SpinSVAR: Estimating Structural Vector Autoregression Assuming Sparse Input

Panagiotis Misiakos[1] **Markus Püschel**[1]

[1]Department of Computer Science, ETH Zurich, Zürich, Switzerland

Abstract

We introduce SpinSVAR, a novel method for estimating a (linear) structural vector autoregression (SVAR) from time-series data under a sparse input assumption. Unlike prior approaches using Gaussian noise, we model the input as independent and identically distributed (i.i.d.) Laplacian variables, enforcing sparsity and yielding a maximum likelihood estimator (MLE) based on least absolute error regression. We provide theoretical consistency guarantees for the MLE under mild assumptions. SpinSVAR is efficient: it can leverage GPU acceleration to scale to thousands of nodes. On synthetic data with Laplacian or Bernoulli-uniform inputs, SpinSVAR outperforms state-of-the-art methods in accuracy and runtime. When applied to S&P 500 data, it clusters stocks by sectors and identifies significant structural shocks linked to major price movements, demonstrating the viability of our sparse input assumption.

1 INTRODUCTION

Time series arise in numerous applications where multidimensional observations are recorded at regular intervals, such as meteorology [Yang et al., 2022], finance [Kleinberg, 2013, Jiang and Shimizu, 2023], and brain imaging [Smith et al., 2011]. A fundamental challenge in analyzing time series is causal discovery, which seeks to uncover causal dependencies over time [Assaad et al., 2022b, Hasan et al., 2023]. If causal effects occur faster than the data's temporal resolution, they appear instantaneous and can be modeled with a linear structural equation model (SEM) [Peters et al., 2017]. When the resolution is higher, they appear as lagged effects, typically captured by vector autoregression (VAR) [Kilian, 2013]. Regardless of the model, recovering true causal relationships requires additional assumptions, such as the absence of latent confounders, identifiability conditions, or access to interventions [Vowels et al., 2021], which rarely hold in real-world settings. For instance, in financial markets, it is nearly impossible to observe all hidden confounders or directly intervene in stock prices. Instead of identifying true causal effects, we focus on learning instantaneous and lagged dependencies through a (linear) structural vector autoregression (SVAR), which unifies a linear SEM and a VAR [Hyvärinen et al., 2010].

Structural vector autoregression Originally introduced by Sims [1980], SVAR has been widely applied in econometrics [Lütkepohl, 2005, Kilian, 2013] and serves as a foundation for causal discovery in time-series data [Pamfil et al., 2020]. SVAR models linear dependencies between variables, distinguishing between instantaneous effects (occurring within the same time step) and lagged effects (propagating over time). The model naturally associates time-series data with a directed acyclic graph (DAG), which encodes how each time step is generated from previous ones. These relationships collectively form the window graph, a DAG that uniquely determines the SVAR parameters. SVAR further imposes causal stationarity, meaning that the dependencies remain constant over time [Assaad et al., 2022b].

Challenges and limitations Even when the goal is limited to learning the DAG structure—without the added task of identifying causally relevant edges, under the pure causality notion [Pearl, 2009]—inferring DAGs from time-series data remains computationally challenging. This difficulty arises from the complex temporal dependencies and the high dimensionality typical of real-world datasets. From a theoretical viewpoint, the problem generalizes DAG learning from static data, which is already known to be NP-hard [Chickering et al., 2004]. Several methods have been proposed to estimate the weighted window graph from time-series data, including approaches specifically tailored for SVAR [Hyvärinen et al., 2010]. However, many existing methods suffer from critical limitations. Some approaches, such as Granger causality-based methods, learn the sum-

mary graph that fails to incorporate time lags [Bussmann et al., 2021], while others do not account for instantaneous dependencies [Khanna and Tan, 2019]. Most methods face computational challenges when applied to large DAGs, making them impractical for graphs with thousands of nodes [Cheng et al., 2024]. Structural shocks, i.e., the input variables of an SVAR [Lanne et al., 2017a] at each node, are often interpreted merely as noise variables in prior work [Hyvärinen et al., 2010, Pamfil et al., 2020], limiting their interpretability and potential insights into the underlying causal mechanisms. To address these challenges, we introduce a novel, computationally efficient method that enforces sparsity in the input of the SVAR.

SpinSVAR: Sparse input SVAR Hyvärinen et al. [2010] model SVAR under a non-Gaussian noise assumption for the inputs. We extend it by additionally enforcing sparsity in the input, following Misiakos et al. [2023], and model it as independent and identically distributed (i.i.d.) Laplacian random variables. The Laplace distribution naturally promotes sparsity [Jing et al., 2015] due to its sharp peak at zero and heavy tails. Intuitively, this means that a few significant independent events drive the observed data through the SVAR structure. This contrasts with prior work, which typically assumes zero-mean Gaussian input, either explicitly [Lachapelle et al., 2019] or implicitly via mean square error-based optimization objectives [Pamfil et al., 2020, Sun et al., 2023, Tank et al., 2021]. By incorporating this Laplacian input model, we derive a maximum likelihood estimator (MLE) based on least absolute error regression, leading to SpinSVAR, a new method for efficient SVAR estimation from time-series data. This framework provides both theoretical and empirical advantages.

Contributions Our main contributions are:

- We model sparse SVAR input as independent zero-mean Laplacian variables, yielding an MLE formulation for estimating SVAR parameters.
- We prove the consistency of this MLE under mild assumptions on the window graph weights.
- We introduce SpinSVAR, a regularized MLE framework enabling fast, and accurate SVAR estimation from time-series data.
- In synthetic experiments with sparse SVAR input, generated via Laplacian or Bernoulli-uniform distribution as in [Misiakos et al., 2023], SpinSVAR can learn an associated DAG with up to several thousands of nodes and outperforms various state-of-the-art methods.
- On real-world financial data from the S&P 500 index, we show that the sparse input assumption clusters stocks by sector and identifies structural shocks reflecting significant changes in the stock prices due to unexpected news.

