\times 2}}, d^2_{\min} \leftarrow \infty$;
      foreach $\boldsymbol{X'} \in \hat{\mathcal{S}}(\boldsymbol{T_{i,i}})$ **do**
        $d^2 \leftarrow \|\boldsymbol{T_{i,i}} - \boldsymbol{X'}\|_F^2$;
        if $d^2 \geq d^2_{\min}$ **then continue**;
        $\lambda_1, \lambda_2 \leftarrow \Lambda_{\boldsymbol{X'}}$;
        if $|\lambda_1| > 1$ **or** $|\lambda_2| > 1$ **then continue**;
        $d^2_{\min} \leftarrow d^2$; $\boldsymbol{X} \leftarrow \boldsymbol{X'}$
      end
      $\hat{\boldsymbol{T}}_{\boldsymbol{i,i}} \leftarrow \boldsymbol{X}$
    end
  end
end

Theorem 3.2 (Horn and Johnson [24])**.** *Given $\boldsymbol{X} \in \mathbb{R}^{n\times n}$, there exists a pair of matrices $\boldsymbol{Z} \in O(n) \triangleq \{\boldsymbol{W} \in \mathbb{R}^{n\times n} : \boldsymbol{W^\intercal W} = \boldsymbol{W W^\intercal} = \boldsymbol{I}_n\}$ real orthogonal and $\boldsymbol{T} \in \mathbb{R}^{n\times n}$ upper quasi-triangular defining the Schur decomposition; i.e., a pair such that*

$$\boldsymbol{Z^\intercal X Z} = \boldsymbol{T} = \begin{bmatrix} \boldsymbol{T_{1,1}} & \boldsymbol{T_{1,2}} & \cdots & \boldsymbol{T_{1,k-1}} & \boldsymbol{T_{1,k}} \\ \mathbf{0} & \boldsymbol{T_{2,2}} & \cdots & \boldsymbol{T_{2,k-1}} & \boldsymbol{T_{2,k}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{T_{k-1,k-1}} & \boldsymbol{T_{k-1,k}} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \boldsymbol{T_{k,k}} \end{bmatrix}, \tag{6}$$

where $\boldsymbol{T_{i,i}} \in \mathbb{R}^{1\times 1} \cup \mathbb{R}^{2\times 2}$ and $\lceil \frac{n}{2} \rceil \leq k \leq n$.

This method could, in theory, be used as is to constrain the state matrix, as it provides rigorous asymptotic-stability guarantees (unlike regularization methods) without the over-parameterization overhead introduced by LMI methods such as SIMBa. However, doing so would require solving a computationally-expensive, numerically-sensitive surrogate optimization problem for every update of $\boldsymbol{A}$ (i.e., once per training batch). Thus, its applicability to ML is limited in its current form.

3.2 The Truncated Nearest Schur-Stable Matrix Projection

To make the method proposed in Noferini and Poloni [9] feasible for SS SysID, we restrict the search space to stable matrices sharing the same orthogonal factor. Under this constraint, $\hat{\boldsymbol{A}}$ is found by first calculating the Schur decomposition $\boldsymbol{A} = \boldsymbol{Z T Z^\intercal}$ and then applying the projection scheme in Algorithm 1 to the quasi-triangular factor $\boldsymbol{T}$, which is equivalent to performing a single iteration of the bi-level optimization problem at the end of every training batch. This approach is not only computationally efficient, but also theoretically sound, as one of the results in Noferini and Poloni [9] is that this projection is guaranteed to yield the nearest quasi-triangular Schur-stable matrix.

To evaluate the performance of the fast approximation relative to the original algorithm, the following metrics are considered:

1. The Normalized Squared Frobenius Error (NSFE), defined as

$$\mathrm{NSFE}(\boldsymbol{A}, \boldsymbol{X}) = \frac{\|\boldsymbol{A} - \boldsymbol{X}\|_F^2}{\|\boldsymbol{A}\|_F^2}, \tag{7}$$

Table 1: Performance metrics for the matrix-approximation scheme from Noferini and Poloni [9] and the truncated projection. Hardware specs and extended benchmarks in Appendix B.

n	Method	NSFE	NSSR	MSVR	Time [s]
10	Bi-level Optimization	4.08e−1	5.57e−1	2.18e−11	1.02e3
	Block-Diagonal Projection	6.67e−1	4.06e−1	2.43e−6	2.43e−6
20	Bi-level Optimization	4.55e−1	6.83e−1	9.10e−9	1.24e3
	Block-Diagonal Projection	3.42e−1	5.54e−1	1.36e−6	9.49e−5
50	Bi-level Optimization	4.42e−1	7.48e−1	1.36e−3	8.20e3
	Block-Diagonal Projection	3.24e−1	6.13e−1	5.66e−2	4.76e−5
100	Bi-level Optimization	4.85e−1	7.61e−1	2.65e−2	7.05e4
	Block-Diagonal Projection	3.73e−1	6.28e−1	4.17e−1	7.03e−5

normalizes the objective function in Equation (5) to be matrix-size agnostic. The optimal value of this metric is not zero, as this would require the target matrix $\boldsymbol{A}$ to be stable. Instead, the NSFE of the truncated method should be evaluated relative to the one yielded by the bi-level optimization algorithm.

2. The Normalized Squared Spectral Radius (NSSR), defined as the optimal cost of the integer linear program

$$\min_{\boldsymbol{C}\in\{0,1\}^{n\times n}} \frac{\sum_{\lambda_i\in\Lambda_{\boldsymbol{X}}} c_{ij}|\lambda_i-\lambda_j|^2}{\sum_{\lambda_j\in\Lambda_{\boldsymbol{A}}}|\lambda_j|^2}, \quad s.t. \quad \sum_{i=1}^{n} c_{ij} = n \quad \forall j, \quad \sum_{j=1}^{n} c_{ij} = n \quad \forall i, \tag{8}$$

measures the average spectral distance relative to the original matrix eigenvalues by matching the nearest one in the approximation. This metric quantifies the *dynamics' distortion* induced by the projection method. Furthermore, with the state-observation matrix $\boldsymbol{C}$ being a free parameter, the state order of the projected matrix is not a concern. Just like the NSFE, this metric should be evaluated relative to the full projection, not to 0.

3. The Mean Squared Violation Radius (MSVR), defined as

$$\mathrm{MSVR}(\boldsymbol{X}) = \frac{1}{|\Lambda_{\boldsymbol{X}}|}\sum_{\lambda_i\in\Lambda_{\boldsymbol{X}}} \max\{|\lambda_i|-1,0\}^2, \tag{9}$$

provides a size-agnostic measure of the compliance with the Schur-stability bounds that mirrors the regularization term from Bemporad [8] in Equation (3). Since both projection methods are theoretically guaranteed to be stable, this metric evaluates the numerical stability of the algorithm in practice.

4. The execution time. It is worth noting that the truncated projection is Just-In-Time (JIT)-compiled, which is not possible for the Bi-level optimization due to its dynamic stop condition.

Table 1 reports the above metrics for the projection of various matrices with normally-distributed random coefficients, with additional cases available in Appendix B. Since the orthogonal factor $\boldsymbol{Z}$ is no longer a free variable, Algorithm 1 initially incurs a higher NSFE; yet this advantage vanishes for larger matrices as the execution-time limit of the manifold optimization leads to an early interruption of the minimization process. Moreover, the NSSR results indicate that this approach yields a better approximation of the original matrix eigenvalues, which is a more relevant property for sequence modelling. Finally, and more importantly, the full projection is several orders of magnitude slower than Algorithm 1, making the former prohibitively expensive to use as a weight constraint.

The truncated projection serves as an effective weight constraint when fitting models containing structured SS layers. For a Hammerstein-Wiener architecture, such as the one shown in Figure 1b, the identification process involves solving the following optimization problem:

$$\min_{\boldsymbol{\Theta}\triangleq\{\left[\begin{smallmatrix}\boldsymbol{A}&\boldsymbol{B}\\\boldsymbol{C}&\boldsymbol{D}\end{smallmatrix}\right],\boldsymbol{\theta_f},\boldsymbol{\theta_g}\}} \ell(\hat{\boldsymbol{\Theta}};\boldsymbol{y},\boldsymbol{x}[0]) \quad s.t. \quad \hat{\boldsymbol{\Theta}} = \left\{\begin{bmatrix}\hat{\boldsymbol{A}}&\boldsymbol{B}\\\boldsymbol{C}&\boldsymbol{D}\end{bmatrix},\boldsymbol{\theta_f},\boldsymbol{\theta_g}\right\}, \tag{10}$$

where ℓ mirrors the function defined in Equation (2), although driven by the output of the Hammerstein-Wiener model; $\boldsymbol{\theta_f}$ and $\boldsymbol{\theta_g}$ denote the parameters of the input and output nonlinearities, respectively; and $\hat{\boldsymbol{A}} = \boldsymbol{Z}\hat{\boldsymbol{T}}\boldsymbol{Z}^\intercal$ is calculated using the output of Algorithm 1. To generalize this definition, it is sufficient to apply the stabilization algorithm to the state matrix of each SS layer in the NN.

Alternatively, $\boldsymbol{A}$ can be parameterized directly through its Schur decomposition factors, removing the computational overhead of having to calculate the Schur factorization for every training step. In this case, $\boldsymbol{Z} \in \mathbb{R}^{n_x \times n_x}$ is constrained to be orthogonal via the Singular-Value Decomposition (SVD) method from Golub and Van Loan [25] (see Appendix D), while $\hat{\boldsymbol{T}}$ is stabilized using Algorithm 1, zeroing the elements below the block-diagonal. For simplicity, and without loss of generality since $\hat{\boldsymbol{Z}}$ is also a free parameter, $\hat{\boldsymbol{T}}$ is assumed to consist of 2×2 blocks, with a final 1×1 element in the bottom-right if and only if n_x is odd.

Regardless of the parameterization, compared to SIMBa, the proposed methods trade computational complexity for weight efficiency. Indeed, while the former demands $5n_x^2$ weights to yield a Schur-stable state matrix, the dynamically decomposed and pre-factorized approaches require only n_x^2 and $2n_x^2$ weights, respectively. Moreover, unlike regularization methods, Algorithm 1 guarantees Schur stability (up to numerical error) without requiring relative-strength coefficient tuning.

4 Experimental Results

In this section, the following stable-identification methods were evaluated: 1. Weight constraint over the state matrix following Algorithm 1, denoted *Schur (Proj.)* hereinafter; 2. Weight constraint over the state-matrix factors, using Algorithm 1 for the quasi-triangular matrix and the SVD method for the orthogonal factor, denoted *Schur (Built)* henceforth; 3. SIMBa parameterization [7]; 4. Weight regularization [8], as in Equation (3), denoted *Regularized* in tables and figures.

Large SS architectures, such as S4 and *Mamba*, were excluded as baselines due to their hardware overhead and lack of stability guarantees [21]. The implementation was carried out in *Keras 3* with the *JAX* backend. Despite experiments being conducted on HPC CPU cores (hardware specs in Appendices E.1 and F.1), as CPU execution is faster for these dataset sizes [7], the codebase remains GPU-compatible via a custom Schur decomposition implementation (see Appendix C).

4.1 Synthetic-Data Benchmarks

The first experiment involves the replication and extension of the random-system benchmarks from Di Natale et al. [7, Section V.A]. The **original** setup consists of 50 stable discrete-time systems ($n_x = 5, n_y = 3, n_u = 3$) with a single trajectory of 300 samples per partition (training, validation, and testing). Optimization is performed for 50,000 epochs using *AdamW* [26] (initial learning rate $\eta_0 = 1\text{e}{-3}$), retaining the weights yielding the best validation loss. Inputs follow a Generalized Binary Noise (GBN) distribution ($p = 0.1$), and unbiased Gaussian noise ($\sigma = 0.25$) is added to the training partition's output.

An analysis of the SIMBa codebase revealed that the generated systems lacked complex-conjugate eigenvalues, which is not representative of most physical systems. Furthermore, a low sample-to-parameter ratio was identified: a SIMBa layer of the aforementioned system size requires $5n_x^2 + n_x(n_u + n_y) + n_u n_y = 164$ parameters, which is more than half of the training sample count. Although the proposed Schur-based formulations are more efficient (64 and 89 parameters for the direct and parameterized formulations, respectively), additional configurations were introduced to ensure a comprehensive performance evaluation: **Extended** increases the number of samples and incorporates early stopping along with a checkpoint mechanism that restores the weights yielding the best validation loss; **Small** and **Smaller** reduce the total number of samples to **128** and **64**, respectively, to evaluate performance under low data availability; **Large** features increased system dimensionality to $(n_x, n_y, n_u) = (10, 6, 6)$ and splits the dataset into multiple sequences per partition to evaluate the methods' performance with larger n_x values and batched data.

Furthermore, the output-noise scale for the training partition of the new cases was reduced to $\sigma = 0.01$ due to the high level of uncertainty relative to the expected error, which was an issue as the accuracy gap between methods is too narrow. A detailed description of the experimental parameters for each

Table 2: Median numerical results for the synthetic-data benchmarks (uncertainty defined as half of the Inter-Quantile Range (IQR)). Epoch time is measured in seconds. Row-wise best is highlighted.

Setup	Metric	Schur (Proj.)	Schur (Built)	SIMBa	Regularized
Smaller	NMSE	(5.91 ± 5.31)e-4	(7.14 ± 8.32)e-4	$\mathbf{(3.78 \pm 3.45)e-4}$	(5.25 ± 5.01)e-4
	NSSR	(1.04 ± 5.81)e-2	(7.02 ± 11.9)e-2	$\mathbf{(1.07 \pm 5.43)e-2}$	(2.03 ± 7.62)e-2
	Epoch Time	(4.61 ± 0.30)e-3	(5.06 ± 0.32)e-3	$\mathbf{(4.37 \pm 0.27)e-3}$	(4.41 ± 0.28)e-3
	Epoch Count	(1.84 ± 0.26)e4	$\mathbf{(1.77 \pm 0.15)e4}$	(1.79 ± 0.20)e4	(2.13 ± 0.22)e4
Small	NMSE	$\mathbf{(5.00 \pm 4.18)e-4}$	(5.83 ± 4.07)e-4	(5.30 ± 4.18)e-4	(5.59 ± 4.75)e-4
	NSSR	(2.23 ± 8.92)e-2	(5.74 ± 11.5)e-2	$\mathbf{(1.69 \pm 7.27)e-2}$	(5.47 ± 13.5)e-2
	Epoch Time	(4.96 ± 0.33)e-3	(5.33 ± 0.34)e-3	(4.65 ± 0.30)e-3	$\mathbf{(4.60 \pm 0.30)e-3}$
	Epoch Count	(1.86 ± 0.28)e4	(1.76 ± 0.20)e4	$\mathbf{(1.73 \pm 0.16)e4}$	(2.04 ± 0.21)e4
Original	NMSE	(4.97 ± 7.92)e-5	(3.87 ± 7.00)e-5	$\mathbf{(3.43 \pm 6.04)e-5}$	(3.79 ± 6.99)e-5
	NSSR	(3.60 ± 10.3)e-2	$\mathbf{(9.27 \pm 33.0)e-3}$	(1.50 ± 6.11)e-2	(1.72 ± 8.46)e-2
	Epoch Time	(5.20 ± 0.39)e-3	(5.89 ± 0.46)e-3	(5.23 ± 0.40)e-3	$\mathbf{(5.17 \pm 0.38)e-3}$
	Epoch Count	$\mathbf{(5.00 \pm 0.00)e4}$			
Extended	NMSE	(5.10 ± 2.66)e-4	$\mathbf{(5.02 \pm 2.93)e-4}$	(6.11 ± 2.29)e-4	(5.29 ± 2.96)e-4
	NSSR	$\mathbf{(1.18 \pm 5.17)e-2}$	(3.80 ± 10.5)e-2	(1.32 ± 6.67)e-2	(5.17 ± 8.19)e-2
	Epoch Time	(7.47 ± 0.56)e-3	(8.08 ± 0.67)e-3	(7.51 ± 0.62)e-3	$\mathbf{(7.35 \pm 0.55)e-3}$
	Epoch Count	(1.68 ± 0.17)e4	(1.68 ± 0.14)e4	$\mathbf{(1.62 \pm 0.12)e4}$	(1.91 ± 0.23)e4
Large	NMSE	(4.49 ± 1.55)e-4	$\mathbf{(4.09 \pm 1.93)e-4}$	(6.49 ± 2.63)e-4	(5.19 ± 1.22)e-4
	NSSR	(4.61 ± 5.35)e-2	$(1.11 \pm .50)$e-1	(5.46 ± 5.54)e-2	$\mathbf{(3.64 \pm 3.86)e-2}$
	Epoch Time	(2.40 ± 0.11)e-2	(2.47 ± 0.90)e-2	(2.34 ± 0.12)e-2	$\mathbf{(2.31 \pm 0.10)e-2}$
	Epoch Count	(3.48 ± 0.17)e4	(3.40 ± 0.26)e4	$\mathbf{(3.11 \pm 0.28)e4}$	(3.45 ± 0.16)e4

case is available in Appendix E.1. The numerical results from the experiments above are shown in Table 2, whereas supplementary figures are delegated to Appendix E.2.

Despite the drastic reduction in output-noise scale, all evaluated methods demonstrate comparable performance in terms of Normalized Mean Squared Error (NMSE), managing to correctly identify the underlying dynamics, with each method's expected tracking error falling within the uncertainty range of its peers. Similar conclusions can be drawn from the NSSR and epoch time, although the latter tends to be smaller for the regularized method due to its simplicity in formulation and the low relative computational cost of the spectral norm for the considered n_x values. The true discriminant is the epoch count, which is consistently larger for the regularized method as it does not strictly guarantee stable dynamics, to the detriment of the model's convergence rate.

4.2 Real-Data Benchmarks

The second round of experiments adapts the nonlinear SysID benchmark from Bemporad [8, Section IV.C] to a stacked Hammerstein-Wiener structure (see Figure 1b), as opposed to the original single-layer discrete-time Non-Linear Time-Invariant (NLTI) SS model, and extends it to 4 additional datasets [27] and model configurations (see Appendix F.1). To accommodate these architectural changes, the original step-wise training scheme was replaced by a single-stage backpropagation cycle: assuming $\boldsymbol{x}[0] = \mathbf{0}_{\boldsymbol{n_x}}$, each method was tested over five random initializations to minimize the NMSE using the *AdamW* optimizer ($\eta_0 = 1\text{e}{-3}$) for up to 50,000 epochs (early stopping with patience of 10,000 epochs). Consistency with the original experiment was maintained by calculating the optimal initial state on the validation and test partitions via an Extended Kalman Filter (EKF) + Rauch-Tung-Striebel smoother (RTSs) algorithm [8], using the former to select the best model and the latter to get the output sequences over which the error metrics are calculated.

Table 3: Numerical results for the real-data benchmarks. Error uncertainty is defined as the standard deviation of the normalized squared differences. Median epoch time in seconds (uncertainty defined as half of the IQR). Epoch count reported for the best-performing model. Row-wise best is highlighted.

Dataset	Metric	Schur (proj.)	Schur (built)	SIMBa	Regularized
Silverbox	Test NMSE	$(1.52 \pm 7.86)\mathrm{e}{-2}$	$(1.51 \pm 7.87)\mathrm{e}{-2}$	$\mathbf{(1.50 \pm 7.97)e-2}$	$(1.53 \pm 7.49)\mathrm{e}{-2}$
	Epoch Time	$(1.59 \pm 0.02)\mathrm{e}{-1}$	$(1.42 \pm 0.01)\mathrm{e}{-1}$	$\mathbf{(1.39 \pm 0.06)e-1}$	$(1.43 \pm 0.03)\mathrm{e}{-1}$
	Epoch Count	8.36e3	**6.89e3**	1.40e4	1.05e4
CED	Test NMSE	$\mathbf{(3.30 \pm 7.40)e-2}$	$(3.70 \pm 8.79)\mathrm{e}{-2}$	$(4.12 \pm 9.92)\mathrm{e}{-2}$	$(4.08 \pm 8.49)\mathrm{e}{-2}$
	Epoch Time	$\mathbf{(6.14 \pm 0.27)e-3}$	$(6.80 \pm 0.16)\mathrm{e}{-3}$	$(6.29 \pm 0.15)\mathrm{e}{-3}$	$(6.56 \pm 0.10)\mathrm{e}{-3}$
	Epoch Count	1.14e4	**7.83e3**	2.65e4	1.75e4
EMPS	Test NMSE	$(3.21 \pm 3.24)\mathrm{e}{-3}$	$\mathbf{(3.07 \pm 4.51)e-3}$	$(4.86 \pm 5.78)\mathrm{e}{-3}$	$(2.50 \pm 5.18)\mathrm{e}1$
	Epoch Time	$(7.49 \pm 0.42)\mathrm{e}{-3}$	$(7.46 \pm 0.42)\mathrm{e}{-3}$	$(6.51 \pm 0.10)\mathrm{e}{-3}$	$\mathbf{(6.32 \pm 0.09)e-3}$
	Epoch Count	3.16e4	1.33e4	**1.31e4**	4.78e4
Industrial Robot	Test NMSE	$(6.63 \pm 12.3)\mathrm{e}{-1}$	$\mathbf{(5.76 \pm 10.9)e-1}$	$(7.25 \pm 17.5)\mathrm{e}{-1}$	$(7.40 \pm 17.5)\mathrm{e}{-1}$
	Epoch Time	$\mathbf{(1.78 \pm 0.01)e-1}$	$(1.83 \pm 0.01)\mathrm{e}{-1}$	$(1.82 \pm 0.04)\mathrm{e}{-1}$	$(1.86 \pm 0.02)\mathrm{e}{-1}$
	Epoch Count	1.25e4	**1.20e4**	2.13e4	2.95e4
Fine-Steering Mirror	Test NMSE	$(4.76 \pm 9.31)\mathrm{e}{-2}$	$(3.96 \pm 7.76)\mathrm{e}{-2}$	$\mathbf{(3.62 \pm 6.68)e-2}$	$(3.99 \pm 7.82)\mathrm{e}{-2}$
	Epoch Time	$(3.03 \pm 0.04)\mathrm{e}{-1}$	$(3.02 \pm 0.04)\mathrm{e}{-1}$	$\mathbf{(2.43 \pm 0.02)e-1}$	$(3.03 \pm 0.25)\mathrm{e}{-1}$
	Epoch Count	2.91e4	**2.33e4**	2.68e4	4.54e4

The numerical results for the best-performing initialization per method are detailed in Table 3, with supplementary results available in Appendix F.2. These reinforce the analysis made in Section 4.1: the expected accuracy of all methods falls within each other's uncertainty range (with the EMPS dataset being an outlier), yet the regularized method exhibits the slowest convergence rate. More notably, since the *Fine-Steering Mirror* dynamics are approximated using the largest model configuration (see Appendix F.1), the cost of calculating the matrix norm at each training step makes the epoch time of the regularized method equal to that of the Schur methods. From these two experimental stages, it is possible to place the proposed Schur decomposition-based methods on par with SIMBa, and to deem the former better choices w.r.t. the regularized method for deep stable SysID.

5 Conclusions

In this paper, we presented a novel, backpropagation-ready weight-projection method for estimating discrete LTI SS layers that leverages the stabilization scheme from Noferini and Poloni [9]. By introducing a direct projection method over the state-transition matrix, as well as an alternative pre-factorized formulation, this work provides a robust framework for guaranteeing SS-layer stability. This approach distinguishes itself through its ability to ensure stable dynamics without requiring hyperparameter tuning or the introduction of significant over-parameterization, which has insofar been a limitation in the literature. Experimental results across various scenarios, ranging from shallow linear systems to complex nonlinear models, demonstrate the practical utility of the proposed methodology. Both formulations consistently achieve performance on par with the SotA in terms of accuracy, epoch time, and convergence rate, suggesting that the proposed Schur-based stabilization is not only theoretically sound but also computationally efficient for stacked-model SysID.

Future research will focus on further generalizing the projection algorithm to support arbitrary upper bounds on eigenvalue magnitudes, providing the same level of flexible stability control as SIMBa while maintaining the efficiency inherent to the Schur-based approach. Furthermore, the development of a more efficient GPU-based implementation for the Schur decomposition remains a hurdle to be surpassed so that the methodology scales effectively for larger datasets and architectures. Finally, more niche LMI methods, such as the one presented in Najson [28], remain insufficiently evaluated for ML and could potentially serve as more efficient weight constraints for discrete-time SS models.

Acknowledgments and Disclosure of Funding

Funded by the European Union. Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Executive Agency (REA). Neither the European Union nor the granting authority can be held responsible for them.

This programme has received funding from the European Union through Marie Skłodowska-Curie actions under project number 101081466: SEED - Systems and Engineering Science Doctorate.

The authors wish to acknowledge *CSC –IT Center for Science, Finland*, for computational resources.

Finally, the authors deeply thank Mr. Peter Jones for his invaluable support and writing advice during the preparation of the camera-ready version of the manuscript.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[2] Ubaid M Al-Saggaf and Gene F Franklin. Model reduction via balanced realizations: an extension and frequency weighting techniques. *IEEE Transactions on Automatic Control*, 33(7):687–692, 2002.

[3] Albert Gu, Karan Goel, and Christopher Re. Efficiently Modeling Long Sequences with Structured State Spaces. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=uYLFoz1vlAC.

[4] Jimmy T.H. Smith, Andrew Warrington, and Scott Linderman. Simplified State Space Layers for Sequence Modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Ai8Hw3AXqks.

[5] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=tEYskw1VY2.

[6] Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. HiPPO: Recurrent Memory with Optimal Polynomial Projections. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1474–1487. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/102f0bb6efb3a6128a3c750dd16729be-Paper.pdf.

[7] Loris Di Natale, Muhammad Zakwan, Philipp Heer, Giancarlo Ferrari-Trecate, and Colin Neil Jones. Simba: System identification methods leveraging backpropagation. *IEEE Transactions on Control Systems Technology*, 2024.

[8] Alberto Bemporad. An L-BFGS-B approach for linear and nonlinear system identification under ℓ_1 and group-Lasso regularization. *IEEE Transactions on Automatic Control*, 70(7):4857–4864, 2025. doi: 10.1109/TAC.2025.3541018.

[9] Vanni Noferini and Federico Poloni. Nearest Ω-stable matrix via Riemannian optimization. *Numerische Mathematik*, 148(4):817–851, 2021.

[10] Karl Johan Åström and Peter Eykhoff. System identification—a survey. *Automatica*, 7(2):123–162, 1971.

[11] Peter Van Overschee and Bart De Moor. A unifying theorem for three subspace system identification algorithms. *Automatica*, 31(12):1853–1864, 1995.

[12] Bi-Qiang Mu and Han-Fu Chen. Hankel matrices for system identification. *Journal of Mathematical Analysis and Applications*, 409(1):494–508, 2014.

[13] S. Joe Qin. An overview of subspace identification. *Computers & Chemical Engineering*, 30(10):1502–1513, 2006. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2006.05.045. URL https://www.sciencedirect.com/science/article/pii/S009813540600158X. Papers form Chemical Process Control VII.

[14] Maarten Schoukens and Koen Tiels. Identification of block-oriented nonlinear systems starting from linear approximations: A survey. *Automatica*, 85:272–292, 2017.

[15] Steven L. Brunton, Marko Budišić, Eurika Kaiser, and J. Nathan Kutz. Modern koopman theory for dynamical systems. *SIAM Rev.*, 64(2):229–340, January 2022. ISSN 0036-1445. doi: 10.1137/21M1401243. URL https://doi.org/10.1137/21M1401243.

[16] Adrian Wills, Thomas B Schön, Lennart Ljung, and Brett Ninness. Identification of Hammerstein–Wiener models. *Automatica*, 49(1):70–81, 2013.

[17] Jan M Maciejowski. Guaranteed stability with subspace methods. *Systems & Control Letters*, 26(2): 153–156, 1995.

[18] Gianluigi Pillonetto, Tianshi Chen, Alessandro Chiuso, Giuseppe De Nicolao, and Lennart Ljung. *Regularized system identification: Learning dynamic models from data*. Springer Nature, 2022.

[19] J Sjöberg, T McKelvey, and L Ljung. On the use of regularization in system identification. *IFAC Proceedings Volumes*, 26(2):75–80, 1993.

[20] Tony Van Gestel, Johan AK Suykens, Paul Van Dooren, and Bart De Moor. Identification of stable models in subspace identification by using regularization. *IEEE Transactions on Automatic control*, 46(9): 1416–1420, 2002.

[21] Abdus Salam, Rasel Mahmud, Tohedul Islam, Saddam Mukta, and Swakkhar Shatabda. A comprehensive survey on Mamba: Architectures, challenges, and opportunities. *Computer*, 58(8):64–76, 2025.

[22] Badri N. Patro and Vijay S. Agneeswaran. SiMBA: Simplified Mamba-based architecture for vision and multivariate time series, 2024. URL http://arxiv.org/abs/2403.15360.

[23] Biswa Datta. *Numerical methods for linear control systems*. Elsevier, 2004.

[24] Roger A Horn and Charles R Johnson. *Matrix Analysis*, chapter 2, page 82. Cambridge University Press, 2012.

[25] Gene H Golub and Charles F Van Loan. *Computing Subspaces with the SVD*, pages 601–606. JHU press, 3rd edition, 1996. ISBN 978-0-8018-5414-9.

[26] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

[27] Maarten Schoukens. Nonlinear Benchmarks. https://github.com/MaartenSchoukens/nonlinear_benchmarks, 2026.

[28] Federico Najson. The Kalman-Yakubovich-Popov lemma for discrete-time positive linear systems. In *2012 American Control Conference (ACC)*, pages 5188–5193. IEEE, 2012.

[29] Leslie Hogben, editor. *Handbook of Linear Algebra*. Discrete Mathematics and Its Applications. Chapman & Hall/CRC, Philadelphia, PA, 2 edition, November 2013.

[30] Diederich Hinrichsen and Anthony J. Pritchard. *Mathematical Systems Theory I*. Springer Berlin Heidelberg, 2005. ISBN 9783540264101. doi: 10.1007/b137541. URL http://dx.doi.org/10.1007/b137541.

[31] Nicolas Gillis and Punit Sharma. On computing the distance to stability for matrices using linear dissipative Hamiltonian systems. *Automatica*, 85:113–121, 2017.

[32] Nicolas Gillis, Michael Karow, and Punit Sharma. Approximating the nearest stable discrete-time system. *Linear Algebra and its Applications*, 573:37–53, 2019.

[33] N. Boumal, B. Mishra, P.-A. Absil, and R. Sepulchre. Manopt, a Matlab toolbox for optimization on manifolds. *Journal of Machine Learning Research*, 15(42):1455–1459, 2014. URL https://www.manopt.org.

[34] James Townsend, Niklas Koep, and Sebastian Weichwald. Pymanopt: A Python toolbox for optimization on manifolds using automatic differentiation. *Journal of Machine Learning Research*, 17(137):1–5, 2016. URL http://jmlr.org/papers/v17/16-177.html.

[35] Peter Arbenz. Lecture Notes in Numerical Methods for Solving Large Scale Eigenvalue Problems, 2016. URL https://people.inf.ethz.ch/arbenz/ewp/lnotes.html.

[36] Mario Ahues and Francoise Tisseur. A new deflation criterion for the QR algorithm. *LAPACK Working Note*, 122, 1997.

[37] John C Gower and Garmt B Dijksterhuis. *Procrustes problems*, volume 30. Oxford University Press, 2004.

[38] Berthold KP Horn, Hugh M Hilden, and Shahriar Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. *Journal of the Optical Society of America A*, 5(7):1127–1135, 1988.

[39] Nagwa Sherif. On the computation of a matrix inverse square root. *Computing*, 46(4):295–305, December 1991. ISSN 1436-5057. doi: 10.1007/bf02257775. URL `http://dx.doi.org/10.1007/BF02257775`.

[40] Torbjorn Wigren and Johan Schoukens. Three free data sets for development and benchmarking in nonlinear system identification. In *2013 European Control Conference (ECC)*, pages 2933–2938. IEEE, 7 2013. doi: 10.23919/ecc.2013.6669201. URL `http://dx.doi.org/10.23919/ECC.2013.6669201`.

[41] Torbjörn Wigren and Maarten Schoukens. Coupled electric drives data set and reference models. Technical report, Uppsala University Sweden, 2017.

[42] Alexandre Janot, Maxime Gautier, and Mathieu Brunot. Data set and reference models of EMPS. In *Nonlinear system identification benchmarks*, 2019.

[43] Jonas Weigand, Julian Götz, Jonas Ulmen, and Martin Ruskowski. Dataset and Baseline for an Industrial Robot Identification Benchmark, 2022. URL `https://fdm-fallback.uni-kl.de/TUK/FB/MV/WSKL/0001/`.

[44] Merijn Floren, Leonardo Peri, Jeroen De Maeyer, Wim De Munter, Dirk Vandepitte, and Jean-Philippe Noël. Data-driven state-space identification and nonlinearity assessment of the CubeSpec Fine Steering Mirror. *ISMA-USD2024*, pages 2042–2052, 2024.

[45] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[46] Max Champneys, Gerben I Beintema, Roland Tóth, Maarten Schoukens, and Timothy J Rogers. Baseline results for selected nonlinear system identification benchmarks. *IFAC-PapersOnLine*, 58(15):474–479, 2024.

[47] Merijn Floren. Baseline models for the Fine Steering Mirror benchmark dataset. `https://doi.org/10.5281/zenodo.19591136`, 2026. URL `https://doi.org/10.5281/zenodo.19591136`. Version v2026.04.15.

A Computing the Nearest Real 2-by-2 Schur-Stable Matrix

This appendix is included for the sake of convenience, but it is taken almost verbatim from Noferini and Poloni [9, Section 5]. Some symbols, however, are modified to fit the notation used in this manuscript. Some proofs originally included in the source material are omitted.

First, some auxiliary results useful for the definition of the stable-candidate set $\hat{\mathcal{S}}(\boldsymbol{A})$ are presented. It is worth recalling, however, that not all elements in $\hat{\mathcal{S}}$ are guaranteed to be stable (e.g., trivially, $\boldsymbol{A} \in \hat{\mathcal{S}}(\boldsymbol{A})$), so they have to be filtered based on their eigenvalues. This property, however, is trivial to verify for 2×2 matrices, as presented in Lemma A.1.

Lemma A.1. *Let* $\boldsymbol{A} = \left[\begin{smallmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{smallmatrix}\right] \in \mathbb{R}^{2\times 2}$. *Then, its eigenvalues are given by*

$$\lambda_{\pm} = \frac{\operatorname{tr}(\boldsymbol{A}) \pm \sqrt{\operatorname{tr}(\boldsymbol{A})^2 - 4 \cdot \det(\boldsymbol{A})}}{2}. \tag{11}$$

Proof. The eigenvalues of $\boldsymbol{A}$ are the roots of the characteristic polynomial

$$\begin{aligned} p(\lambda) &= \det(\boldsymbol{A} - \lambda \boldsymbol{I}) \\ &= \left| \left[\begin{smallmatrix} A_{1,1}-\lambda & A_{1,2} \\ A_{2,1} & A_{2,2}-\lambda \end{smallmatrix}\right] \right| \\ &= (A_{1,1} - \lambda)(A_{2,2} - \lambda) - A_{1,2}A_{2,1} \\ &= \lambda^2 + \underbrace{(-A_{1,1} - A_{2,2})}_{-\operatorname{tr}(\boldsymbol{A})} \lambda + \underbrace{(A_{1,1}A_{2,2} - A_{1,2}A_{2,1})}_{\det(\boldsymbol{A})}. \end{aligned} \tag{12}$$

Applying the quadratic formula to $p(\lambda)$ yields Equation (11). $\square$

Given $\alpha \in \mathbb{R}$, the 2D rotation matrix $\boldsymbol{U}(\alpha)$ is defined

$$\boldsymbol{U}(\alpha) \triangleq \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix}, \tag{13}$$

where $\alpha \in \mathbb{R}$ is the rotation angle.

Lemma A.2. *Let $\boldsymbol{A} \in \mathbb{R}^{2\times 2}$. Then, $\exists \alpha \in [0, \pi/2)$ such that $\check{\boldsymbol{A}} = \boldsymbol{U}(\alpha)\boldsymbol{A}\boldsymbol{U}(\alpha)^\intercal$ has $\check{A}_{1,1} = \check{A}_{2,2} = \frac{1}{2}\mathrm{tr}(\check{\boldsymbol{A}})$.*

A non-empty closed set $\mathcal{S} \subseteq \mathbb{R}^{2\times 2}$ is said to be rotation-invariant if $\boldsymbol{X} \in \mathcal{S} \implies \boldsymbol{U}(\alpha)\boldsymbol{X}\boldsymbol{U}(\alpha)^\intercal \in \mathcal{S} \quad \forall \alpha \in \mathbb{R}$.

Lemma A.3. *Let $\mathcal{S} \subseteq \mathbb{R}^{2\times 2}$ be rotation-invariant, and let $\hat{\boldsymbol{A}}$ be a local minimizer of*

$$\min_{\boldsymbol{X}\in\mathcal{S}} \|\boldsymbol{A} - \boldsymbol{X}\|_F$$

for some $\boldsymbol{A} \in \mathbb{R}^{2\times 2}$ satisfying $A_{1,1} = A_{2,2}$. Then:

1. *If $A_{2,1} \neq -A_{1,2}$, then $\hat{A}_{1,1} = \hat{A}_{2,2}$.*
2. *If $A_{2,1} = -A_{1,2}$, then $\exists \alpha \in [0, \pi/2) : \hat{\boldsymbol{A}} = \boldsymbol{U}(\alpha)\check{\boldsymbol{A}}\boldsymbol{U}(\alpha)^\intercal$, where $\check{\boldsymbol{A}}$ is another local minimizer with the same objective value and $\check{A}_{1,1} = \check{A}_{2,2}$.*

A non-empty closed set $\mathcal{S} \subseteq \mathbb{R}^{2\times 2}$ is doubly rotation-invariant if $\boldsymbol{X} \in \mathcal{S} \implies \boldsymbol{U}(\alpha)\boldsymbol{X}\boldsymbol{U}(\beta)^\intercal \in \mathcal{S} \quad \forall \{\alpha, \beta\} \in \mathbb{R}^2$.

Lemma A.4. *Let $\mathcal{S} \in \mathbb{R}^{2\times 2}$ be doubly rotation-invariant, and let $\hat{\boldsymbol{A}}$ be a local minimizer of*

$$\min_{\boldsymbol{X}\in\mathcal{S}} \|\boldsymbol{A} - \boldsymbol{X}\|_F$$

for some diagonal $\boldsymbol{A} = diag(\sigma_1, \sigma_2) = \left[\begin{smallmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{smallmatrix}\right] \in \mathbb{R}^{2\times 2}$ with $\sigma_1 > 0$ and $\sigma_1 \geq \sigma_2 \geq 0$. Then:

1. *If $\sigma_1 \neq \sigma_2$, then $\hat{\boldsymbol{A}}$ is diagonal with $\hat{A}_{1,1} - \hat{A}_{2,2} \geq 0$ and $\hat{A}_{1,1} + \hat{A}_{2,2} \geq 0$.*
2. *If $\sigma_1 = \sigma_2$, then $\exists \alpha \in [0, \pi/2) : \hat{\boldsymbol{A}} = \boldsymbol{U}(\alpha)\check{\boldsymbol{A}}\boldsymbol{U}(\alpha)^\intercal$, where $\check{\boldsymbol{A}}$ is another local minimizer with the same objective value and $\check{A}_{1,1} = \check{A}_{2,2} = 0$.*

The result in Lemma A.5 is a variant of the Routh-Hurwitz stability criterion (see Hogben [29, Sec. 26.2]).

Lemma A.5. *Both roots of the polynomial $p(\lambda) = a\lambda^2 + b\lambda + c$, with $a \neq 0$, lie in the closed left half-plane $\{\lambda \in \mathbb{C} : \mathfrak{Re}(\lambda) \leq 0\}$ if and only if a, b, c are either all non-negative or all non-positive.*

From the above results, the set of 2×2 Schur-stable matrices is explicitly defined as in Lemma A.6.

Lemma A.6. $\mathcal{S}(\Omega_S, 2, \mathbb{R}) = \mathcal{S}_- \cap \mathcal{S}_0 \cap \mathcal{S}_+$, *where*

$$\mathcal{S}_0 = \{\boldsymbol{X} \in \mathbb{R}^{2\times 2} : \det(\boldsymbol{X}) \leq 1\} \tag{14a}$$

$$\mathcal{S}_\pm = \{\boldsymbol{X} \in \mathbb{R}^{2\times 2} : \pm\mathrm{tr}(\boldsymbol{X}) \leq 1 + \det(\boldsymbol{X})\} \tag{14b}$$

Proof. Let $t = \mathrm{tr}(\boldsymbol{X})$ and $d = \det(\boldsymbol{X})$ for brevity. The matrix $\boldsymbol{X}$ is Schur stable if and only if its characteristic polynomial $p(\lambda) = \lambda^2 - t\lambda + d$ has both roots inside the closed unit disk. This is equivalent to imposing that its Cayley transform [30],

$$q(\mu) \triangleq (\mu - 1)^2 p\left(\frac{\mu + 1}{\mu - 1}\right) = (1 - t + d)\mu^2 + 2(1 - d)\mu + (1 + t + d),$$

has both roots inside the (closed) left half-plane. By Lemma A.5, this holds if and only if the coefficients $1 - t + d, 2(1 - d), 1 + t + d$ are all non-negative or all non-positive. These 3 coefficients, however, cannot all be non-positive at the same time: indeed, if the three relations $1 - t + d \leq 0, 1 - d \leq 0, 1 + t + d \leq 0$ hold, then one gets

$$2 \leq 1 + d \leq t \leq -1 - d \leq -2,$$

which is impossible. Hence, Schur stability is equivalent to the three coefficients being non-negative, which are the conditions in Equation (14). $\square$

Now, to obtain an explicit formulation of the candidate set $\hat{\mathcal{S}}$, some additional expressions are necessary. Let $\mathcal{M}(\sigma_1, \sigma_2)$ be the set of critical points $(\tau_1, \tau_2) \in \mathbb{R}^2$ of the function

$$(\tau_1 - \sigma_1)^2 + (\tau_2 - \sigma_2)^2$$

subject to $\tau_1\tau_2 = 1$. The solutions to this problem can be computed exactly for any given pair (σ_1, σ_2), since solving

$$0 = \frac{d}{dt}\left((\sigma_1 - t)^2 + \left(\sigma_2 - \frac{1}{t}\right)^2\right) = \overbrace{\frac{2}{t^3}}^{\text{Never 0 for } t\in(-\infty,\infty)} \underbrace{\left(t^4 - \sigma_1 t^3 + \sigma_2 t - 1\right)}_{\text{Target Polynomial}} \tag{15}$$

amounts to computing the roots of a 4^{th}-degree polynomial. Thus, $\mathcal{M}$ has at most four elements for any choice of (σ_1, σ_2). With this definition, Lemma A.7 can be stated.

Lemma A.7. *Let $\boldsymbol{A} \in \mathbb{R}^{2\times 2}$, and $\boldsymbol{G} \in SO(2) \triangleq \{\boldsymbol{X} \in O(2) : \det(\boldsymbol{X}) = 1\}$ be a matrix such that $\check{\boldsymbol{A}} = \boldsymbol{G}^\intercal \boldsymbol{A}\boldsymbol{G}$ satisfies $\check{A}_{1,1} = \check{A}_{2,2}$ (G exists by Lemma A.2). Suppose, moreover, that $\boldsymbol{A}$ has SVD $\boldsymbol{A} = \boldsymbol{U_0}\left[\begin{smallmatrix}\sigma_{0,1} & \\ & \sigma_{0,2}\end{smallmatrix}\right]\boldsymbol{V_0}^\intercal$ and that $\boldsymbol{A} \mp \boldsymbol{I}$ have SVDs $\boldsymbol{A} \mp \boldsymbol{I} = \boldsymbol{U}_\pm\left[\begin{smallmatrix}\sigma_{\pm,1} & \\ & \sigma_{\pm,2}\end{smallmatrix}\right]\boldsymbol{V}_\pm^\intercal$. Then, the set*

$$\hat{\mathcal{S}}(\boldsymbol{A}) \triangleq \left\{\boldsymbol{A}, \hat{\boldsymbol{A}}_+, \hat{\boldsymbol{A}}_-\right\} \cup \hat{\mathcal{A}}_0 \cup \hat{\mathcal{A}}_+ \cup \hat{\mathcal{A}}_- \cup \hat{\mathcal{A}}_*, \tag{16}$$

where

$$\begin{aligned}
\hat{\boldsymbol{A}}_\pm &= \pm\boldsymbol{I} + \boldsymbol{U}_\pm\left[\begin{smallmatrix}\sigma_{\pm,1} & \\ & 0\end{smallmatrix}\right]\boldsymbol{V}_\pm, \\
\hat{\mathcal{A}}_0 &= \left\{\boldsymbol{U}_0\left[\begin{smallmatrix}\tau_1 & \\ & \tau_2\end{smallmatrix}\right]\boldsymbol{V}_0^\intercal : (\tau_1, \tau_2) \in \mathcal{M}(\sigma_{0,1}, \sigma_{0,2})\right\}, \\
\hat{\mathcal{A}}_\pm &= \left\{\boldsymbol{G}\begin{bmatrix}\pm 1 & \check{A}_{1,2} \\ 0 & \pm 1\end{bmatrix}\boldsymbol{G}^\intercal, \boldsymbol{G}\begin{bmatrix}\pm 1 & 0 \\ \check{A}_{2,1} & \pm 1\end{bmatrix}\boldsymbol{G}^\intercal\right\},
\end{aligned}$$

and

$$\hat{\mathcal{A}}_* = \left\{\boldsymbol{G}\left[\begin{smallmatrix}0 & \tau_1 \\ \tau_2 & 0\end{smallmatrix}\right]\boldsymbol{G}^\intercal : (\tau_1, \tau_2) \in \mathcal{M}(\check{A}_{1,2}, \check{A}_{2,1})\right\},$$

contains the Schur-stable matrix nearest to $\boldsymbol{A}$.

Proof. Let $\boldsymbol{A}$ be Schur-unstable (otherwise the result is trivial), and let $\hat{\boldsymbol{A}}$ be a Schur-stable matrix nearest to $\boldsymbol{A}$. Any such matrix must belong to the boundary of $\mathcal{S}_- \cap \mathcal{S}_0 \cap \mathcal{S}_+$. Observe that $\partial\mathcal{S}_0$ is the set of matrices with determinant 1, whereas $\partial\mathcal{S}_\pm$ is the set of matrices with an eigenvalue ± 1. In particular, the set $\partial\mathcal{S}_- \cap \partial\mathcal{S}_0 \cap \partial\mathcal{S}_+$ where all three constraints are active is empty, since a 2×2 matrix with both 1 and -1 as eigenvalues cannot have determinant 1. Hence, one of the following mutually exclusive cases must hold:

1. $\hat{\boldsymbol{A}} \in \partial\mathcal{S}_0$, $\hat{\boldsymbol{A}} \notin \partial\mathcal{S}_+ \cup \partial\mathcal{S}_-$;
2. $\hat{\boldsymbol{A}} \in \partial\mathcal{S}_+$, $\hat{\boldsymbol{A}} \notin \partial\mathcal{S}_0 \cup \partial\mathcal{S}_-$;
3. $\hat{\boldsymbol{A}} \in \partial\mathcal{S}_-$, $\hat{\boldsymbol{A}} \notin \partial\mathcal{S}_0 \cup \partial\mathcal{S}_+$;
4. $\hat{\boldsymbol{A}} \in \partial\mathcal{S}_0 \cap \partial\mathcal{S}_+$, $\hat{\boldsymbol{A}} \notin \partial\mathcal{S}_-$;
5. $\hat{\boldsymbol{A}} \in \partial\mathcal{S}_0 \cap \partial\mathcal{S}_-$, $\hat{\boldsymbol{A}} \notin \partial\mathcal{S}_+$;
6. $\hat{\boldsymbol{A}} \in \partial\mathcal{S}_+ \cap \partial\mathcal{S}_-$, $\hat{\boldsymbol{A}} \notin \partial\mathcal{S}_0$.

The six cases are treated separately:

1. $\hat{\boldsymbol{A}} \in \partial\mathcal{S}_0$, $\hat{\boldsymbol{A}} \notin \partial\mathcal{S}_+ \cup \partial\mathcal{S}_-$. Then, $\hat{\boldsymbol{A}}$ is a local minimizer of $\|\boldsymbol{A} - \boldsymbol{X}\|_F$ in the doubly rotation-invariant set $\partial\mathcal{S}_0$, and $\boldsymbol{\Xi} = \boldsymbol{U_0}^\intercal\hat{\boldsymbol{A}}\boldsymbol{V_0}$ is a local minimizer of $\left\|\left[\begin{smallmatrix}\sigma_{0,1} & \\ & \sigma_{0,2}\end{smallmatrix}\right] - \boldsymbol{X}\right\|_F$ in $\partial\mathcal{S}_0$.

   If $\sigma_{0,1} \neq \sigma_{0,2}$, then $\boldsymbol{\Xi}$ is diagonal by Lemma A.4; and for $\boldsymbol{\Xi} = \left[\begin{smallmatrix}\tau_1 & \\ & \tau_2\end{smallmatrix}\right]$ (with $\tau_1\tau_2 = \det(\boldsymbol{\Xi}) = 1$), it is necessary to have $(\tau_1, \tau_2) \in \mathcal{M}(\sigma_{0,1}, \sigma_{0,2})$. Hence, $\hat{\boldsymbol{A}} \in \hat{\mathcal{A}}_0$.

Otherwise, if $\sigma_{0,1} = \sigma_{0,2}$, then $\boldsymbol{\Xi} = \boldsymbol{U}(\alpha)[\begin{smallmatrix}\tau_1 & \\ & \tau_2\end{smallmatrix}]\boldsymbol{U}(\alpha)^\intercal$ and

$$\hat{\boldsymbol{A}} = \hat{\boldsymbol{A}}_{\boldsymbol{\alpha}} = \boldsymbol{U_0}\boldsymbol{U}(\alpha)[\begin{smallmatrix}\tau_1 & \\ & \tau_2\end{smallmatrix}]\boldsymbol{U}(\alpha)^\intercal\boldsymbol{V_0}^\intercal$$

for some $\alpha \in [0, \pi/2)$. If $\hat{\boldsymbol{A}}_{\mathbf{0}}$ is stable, then $\hat{\boldsymbol{A}}_{\mathbf{0}} \in \hat{\mathcal{A}}_0$ is another Schur-stable matrix nearest to $\boldsymbol{A}$, and the proof is concluded.

If $\hat{\boldsymbol{A}}_{\mathbf{0}}$ is not stable, then there is a minimum α such that $\hat{\boldsymbol{A}}_{\boldsymbol{\alpha}}$ is stable, and this is another Schur-stable matrix nearest to $\boldsymbol{A}$ in which one more constraint is active; hence, it falls in one of cases 4-6.

2. $\hat{\boldsymbol{A}} \in \partial\mathcal{S}_+$, $\hat{\boldsymbol{A}} \notin \partial\mathcal{S}_0 \cup \partial\mathcal{S}_-$. Then,

$$\min_{\boldsymbol{X}\in\partial\mathcal{S}_+} \|\boldsymbol{A} - \boldsymbol{X}\|_F = \min_{\boldsymbol{X}\in\partial\mathcal{S}_+} \|\boldsymbol{U}_+^\intercal(\boldsymbol{A} - \boldsymbol{I})\boldsymbol{V}_+ - \boldsymbol{U}_+^\intercal(\boldsymbol{X} - \boldsymbol{I})\boldsymbol{V}_+\|_F.$$

Hence, $\hat{\boldsymbol{A}}$ is a local minimizer of $\|\boldsymbol{A} - \boldsymbol{X}\|_F$ if and only if $\boldsymbol{\Xi} = \boldsymbol{U}_+^\intercal(\boldsymbol{X} - \boldsymbol{I})\boldsymbol{V}_+$ in $\partial\mathcal{S}_d \triangleq \left\{\boldsymbol{X} \in \mathbb{R}^{2\times 2} : \det(\boldsymbol{X}) = 0\right\}$: indeed, $\boldsymbol{X}$ has an eigenvalue 1 if and only if $\boldsymbol{\Xi}$ has determinant 0. Thus, either $\boldsymbol{\Xi} = [\begin{smallmatrix}\sigma_{+,1} & \\ & 0\end{smallmatrix}]$ and hence $\hat{\boldsymbol{A}} = \hat{\boldsymbol{A}}_+$, or a minimizer with the same objective value in one of cases 4-6 can be found.

3. $\hat{\boldsymbol{A}} \in \partial\mathcal{S}_-$, $\hat{\boldsymbol{A}} \notin \partial\mathcal{S}_0 \cup \partial\mathcal{S}_+$. Proof as in case 2, swapping all plus and minus signs.

4. $\hat{\boldsymbol{A}} \in \partial\mathcal{S}_0 \cap \partial\mathcal{S}_+$, $\hat{\boldsymbol{A}} \notin \partial\mathcal{S}_-$. Note that $\partial\mathcal{S}_0 \cap \partial\mathcal{S}_+$ is the rotation-invariant set of matrices with double eigenvalue $+1$ (since they must have eigenvalue $+1$ and determinant 1). $\hat{\boldsymbol{A}}$ is a minimizer if and only if $\boldsymbol{\Xi} = \boldsymbol{G}^\intercal\hat{\boldsymbol{A}}\boldsymbol{G}$ is a minimizer of $\|\check{\boldsymbol{A}} - \boldsymbol{X}\|$ in $\partial\mathcal{S}_0 \cap \partial\mathcal{S}_+$; so, by Lemma A.3, $\frac{1}{2}tr(\boldsymbol{\Xi}) = \Xi_{1,1} = \Xi_{2,2} = \frac{1}{2}tr(\hat{\boldsymbol{A}}) = +1$. Since $\det(\boldsymbol{\Xi}) = 1$, at least one amongst $\Xi_{1,2}$ and $\Xi_{2,1}$ is zero, and to minimize the projection distance, the other must be equal to the corresponding element of $\check{\boldsymbol{A}}$. Thus, $\hat{\boldsymbol{A}} \in \hat{\mathcal{A}}_+$.

5. $\hat{\boldsymbol{A}} \in \partial\mathcal{S}_0 \cap \partial\mathcal{S}_-$, $\hat{\boldsymbol{A}} \notin \partial\mathcal{S}_+$. Proof as in case 4, swapping all plus and minus signs.

6. $\hat{\boldsymbol{A}} \in \partial\mathcal{S}_+ \cap \partial\mathcal{S}_-$, $\hat{\boldsymbol{A}} \notin \partial\mathcal{S}_0$. Note that $\partial\mathcal{S}_+ \cap \partial\mathcal{S}_-$ is the rotation-invariant set of matrices with eigenvalues 1 and -1. Arguing as in case 4,

$$\boldsymbol{G}^\intercal\hat{\boldsymbol{A}}\boldsymbol{G} = \boldsymbol{\Xi} = \left[\begin{smallmatrix} 0 & \tau_1 \\ \tau_2 & 0 \end{smallmatrix}\right],\ \tau_1\tau_2 = 1,$$

as $\boldsymbol{\Xi}$ must have $\Xi_{1,1} = \Xi_{2,2} = \frac{1}{2}tr(\boldsymbol{\Xi}) = 0$ and $\det(\boldsymbol{\Xi}) = 1$. Moreover, to minimize $\|\check{\boldsymbol{A}} - \boldsymbol{\Xi}\|_F$, (τ_1, τ_2) must belong to $\mathcal{M}(\check{A}_{1,2}, \check{A}_{2,1})$. Thus, $\hat{\boldsymbol{A}} \in \mathcal{A}_*$.

□

B Projection Scheme - Full Benchmark

The experiments in this appendix, as well as those yielding the values reported in Table 1, are executed on an isolated HPC instance with 4 Xeon Gold 6230 logical cores and 8 GiB of RAM. Table 4 reports the same metrics as Table 1 for the 4 real-matrix cases originally considered in Noferini and Poloni [9, Section 7.4], which were in turn taken from Gillis and Sharma [31], Gillis et al. [32]:

1. $A_{i,j} = 2 \quad \forall (i,j) \in \{1, \dots, n\}^2$
2. $A_{i,j} = \begin{cases} -1 & i - j = -1, \\ -1 & j - i \in \{0, 1, 2, 3\}, \\ 0 & \text{otherwise.} \end{cases}$
3. Random with independent entries in $\mathcal{N}(0, 1)$.
4. Random with independent entries in $\mathcal{U}([0, 1])$.

The resulting eigenvalues are shown in the complex plane in Figure 2.

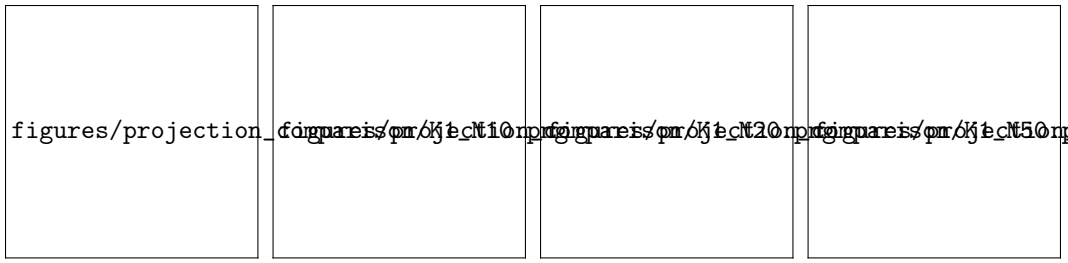
Following the setup from Noferini and Poloni [9], the maximum number of iterations and execution time were set to 1000 and 1000 seconds, respectively. However, the iteration limit only controls the

Table 4: Extended results for the truncated projection benchmark.

Case	n	Method	NSFE	NSSR	MSVR	Time [s]
1	10	Bi-level Optimization	4.75e−1	9.20e−1	6.81e−4	9.51e1
		Block-Diagonal Projection	9.02e−1	9.02e−1	1.28e−14	1.35e−4
	20	Bi-level Optimization	4.87e−1	9.46e−1	3.33e−2	1.54e2
		Block-Diagonal Projection	9.51e−1	9.56e−1	7.11e−16	8.17e−5
	50	Bi-level Optimization	4.95e−1	9.54e−1	6.21e−1	5.45e2
		Block-Diagonal Projection	9.80e−1	9.80e−1	1.14e−5	6.72e−5
	100	Bi-level Optimization	4.97e−1	9.59e−1	2.22e0	1.44e3
		Block-Diagonal Projection	9.90e−1	9.90e−1	0.00e0	7.09e−5
2	10	Bi-level Optimization	8.28e−2	3.58e−1	4.87e−7	5.34e2
		Block-Diagonal Projection	4.61e−1	4.35e−1	3.64e−13	2.93e−5
	20	Bi-level Optimization	3.87e−1	5.69e−1	2.82e−5	1.48e3
		Block-Diagonal Projection	4.61e−1	4.35e−1	1.72e−8	3.88e−5
	50	Bi-level Optimization	3.81e−1	5.73e−1	1.90e−4	9.91e3
		Block-Diagonal Projection	5.27e−1	4.74e−1	1.76e−2	4.55e−5
	100	Bi-level Optimization	3.96e−1	6.09e−1	2.96e−3	6.75e4
		Block-Diagonal Projection	4.97e−1	4.63e−1	9.56e−2	5.82e−5
3	10	Bi-level Optimization	4.08e−1	5.57e−1	2.18e−11	1.02e3
		Block-Diagonal Projection	6.67e−1	4.06e−1	2.43e−6	2.43e−6
	20	Bi-level Optimization	4.55e−1	6.83e−1	9.10e−9	1.24e3
		Block-Diagonal Projection	3.42e−1	5.54e−1	1.36e−6	9.49e−5
	50	Bi-level Optimization	4.42e−1	7.48e−1	1.36e−3	8.20e3
		Block-Diagonal Projection	3.24e−1	6.13e−1	5.66e−2	4.76e−5
	100	Bi-level Optimization	4.85e−1	7.61e−1	2.65e−2	7.05e4
		Block-Diagonal Projection	3.73e−1	6.28e−1	4.17e−1	7.03e−5
4	10	Bi-level Optimization	1.58e−1	7.15e−1	3.17e−7	1.02e3
		Block-Diagonal Projection	4.80e−1	5.59e−1	5.12e−14	6.90e−5
	20	Bi-level Optimization	1.94e−1	7.92e−1	2.35e−4	1.34e3
		Block-Diagonal Projection	6.02e−1	7.01e−1	3.34e−8	4.98e−5
	50	Bi-level Optimization	2.46e−1	7.98e−1	8.52e−2	1.44e4
		Block-Diagonal Projection	7.44e−1	8.14e−1	1.60e−6	6.82e−5
	100	Bi-level Optimization	4.37e−1	8.82e−1	3.38e−2	4.46e4
		Block-Diagonal Projection	7.92e−1	8.63e−1	1.53e−3	8.33e−5

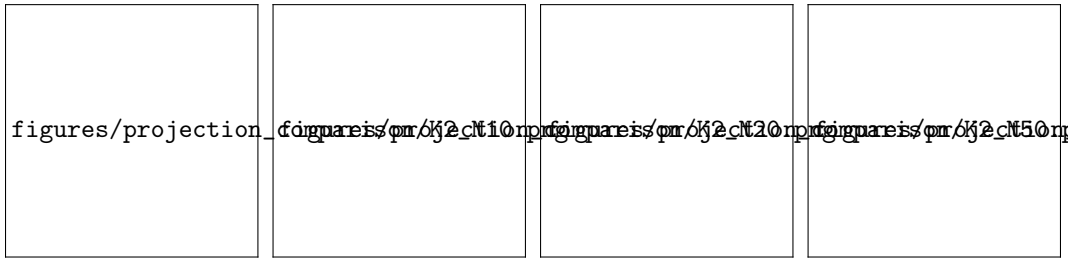
maximum number of outer iterations, meaning that the inner iteration count can be much higher. Moreover, the time-limit check is only performed once per outer iteration, explaining why some of the execution times are much longer than 1000 seconds.

The most noticeable pattern in Table 4 is the difference in execution time, which is consistent across cases. Even without running the algorithm, one would expect the truncated method to be faster since, as pointed out in Section 3.2, Algorithm 1 is equivalent to running a single inner iteration of the optimization problem. However, the actual gap turns out to be 5 orders of magnitude at best, which makes it plainly unfeasible as a projection method during training.



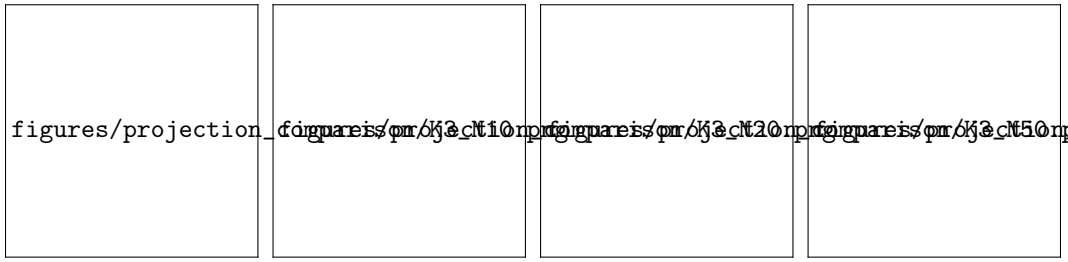
(a) $n = 10$ (b) $n = 20$ (c) $n = 50$ (d) $n = 100$

(e) Case 1



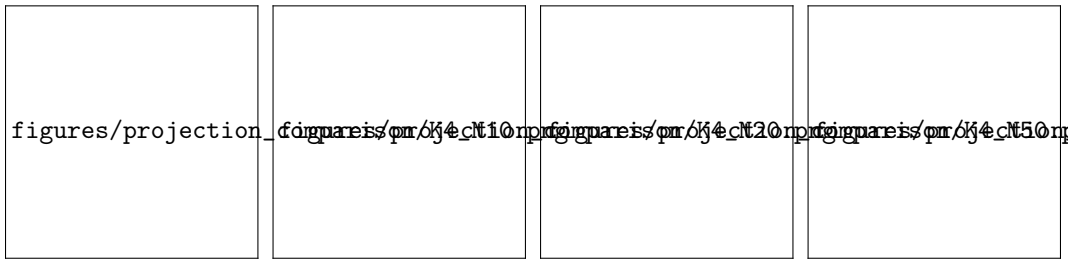
(f) $n = 10$ (g) $n = 20$ (h) $n = 50$ (i) $n = 100$

(j) Case 2



(k) $n = 10$ (l) $n = 20$ (m) $n = 50$ (n) $n = 100$

(o) Case 3



(p) $n = 10$ (q) $n = 20$ (r) $n = 50$ (s) $n = 100$

(t) Case 4

Figure 2: Eigenvalues of the reference matrix (blue), full projection (orange), and truncated projection (green) for the cases evaluated in the scheme comparison (symlog axes with unitary linear threshold).

It is worth pointing out that the timing results in this appendix are vastly different from those in Noferini and Poloni [9]. This is because their experiments were carried out in MATLAB using the *Manopt* [33] toolbox, whereas the experiments in this paper used a re-implementation of their codebase in Python using the *Pymanopt* [34] module along with its JAX interface. This was necessary to simulate the algorithm's performance as a component within a Python ML-model training pipeline, but *Pymanopt* currently lags behind the MATLAB package in terms of performance optimizations (e.g., no result caching), severely impacting its execution time.

Algorithm 2: Reduction to Hessenberg form via Householder reflections [35, Section 4.3].

Input: Matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$
Output: Matrices $\boldsymbol{H} \in \mathbb{R}^{n \times n}$ Hessenberg and $\boldsymbol{U} \in O(n)$ s.t. $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{H}\boldsymbol{U}^\intercal$

begin
  Initialize $\boldsymbol{H}, \boldsymbol{U} \leftarrow \boldsymbol{A}, \boldsymbol{I}$;
  for $k \leftarrow 1$ *to* $n-2$ **do**
    Generate the Householder reflector $\boldsymbol{P_k} \triangleq \boldsymbol{I} - 2\boldsymbol{u_k}\boldsymbol{u_k}^\intercal$ for the k-th column of $\boldsymbol{H}$;
    $\boldsymbol{H}_{k+1:n,k:n} \leftarrow \boldsymbol{H}_{k+1:n,k:n} - 2\boldsymbol{u_k}\left(\boldsymbol{u_k}^\intercal \boldsymbol{H}_{k+1:n,k:n}\right)$; `// Apply` $\boldsymbol{P_k}$ `from the left`
    $\boldsymbol{H}_{1:n,k+1:n} \leftarrow \boldsymbol{H}_{1:n,k+1:n} - 2\left(\boldsymbol{H}_{1:n,k+1:n}\boldsymbol{u_k}\right)\boldsymbol{u_k}^\intercal$; `// Apply` $\boldsymbol{P_k}$ `from the right`
  end
  for $k \leftarrow n-2$ *downto* 1 **do**
    $\boldsymbol{U}_{k+1:n,k+1:n} \leftarrow \boldsymbol{U}_{k+1:n,k+1:n} - 2\boldsymbol{u_k}\left(\boldsymbol{u_k}^\intercal \boldsymbol{U}_{k+1:n,k+1:n}\right)$; `// Update` $\boldsymbol{U} \leftarrow \boldsymbol{P_k}\boldsymbol{U}$
  end
end

Algorithm 3: The Francis (explicit) double-step QR algorithm [35, Section 4.5].

Input: Matrix $\boldsymbol{H} \in \mathbb{R}^{n \times n}$ upper Hessenberg, $\epsilon \geq 0$ relative tolerance
Output: Matrices $\boldsymbol{T} \in \mathbb{R}^{n \times n}$ quasi-triangular and $\boldsymbol{Z} \in O(n)$ s.t. $\boldsymbol{H} = \boldsymbol{Z}\boldsymbol{H}\boldsymbol{Z}^\intercal$

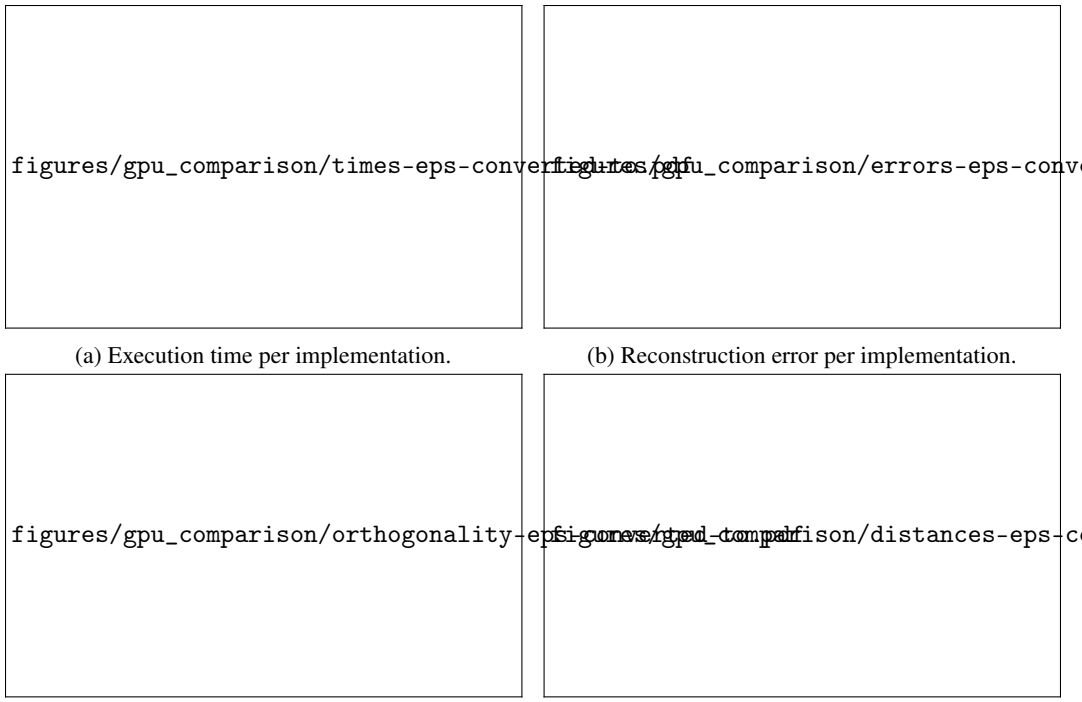
begin
  Initialize $\boldsymbol{T} \leftarrow \boldsymbol{H}, \boldsymbol{Z} \leftarrow \boldsymbol{I}$;
  $p \leftarrow n$; `//` p `indicates the` *`active`* `matrix size`
  while $p > 2$ **do**
    $\boldsymbol{G} \leftarrow \boldsymbol{T}_{p-1:p,p-1:p}$;
    $s, t \leftarrow \mathrm{tr}(\boldsymbol{g}), \det(\boldsymbol{G})$;
    $\boldsymbol{M} \leftarrow \boldsymbol{T}^2 - s\boldsymbol{T} + t\boldsymbol{I}$;
    $\boldsymbol{Q}, \boldsymbol{R} \leftarrow \mathrm{QR}(\boldsymbol{M})$; `// QR factorization of` $\boldsymbol{M}$
    $\boldsymbol{T} \leftarrow \boldsymbol{Q}^\intercal \boldsymbol{T} \boldsymbol{Q}$;
    $\boldsymbol{Z} \leftarrow \boldsymbol{Z}\boldsymbol{Q}$;
    `// Check for deflation`
    if $|\boldsymbol{T}_{p,p-1}| < \epsilon(|\boldsymbol{T}_{p-1,p-1}| + |\boldsymbol{T}_{p,p}|)$ **then** `//` 1×1 `block found`
      $\boldsymbol{T}_{p,p-1}, p \leftarrow 0, p-1$
    else if $|\boldsymbol{T}_{p-1,p-2}| < \epsilon(|\boldsymbol{T}_{p-2,p-2}| + |\boldsymbol{T}_{p-1,p-1}|)$ **then** `//` 2×2 `block found`
      $\boldsymbol{T}_{p-1,p-2} \leftarrow 0; p \leftarrow p-2$
    end
  end
end

C GPU Implementation

As of version `0.10.0`, the JAX library only implements the Schur decomposition on CPU. Moreover, we set ourselves to implement the necessary components in a (mostly) backend-agnostic way, so that the model could be instantiated as a Keras component regardless of the chosen backend library (PyTorch, JAX, ...). Thus, based on the lecture notes in Arbenz [35, Chapter 4], Algorithms 2 and 3 were implemented using the Keras OPS API. It is worth noting that the original notes recommend using the implicit Q theorem to reduce the computational overhead introduced by the QR factorization, but this optimization is targeted towards a CPU implementation and turned out to be detrimental to the algorithm's speed on the GPU during early implementations.

Some of the most notable JIT-compilable optimizations were the following:

- The final reduction for $\boldsymbol{U}$ in Algorithm 2 was replaced by an associative scan multiplying all reflectors together.
- The while loop in Algorithm 3 is replaced with a fixed-length `for` loop executing null iterations (i.e., without modifying the result) once $p \leq 2$.



(a) Execution time per implementation.

(b) Reconstruction error per implementation.

(c) Orthogonality error per implementation.

(d) Spectral error per implementation.

Figure 3: GPU benchmark figures.

- Following LAPACKS's `SLAHQR` subroutine implementation, the convergence criterion in Algorithm 3 is replaced by the one proposed in Ahues and Tisseur [36]; i.e.,

$$|\boldsymbol{X}_{2,1}||\boldsymbol{X}_{1,2}| \leq \epsilon|\boldsymbol{X}_{2,2}||\boldsymbol{X}_{2,2} - \boldsymbol{X}_{1,1}|,$$

  where $\boldsymbol{X} \in \mathbb{R}^{2\times 2}$ takes the value of $\boldsymbol{T}_{p-1:p,p-1:p}$ and $\boldsymbol{T}_{p-2:p-1,p-2:p-1}$ for the first and second deflation checks, respectively.
- Upper off-triangularity is explicitly enforced at every iteration to account for numerical error introduced by the matrix products.
- The p steps, representing the block-diagonal pattern, are stored and returned as an n-dimensional vector $\boldsymbol{b} \in \{0, 1, 2\}^n$, where a null value indicates that the corresponding element of $\boldsymbol{T}$'s diagonal is the second diagonal element of a 2×2 block.

The performance of the custom Schur decomposition implementation, running both on CPU and GPU, compared to the built-in `schur` method as a function of the matrix size n_x (illustrated in Figure 4), is evaluated using 4 metrics:

1. The execution time of each method (Figure 4a).
2. The reconstruction error (Figure 4b), defined as the NSFE between the original matrix $\boldsymbol{A}$ and the product of its Schur factors $\boldsymbol{Z}\boldsymbol{T}\boldsymbol{Z}^\intercal$. This metric indicates whether or not each implementation yields a valid factorization, regardless of the factors' orthogonality and sparsity pattern.
3. The orthogonality error (Figure 4c), defined as the NSFE between the identity matrix and the symmetric matrix $\boldsymbol{Z}\boldsymbol{Z}^\intercal$. This metric indicates whether $\boldsymbol{Z}$ fulfils the orthogonality requirement.
4. The spectral error (Figure 4c), defined as the NSSR between the original matrix A and the quasi-triangular factor $\boldsymbol{T}$. By Theorem 3.2, under a correct factorization, this metric should only be bounded by numerical error.

This experiment is executed on an isolated HPC instance with 4 Xeon Gold 6230 logical cores, 32 GiB of RAM, and a single Nvidia Volta V100 with 32 GiB of VRAM. Each metric is sampled 5 times

per n_x (20 logarithmically spaced values from 5 to 500), reporting the median as the expected value and half of the IQR as its uncertainty. For the timings, each sample corresponds to the average time of 20 consecutive runs. Likewise, each error sample is drawn from a random matrix with normally distributed coefficients.

From this experiment, the following conclusions are drawn:

- Figures 3b and 3c prove that the Schur decomposition is correctly implemented, with all approximation errors being on par with the built-in function. Although Figure 3d would at first glance seem to indicate otherwise, the scale of the NSSR lies below the level of accuracy offered by 32-bit floating-point math. However, the authors recognize that more sophisticated numerical robustness measures must have been put in place for the built-in method to yield this level of spectral accuracy.
- Unsurprisingly, a Python rewrite of a function that is originally implemented as an XLA binding results in lower performance, as illustrated by Figure 3a. Moreover, as is the case for most GPU algorithm implementations, the advantages of parallelization only outweigh the drawbacks of the slower cores for a sufficiently large array size, which in this case happens at $n_x \gtrapprox 200$. Nevertheless, it is also worth highlighting that the execution-time slope of the GPU implementation is not as steep as that of both CPU implementations, meaning that there is potential for a well-optimized GPU rewrite.

Due to the aforementioned reasons, as well as the low batch count and state sizes assessed in Section 4, experiments are carried out using the CPU and built-in Schur decomposition implementation.

D Nearest Orthogonal Matrix

As mentioned near the end of Section 3.2, it is possible to independently optimize the Schur decomposition factors of the state matrix to avoid having to compute the decomposition itself after every training batch. However, this introduces a new challenge: efficiently projecting $\boldsymbol{Z} \in \mathbb{R}^{n_x \times n_x}$ to its nearest orthogonal matrix $\hat{\boldsymbol{Z}}$; i.e.,

$$\hat{\boldsymbol{Z}} \triangleq \underset{X \in O(n_x)}{\arg\min} \|\boldsymbol{Z} - \boldsymbol{X}\|. \tag{17}$$

The constrained optimization problem in Equation (17) is a special case of the Orthogonal Procrustes problem [37], a well-studied mathematical problem with closed-form solutions and derived iterative approximations. In this appendix, the following approaches are evaluated:

1. The method from Golub and Van Loan [25], which finds the nearest orthogonal matrix by multiplying together the unitary factors from the Singular-Value Decomposition $\boldsymbol{Z} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\intercal$; i.e.
$$\hat{\boldsymbol{Z}} = \boldsymbol{U}\boldsymbol{V}^*. \tag{18}$$
2. The method from Horn et al. [38], which defines the closed form
$$\hat{\boldsymbol{Z}} = \boldsymbol{Z}\left(\boldsymbol{Z}^\intercal\boldsymbol{Z}\right)^{-1/2},$$
calculating the inverse square root of the product in parentheses through its eigenvalue decomposition $(\boldsymbol{Z}^\intercal\boldsymbol{Z}) = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{-1}$ as
$$\left(\boldsymbol{Z}^\intercal\boldsymbol{Z}\right)^{-1/2} = \boldsymbol{Q}\boldsymbol{\Lambda}^{-1/2}\boldsymbol{Q}^{-1} = \boldsymbol{Q}\begin{bmatrix} \frac{1}{\sqrt{\lambda_1}} & & \\ & \ddots & \\ & & \frac{1}{\sqrt{\lambda_{n_x}}} \end{bmatrix}\boldsymbol{Q}^{-1}. \tag{19}$$
3. The same method as in Horn et al. [38], but instead approximating the inverse square root using the stable iterative scheme from Sherif [39, Section 3]; i.e., via the recursion
$$\boldsymbol{X}^{-1/2} = \lim_{r\to\infty} \boldsymbol{\Xi_r} \; s.t. \begin{cases} \boldsymbol{\Xi_{r+1}} = \boldsymbol{\Xi_r}\left(\boldsymbol{I} + \boldsymbol{E}_r\right), & \boldsymbol{\Xi_0} = \boldsymbol{I}, \\ \boldsymbol{E_{r+1}} = \boldsymbol{E_r^2}\left(2\boldsymbol{I} - \boldsymbol{S_r^2}\right), & \boldsymbol{E_0} = \left(\boldsymbol{I} - \boldsymbol{X}\right)\left(\boldsymbol{I} + \boldsymbol{X}\right)^{-1}. \end{cases} \tag{20}$$
Since the methods are JIT-compiled during training, the stop condition for Equation (20) cannot be a function of $\boldsymbol{X}$'s coefficients; thus, its performance is evaluated by setting a fixed number of iterations equal to n_x.

The performance of the above methods, illustrated as a function of the matrix size n_x in Figure 4, is evaluated using 3 metrics:

1. The execution time of each method (Figure 4a). Execution times for the Schur-stable projection, the Schur decomposition, and the spectral norm are also included for scale and relative-cost analysis.
2. The projection distance (Figure 4b), defined as the NSFE between the original matrix $\boldsymbol{Z}$ and its projection $\hat{\boldsymbol{Z}}$. This metric should only be evaluated relative to other methods, as a null value should only be achievable if $\boldsymbol{Z}$ is already orthogonal.
3. The orthogonality numerical error (Figure 4c), defined as the NSFE between the identity matrix and $\hat{\boldsymbol{Z}}^\intercal \hat{\boldsymbol{Z}}$. Unlike the projection distance, if $\hat{\boldsymbol{Z}} \in O(n_x)$, then this metric should be approximately 0 (bounded by numerical error).

This experiment is executed on an interactive HPC instance with 4 Xeon Gold 6230 logical cores and 8 GiB of RAM. Each metric is sampled 5 times per n_x (20 logarithmically spaced values from 5 to 100), reporting the median as the expected value and half of the IQR as its uncertainty. For the timings, each sample corresponds to the average time of 100 consecutive runs. Likewise, each error sample is drawn from a random matrix with normally distributed coefficients.

From this experiment, the following conclusions are drawn:

- All orthogonal-projection methods in Figure 4a exhibit a similar slope, and can thus be considered to be of equivalent asymptotic complexity (i.e., $\mathcal{O}\left(n_x^3\right)$). Nevertheless, it is clear that the SVD method consistently outperforms the other two, although the iterative approximation could be more efficient with a dynamic termination condition. More importantly, the SVD approach also proves to be faster than the Schur decomposition, supporting the claims regarding increased computational efficiency made in Section 3.2.
- For the n_x values considered in Section 4, the execution times for the Schur-stable projection are an order of magnitude larger than those of the Schur decomposition and non-iterative nearest-orthogonal projection. That is not to say that their computational cost is negligible, as the asymptotic complexity of Algorithm 1 is $\mathcal{O}(n_x)$ (and can reach $\mathcal{O}(1)$ if the block projections are parallelized). Thus, its execution time is bound to be exceeded by all considered orthogonal-projection methods for a sufficiently large matrix size.
- All methods find very similar solutions in terms of projection distance, as illustrated by Figure 4b, but the solution found using the first method is more numerically robust, as demonstrated by Figure 4c. It is worth noting, however, that the expected orthogonality error for all methods is small enough to be acceptable for 32-bit floating-point arithmetic, and would have been a trade-off worth considering if the less precise methods were more computationally efficient. Nevertheless, as mentioned above, this is not the case.

For the aforementioned reasons, the method from Golub and Van Loan [25] is selected to constrain the orthogonal factor.

E Synthetic-Data Benchmark

E.1 Target-System & Model Parameters

Table 5 contains the experimental parameters used for the synthetic-data benchmarks. The following modifications with respect to the original experimental setup are highlighted:

- The output white-noise scale was reduced from 0.25 to 0.01. This was due to the variability of the error metrics being too large, which was not an issue in the original paper, as the performance gap between methods was more pronounced.
- For the sake of numerical stability with 32-bit floating-point arithmetic, the eigenvalue-magnitude upper bound was set to 0.99 for the *original* and *small(er)* settings. Furthermore, tighter bounds equal to 0.95 and 0.9 were necessary due to the increased sequence length and system size of the *extended* and *large* setups, respectively.

Table 5: Experimental parameters for the cases evaluated in the synthetic-data benchmarks.

Parameter	Setup				
	Smaller	**Small**	**Original**	**Extended**	**Large**
System Order (n_x, n_u, n_y)	$5, 3, 3$	$5, 3, 3$	$5, 3, 3$	$5, 3, 3$	$10, 6, 6$
No. of Systems	100	100	50	100	20
Sequences per Partition	1	1	1	1	8
Samples per Sequence	64	128	300	1024	512
Output Noise Scale (σ)	0.01	0.01	0.25	0.01	0.01
$\lvert\lambda\rvert$ upper bound	0.99	0.99	0.99	0.95	0.90
Epoch Limit	5e4	5e4	5e4	5e4	1e5
Patience	1e4	1e4	∞	1e4	1e4
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW
Learning Rate	1e−3	1e−3	1e−3	1e−3	1e−4

- The number of systems was doubled for all experiments of equal system size to further reduce the uncertainty via increased sample size. However, this was not necessary for the *large* setup, as the variability of the results was already low enough with 20 random systems.

Other parameter changes follow a similar line of reasoning. Each experiment is carried out on an interactive HPC instance with 4 Xeon Gold 6230 logical cores and 6 GiB of RAM.

E.2 Supplementary Results

The best and worst-case outputs per experimental setup (*Large* excluded due to the large channel and sequence count) are shown in Figure 6. Consistent with the expected error and uncertainty reported in Section 4.1, the *Smaller* setup is the only one where the low sample count results in a larger probability of misidentification and signal drift. For all other configurations, even in the worst-case scenario, all the evaluated methods manage to converge (albeit at different speeds, as shown in Figure 5) to a proper approximation of the target system.

F Real-Data Benchmarks

F.1 Datasets & Model Configurations

The following datasets, obtained from the official dataloader repository of `nonlinearbenchmark.org/` [27] (released under the BSD 3 license), were used for the real-data experiments in Section 4.2:

1. **Silverbox** [40]: Electronic implementation of the Duffing oscillator, built as a 2nd order LTI system with a 3rd degree polynomial static nonlinearity around it in feedback.
2. **CED (Coupled Electric Drives)** [41]: A setup consisting of two electric motors that drive a pulley using a flexible belt. The pulley is held by a spring, resulting in a lightly damped dynamic mode. The electric drives can be individually controlled, allowing the tension and the speed of the belt to be simultaneously controlled. The drive control is symmetric around zero; hence, both clockwise and counterclockwise movement are possible. The focus is only on the speed control system, and the angular speed of the pulley is measured as an output with a pulse counter insensitive to the sign of the velocity. The available data sets are short, which constitutes a challenge when performing identification.
3. **EMPS (Electro-Mechanical Positioning System)** [42]: A standard configuration of a drive system for the prismatic joint of robots or machine tools. The main source of nonlinearity is caused by friction effects that are present in the setup. Due to the presence of a pure integrator in the system, the measurements are taken in a closed-loop setting. The original dataset was sampled at 1 kHz, but it had to be subsampled by a factor of 20 due to the slow dynamics of the system and the magnitude of the input perturbations.

Table 6: Dataset properties.

Dataset	n_u	n_y	No. of Samples		
			Training	Validation	Test
Silverbox	1	1	52050	13012	21688
CED	2	2	320	80	100
EMPS	1	1	994	248	1242
Industrial Robot	6	6	31990	7998	3636
Fine-Steering Mirror	3	3	73728	73728	73728

Table 7: Real-data benchmark parameters.

(a) Model sizes per dataset.

Dataset	n_f	n_x	n_g	LR
Silverbox	5	2	2	1e−2
CED	7	3	5	1e−2
EMPS	10	4	7	1e−3
Industrial Robot	36	12	24	1e−3
Fine-Steering Mirror	16	28	8	1e−3

(b) Shared parameters.

Parameter	Value
Optimizer	AdamW
Initial $\|\lambda\|$ upper-bound	0.95
No. of initializations	5
Maximum number of epochs	5e4
Patience	5e3
Minimum delta	1e−3
EKF+RTS estimator iterations	20

4. **Industrial Robot** [43]: An identification benchmark dataset for a full robot movement with a KUKA KR300 R2500 ultra SE industrial robot. The robot has a nominal payload capacity of 300 kg, a weight of 1120 kg, and a reach of 2500 mm, exhibiting 12 states accounting for position and velocity for each of the 6 joints. The robot encounters backlash in all joints, pose-dependent inertia, pose-dependent gravitational loads, pose-dependent hydraulic forces, pose- and velocity-dependent centripetal and Coriolis forces, as well as nonlinear friction, which is temperature-dependent and therefore potentially time-varying.

5. **Fine-Steering Mirror** [44]: A multi-input multi-output high-precision control platform used in a small satellite. Voltages applied to three piezo-actuators serve as inputs, while the mirror displacements measured at three non-collocated reference points serve as outputs. The excitation signals are orthogonal random-phase multisines spanning a wide frequency range, and they are applied at three different amplitude levels. The system behaves mostly linearly, but the presence of hysteresis in the piezo-actuators introduces dynamic nonlinearities, making the dataset well-suited for benchmarking nonlinear identification methods.

The window of highest output variance per partition for each dataset can be observed in Figure 7. The data and partitioning schemes were taken from Schoukens [27] (details in Table 6), with the last 20% of the samples being reserved for validation when the training and validation samples are merged together (datasets 1-4).

Each experiment is carried out on an interactive HPC instance with 4 Xeon Gold 6230 logical cores and 4 GiB of RAM. Moreover, to facilitate best-model selection and real-time monitoring, the progress of the experiment was logged using *Optuna* [45] and hosted on a private PostgreSQL instance. Thus, to replicate these results, either a PostgreSQL instance has to be set up, or the script has to be modified to use a local SQLite file (which should not be used when benchmarking multiple datasets concurrently).

As mentioned in Section 4.2, the model follows a Hammerstein-Wiener structure with the following components:

Table 8: Channel-wise NMSE for multiple-output datasets.

Dataset	Channel	Schur (proj.)	Schur (built)	SIMBa	Regularized
CED	y_1	$(4.39 \pm 9.37)\mathrm{e}{-2}$	$(5.33 \pm 11.6)\mathrm{e}{-2}$	$(6.07 \pm 13.4)\mathrm{e}{-2}$	$(5.79 \pm 10.2)\mathrm{e}{-2}$
	y_2	$(2.22 \pm 4.40)\mathrm{e}{-2}$	$(2.08 \pm 3.72)\mathrm{e}{-2}$	$(2.18 \pm 3.32)\mathrm{e}{-2}$	$(2.37 \pm 5.83)\mathrm{e}{-2}$
Industrial Robot	y_1	$(4.71 \pm 6.66)\mathrm{e}{-1}$	$(3.91 \pm 5.53)\mathrm{e}{-1}$	$(4.46 \pm 6.06)\mathrm{e}{-1}$	$(5.66 \pm 1.32)\mathrm{e}{-1}$
	y_2	$(5.71 \pm 1.10)\mathrm{e}{-1}$	$(6.67 \pm 1.63)\mathrm{e}{-1}$	$(4.90 \pm 8.44)\mathrm{e}{-1}$	$(7.71 \pm 20.5)\mathrm{e}{-1}$
	y_3	$(4.38 \pm 6.79)\mathrm{e}{-1}$	$(5.22 \pm 8.46)\mathrm{e}{-1}$	$(5.15 \pm 8.09)\mathrm{e}{-1}$	$(4.48 \pm 6.74)\mathrm{e}{-1}$
	y_4	$(7.14 \pm 11.5)\mathrm{e}{-1}$	$(6.53 \pm 11.3)\mathrm{e}{-1}$	$(8.60 \pm 13.9)\mathrm{e}{-1}$	$(9.92 \pm 26.7)\mathrm{e}{-1}$
	y_5	$(1.21 \pm 2.16)\mathrm{e}{0}$	$(6.83 \pm 12.2)\mathrm{e}{-1}$	$(1.31 \pm 3.63)\mathrm{e}{0}$	$(9.82 \pm 19.5)\mathrm{e}{-1}$
	y_6	$(5.69 \pm 7.35)\mathrm{e}{-1}$	$(5.39 \pm 7.84)\mathrm{e}{-1}$	$(7.25 \pm 9.94)\mathrm{e}{-1}$	$(6.82 \pm 8.79)\mathrm{e}{-1}$
Fine-Steering Mirror	y_1	$(3.02 \pm 5.57)\mathrm{e}{-2}$	$(2.60 \pm 5.00)\mathrm{e}{-2}$	$(2.46 \pm 4.25)\mathrm{e}{-2}$	$(2.59 \pm 4.73)\mathrm{e}{-2}$
	y_2	$(5.35 \pm 1.16)\mathrm{e}{-2}$	$(4.52 \pm 9.58)\mathrm{e}{-2}$	$(3.29 \pm 7.15)\mathrm{e}{-2}$	$(4.57 \pm 9.81)\mathrm{e}{-2}$
	y_3	$(5.91 \pm 9.47)\mathrm{e}{-2}$	$(4.75 \pm 7.82)\mathrm{e}{-2}$	$(5.11 \pm 7.80)\mathrm{e}{-2}$	$(4.81 \pm 7.87)\mathrm{e}{-2}$

- The input nonlinearity f is implemented using a Single-Layer Perceptron (SLP) with n_f neurons and the Sigmoid Linear Unit (SiLU) activation function. An output layer here would be redundant, since the output of f is linearly projected by $\boldsymbol{B}$.
- The SS layer, of state-size n_x, is *transparent*; i.e., $\boldsymbol{C} = \boldsymbol{I}_{\boldsymbol{n_x},\boldsymbol{n_x}}$ and $\boldsymbol{D} = \boldsymbol{0}_{\boldsymbol{n_x},\boldsymbol{n_f}}$.
- The output nonlinearity g is implemented using a MLP with a single hidden layer of size n_g and the SiLU activation function (only for the hidden layer).

The model dimensions, reported in Table 7a, were taken from Champneys et al. [46] for the first 3 datasets, and from Bemporad [8] and Floren [47] for datasets 4 and 5, respectively. On the other hand, the learning rate was initially set to $1\mathrm{e}{-3}$ for all datasets and then increased to $1\mathrm{e}{-2}$ if this change only impacted the convergence rate across the board and not the expected accuracy. Shared experimental parameters are reported in Table 7b.

F.2 Supplementary Results

The channel-wise error for the multiple-output datasets is reported in Table 8. The validation NMSE history is reported only for the best-performing model; i.e., the one yielding the error metrics in Tables 3 and 8. Finally, all test-partition outputs are presented in Figures 9 to 13 (once again, only for the best-performing model).

The following analysis elements are highlighted:

- Consistent with the numbers reported in Table 3, the regularized method consistently exhibits slow convergence rates, with this trend only intensifying as n_x grows larger. Conversely, the Schur (built) method always has the steepest (or at least second-steepest) convergence rate, which is consistent with the results obtained in Section 4.1.
- While all other models manage to almost perfectly replicate the output signal of the EMPS dataset, the regularized method is incapable of capturing its dynamics, as illustrated by Figure 11. This dataset turned out to be more challenging than initially expected, as the input high-frequency components visible in Figure 7c led the models to oversensitive local minima, obstructing accurate identification.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The theoretical claims are properly justified in Section 3, whereas the claim of "real-world" performance matching the SotA is backed by the results in Section 4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The abstract is very clear about the concrete advantage of our method w.r.t. the SotA, which is the lower parameterization overhead (compared to SIMBa) and stability guarantees with no relative-strength tuning (as opposed to the regularized method). Moreover, we exclude methodologies targeting larger architectures and using continuous-time SS representations that do not provide strict asymptotic stability guarantees, as is the case with S4 and *Mamba*.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The only potentially new theoretical result concerns the lower-bound for the NSFE of the truncated projection in Section 3.2, which is trivially given by that of the full projection from Noferini and Poloni [9]. All other theoretical assumptions are properly cited, with the nearest 2×2 Schur-stable matrix theory being summarized in Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All scripts and data, accompanied by instructions on how to run the scripts and fixed random seeds, are provided in the git repository.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: All scripts used to generate the results and figures in this work are provided as a public git repository, including a README file matching each script to the corresponding section of the manuscript and instructions on how to run them.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: A detailed description of the datasets and experimental setup, as well as the reasoning behind these, is presented in Appendices E.1 and F.1 for the synthetic-data and real-data benchmarks, respectively.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: All error metrics and execution times are reported along with their uncertainty. Continuous measurements (i.e., error metrics coming from a single system/sequence) are reported using the mean of squares as the expected value and the variance as its uncertainty, whereas discontinuous measurements (i.e., metrics coming from multiple systems and timing samples) are instead reported using the median value and half of the IQR. The only exception was the projection-method comparison in Tables 1 and 4, where the execution time of the original algorithm from Noferini and Poloni [9] made multiple executions prohibitively expensive.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: CPU, GPU, and RAM specs are provided per experiment on Appendices B to D, E.1 and F.1. As storage consumption lies in the order of a few GiB, it is not reported. Finally, even if preliminary experiments were executed on the researchers' personal computers, no metrics in the manuscript are drawn from these, so these specs are not reported either.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

Justification: Our work complies with the NeurIPS code of ethics, as our experiments involved no human subjects and exclusively made use of random-synthetic data and properly-credited, publicly-available datasets. Furthermore, all researchers were fairly compensated for their contribution to the present work.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here, especially since the experiments target physical (as opposed to human-population) systems.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: No high-risk model/data is handled in this paper.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.

- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The synthetic-benchmark data has no license, as the systems are randomly generated during testing and cannot be copyrighted. The real-data benchmark datasets are jointly credited in Section 4.2, with their individual sources attributed in Appendix F.1 (licenses omitted as they are not listed in the sources).

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: While a custom (and documented) Python module is included as part of the supplementary material, it is not the authors' current intention to release this as a general-purpose SysID library.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No human subjects were involved in this paper's experiments.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No human subjects were involved in this paper's experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

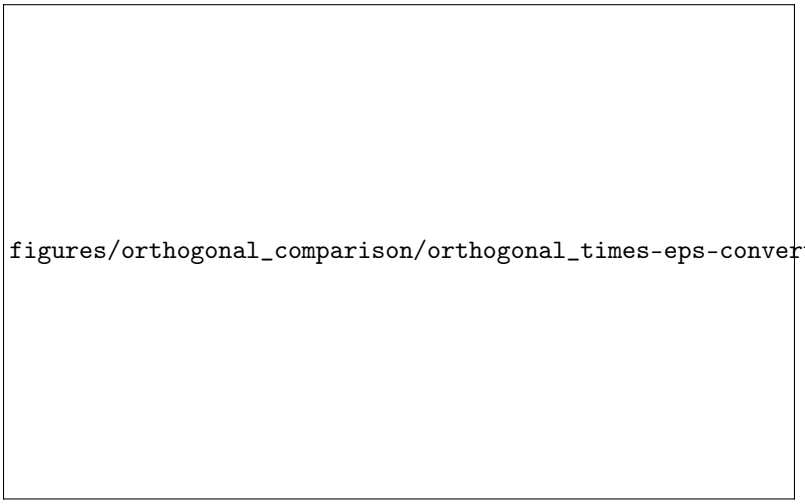
    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

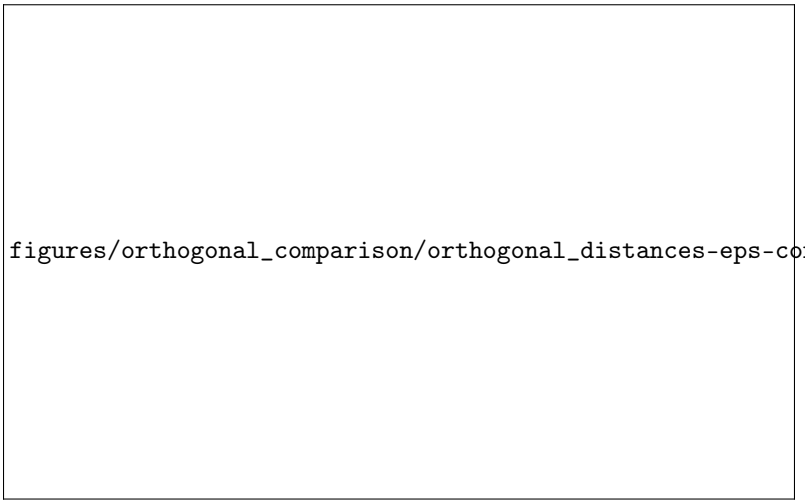
    Justification: The method and experiments target small-scale SS architectures and physical systems. While related methods mentioned in Section 2 have been used for Natural-Language Processing (NLP), and there is no reason why the weigh-projection method could not be used in an LLM, our research did not assess the algorithm's performance in NLP.

    Guidelines:

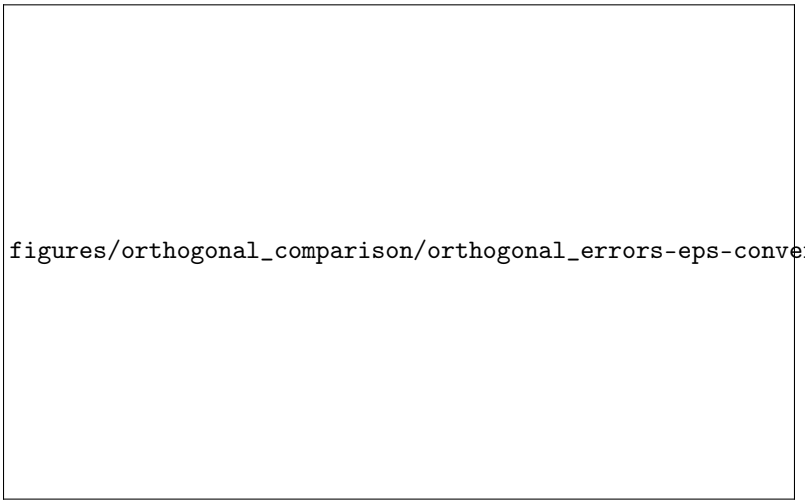
    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.



(a) Execution time per method. The Schur-stable projection, Schur decomposition, and spectral norm execution times are included for reference.

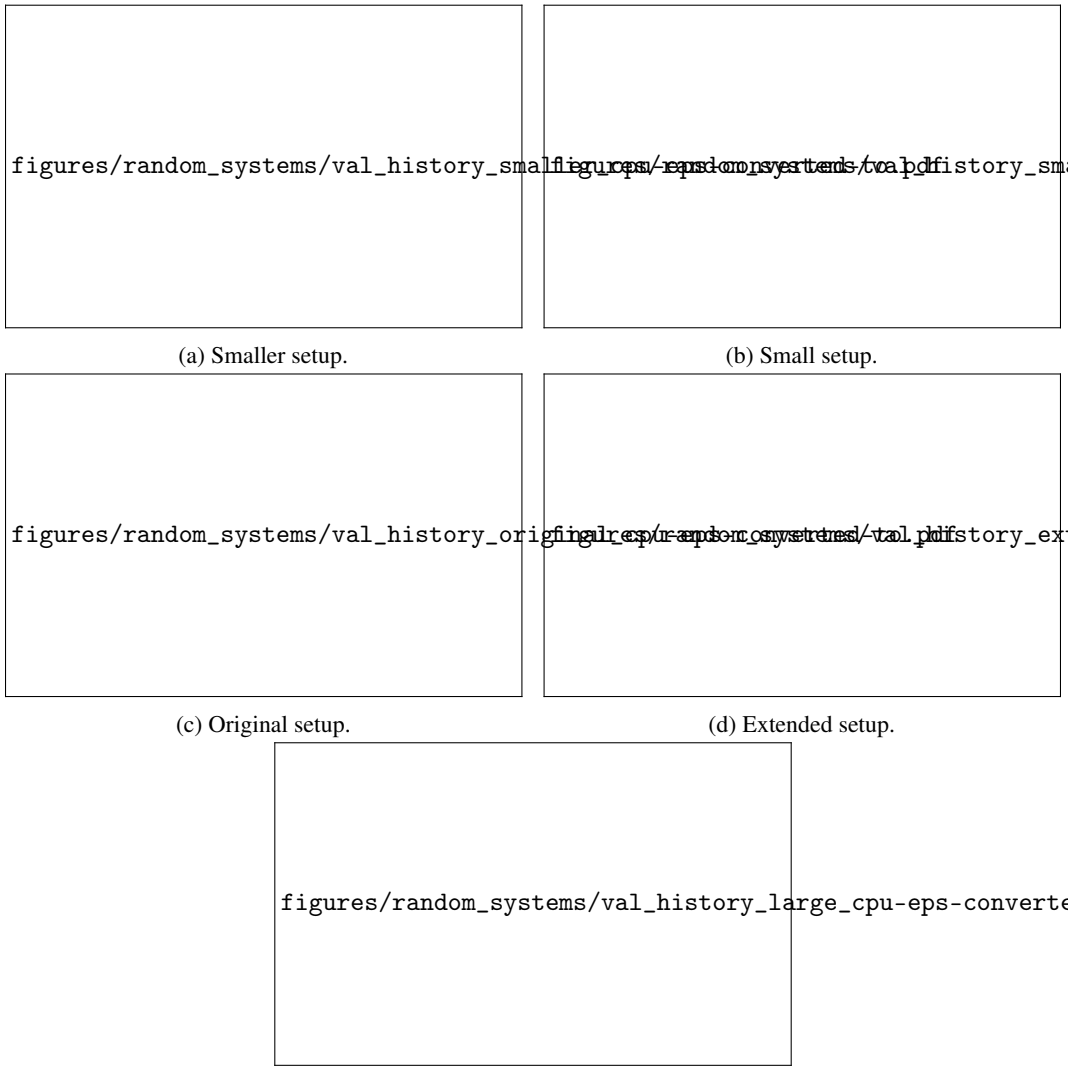


(b) Projection error per method. The high error for the iterative methods at $n_x = 5$ can be attributed to insufficient iterations.



(c) Orthogonality numerical error per method.

Figure 4: Orthogonal benchmark figures.



(a) Smaller setup.

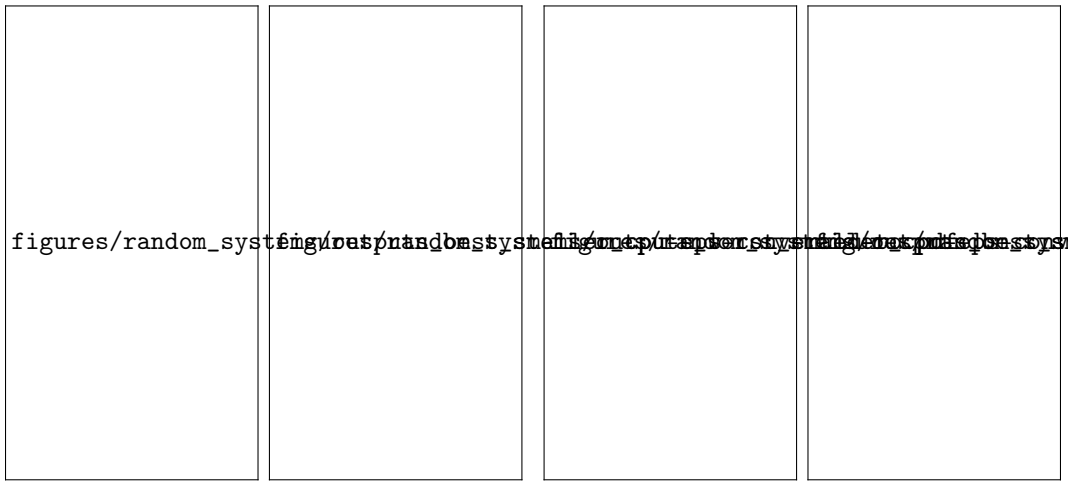
(b) Small setup.

(c) Original setup.

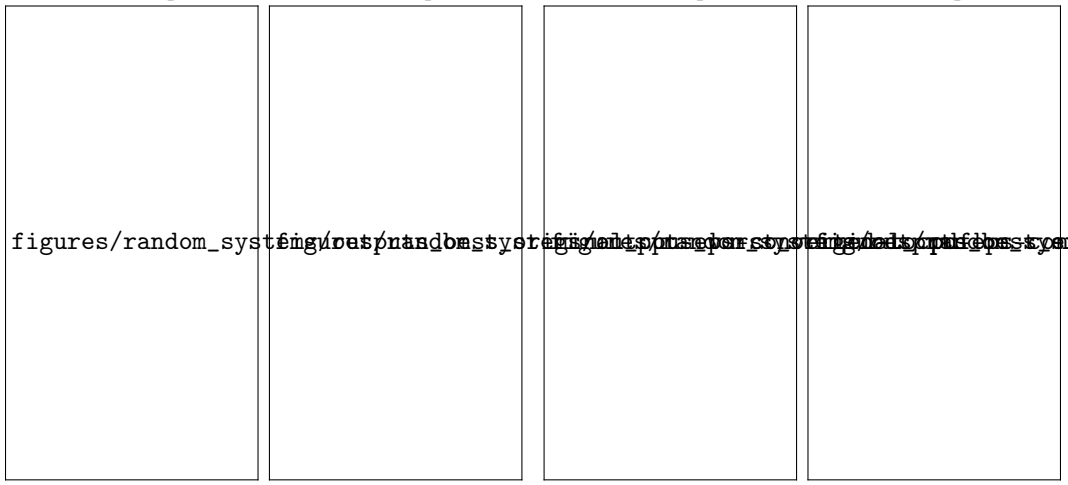
(d) Extended setup.

(e) Large setup.

Figure 5: Effective (cumulative minimum) validation-loss history (normalized w.r.t. the initial value) median for the Extended setup. The training length in epochs corresponds to the median number of iterations, whereas the shaded areas' lower and upper bounds correspond to the 25^{th} and 75^{th} percentiles per epoch, respectively.

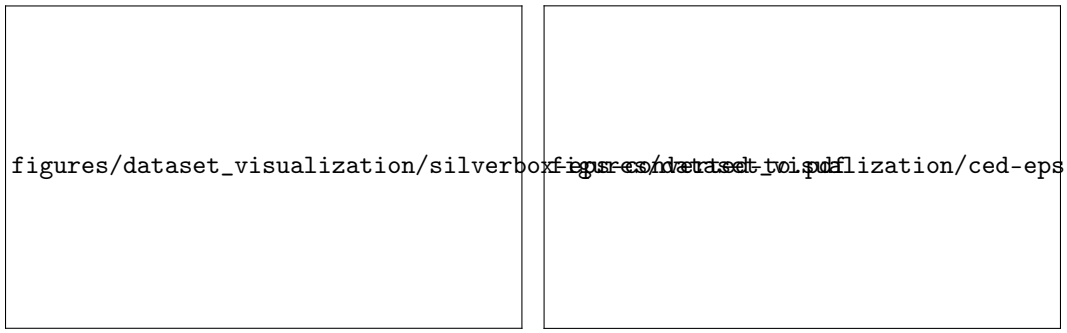


(a) Smaller setup (best). (b) Smaller setup (worst). (c) Small setup (best). (d) Small setup (worst).



(e) Original setup (best). (f) Original setup (worst). (g) Extended setup (best). (h) Extended setup (worst).

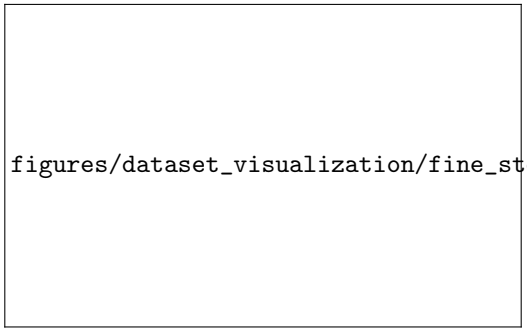
Figure 6: Best and worst cases amongst the random systems (testing partition outputs).



(a) Silverbox (b) CED

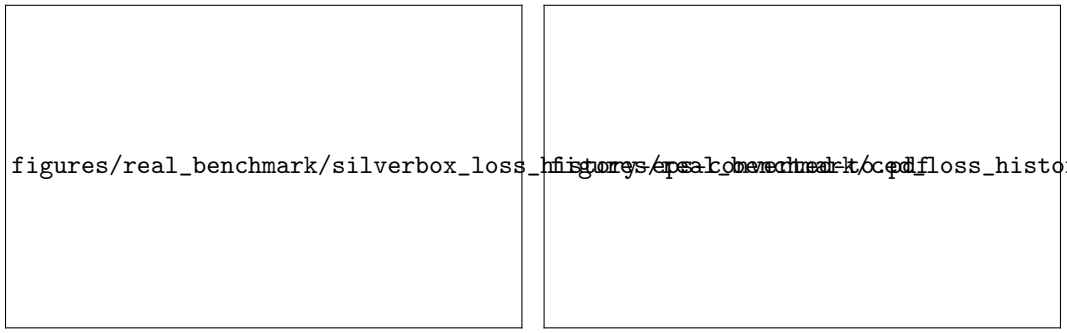


(c) EMPS (d) Industrial Robot



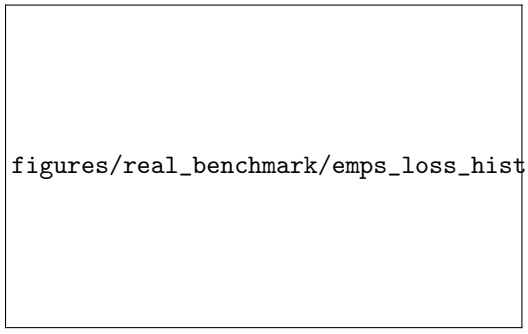
(e) Fine-Steering Mirror

Figure 7: Real-world dataset visualization.

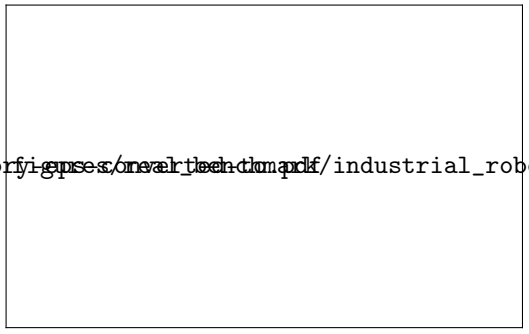


(a) Silverbox

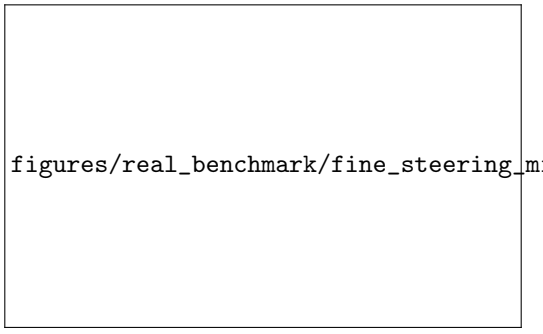
(b) CED



(c) EMPS



(d) Industrial robot



(e) Fine-steering mirror

Figure 8: Validation NMSE history for the best-performing models. The shaded lines correspond to the recorded values per epoch, whereas the solid curves denote the cumulative minima.

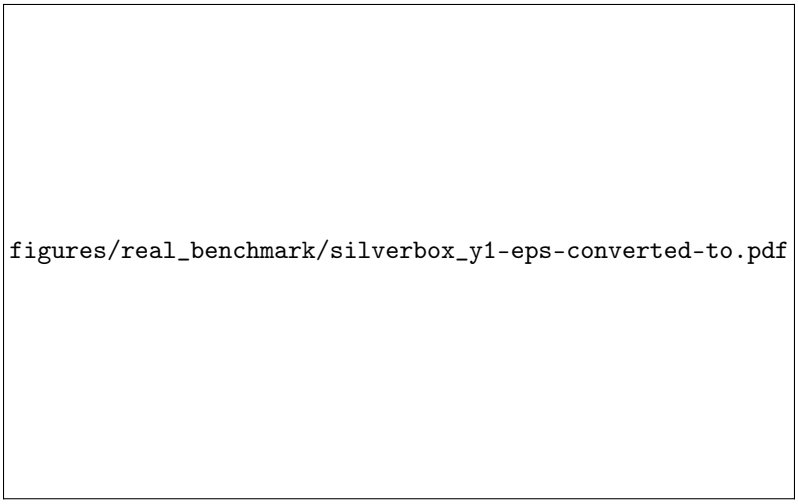
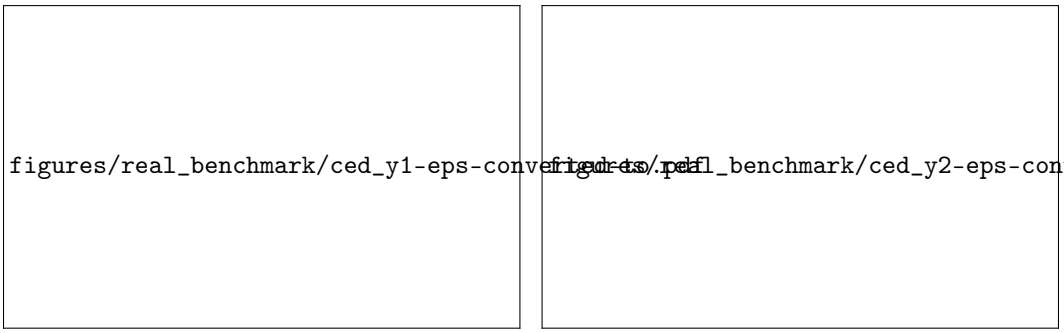


Figure 9: Silverbox test-partition output



(a) y_1 (b) y_2

Figure 10: CED test-partition output

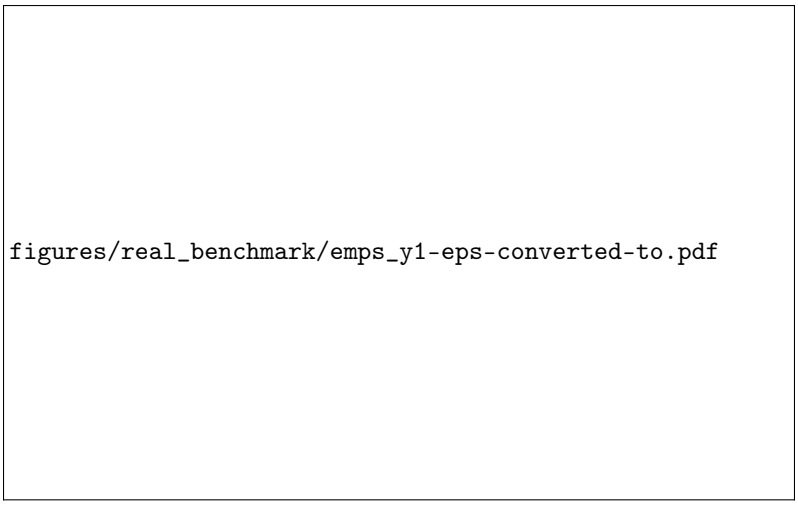


Figure 11: EMPS test-partition output



(a) y_1 (b) y_2

(c) y_3 (d) y_4

(e) y_5 (f) y_6

Figure 12: Industrial robot test-partition output



(a) y_1



(b) y_2



(c) y_3

Figure 13: Fine-Steering Mirror test-partition output (first sequence)