For completeness, we provide detailed proofs, algorithmic explanations, and additional experiments in the supplementary material; however, these are not required for understanding the main paper and are intended for readers seeking deeper insights.

2 SVAR WITH SPARSE INPUT

We introduce notation, the needed background on SVARs, the motivation for a sparsity assumption in the input and its statistical modeling using the Laplace distribution.

Time-series data A multi-dimensional data vector $\boldsymbol{x}_t$, measured at time point $t \in 0, 1, \ldots, T-1 = [T]$, is written as $\boldsymbol{x}_t = (x_{t,1}, x_{t,2}, \ldots, x_{t,d}) \in \mathbb{R}^{1\times d}$. A time series consists of a sequence of such data vectors $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{T-1}$ recorded at consecutive time points. We assume these vectors are stacked as rows in a matrix, representing the entire time series, denoted as $\boldsymbol{X} \in \mathbb{R}^{T\times d}$. When multiple realizations of $\boldsymbol{X}$ are available, they are collected as slices of a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{N\times T\times d}$. These can obtained by dividing a long time series into smaller segments of length T[1].

Example: stock market We consider an example of time-series data from the stock market. We collect daily stock values $\boldsymbol{x}_t$ for a particular stock index (e.g., S&P 500) for, say, 20 years. A time series for one year is denoted with the matrix $\boldsymbol{X}$ and 20 years yield the data tensor $\boldsymbol{\mathcal{X}}$.

Model demonstration We impose a graph-based model on the generation of time-series data and first illustrate it with a simple example. Suppose that the vector $\boldsymbol{x}_t$ at time t is generated from the previous time step's data $\boldsymbol{x}_{t-1}$ according to the equation:

$$\boldsymbol{x}_t = \boldsymbol{x}_{t-1}\boldsymbol{B} + \boldsymbol{s}_t, \tag{1}$$

where $\boldsymbol{s}_t$ represents the input variables, commonly referred to as structural shocks [Kilian, 2013], but they have also been described as root causes [Misiakos et al., 2023]. Given $\boldsymbol{s}_t$ for all t, the data $\boldsymbol{x}_t$ is fully determined by the matrix $\boldsymbol{B}$ through (1). The model in (1) is an instance of vector autoregression (VAR) [Kilian, 2013]. The (i, j) entry of the matrix $\boldsymbol{B} \in \mathbb{R}^{d\times d}$ quantifies the influence of $x_{t-1,i}$ on $x_{t,j}$. This corresponds to the adjacency matrix of a directed graph $\mathcal{G} = (\mathbb{V}, \boldsymbol{B})$, where $\mathbb{V}$ is the set of nodes enumerated as $\mathbb{V} = 1, 2, ..., d$. The primary objective is to learn $\boldsymbol{B}$ from time-series data $\{\boldsymbol{x}_t\}_{t\in[T]}$. The model in (1) is causally stationary, meaning that $\boldsymbol{B}$ remains constant across all time steps. Additionally, it has a time lag of one, as each observation $\boldsymbol{x}_t$ depends only on the previous time step $\boldsymbol{x}_{t-1}$ and the newly introduced inputs $\boldsymbol{s}_t$ at time t.

[1] In practical scenarios this can affect independence between time-series samples, which is a necessary assumption for our theoretical guarantees.

Example In the stock market example, the stocks $1, 2, \ldots, 500$ in the S&P 500 market index would represent the nodes of a graph and $\boldsymbol{B}$ would encode the influences between these stocks. The model then would imply that the value $x_{t,i}$ of stock i on day t is determined by the stock values $\boldsymbol{x}_{t-1}$ from day $t-1$, combined with a structural shock $s_{t,i}$ representing an event occurring on day t.

Structural vector autoregression An SVAR [Lütkepohl, 2005, Pamfil et al., 2020] expands the VAR in (1) to the general form with maximal time lag k. Namely, we assume there exist adjacency matrices $\boldsymbol{B}_0, \boldsymbol{B}_1, ..., \boldsymbol{B}_k \in \mathbb{R}^{d\times d}$ and $\boldsymbol{s}_t \in \mathbb{R}^{1\times d}$, such that $\boldsymbol{x}_t = \boldsymbol{0}$ for $t < 0$ and for $t \in [T]$ [2] :

$$\boldsymbol{x}_t = \boldsymbol{x}_t\boldsymbol{B}_0 + \boldsymbol{x}_{t-1}\boldsymbol{B}_1 + ... + \boldsymbol{x}_{t-k}\boldsymbol{B}_k + \boldsymbol{s}_t. \quad (2)$$

The (i, j) entry of $\boldsymbol{B}_\tau$ represents the influence of i to j after τ time steps (i.e., a lag of τ) and $\boldsymbol{s}_t$ are the structural shocks. $\boldsymbol{B}_0$ represents *instantaneous* dependencies, while the $\boldsymbol{B}_1, ..., \boldsymbol{B}_k$ represent *lagged* dependencies. The SVAR is *causally stationary*, since the $\boldsymbol{B}_\tau$ do not depend on t. Following Pamfil et al. [2020] we assume that $\boldsymbol{B}_0$ corresponds to a DAG, ensuring that the recurrence (2) is solvable for $\boldsymbol{x}_t$.

The instantaneous $\boldsymbol{B}_0$ and lagged $\boldsymbol{B}_1, ..., \boldsymbol{B}_k$ dependencies are collected as block-rows in a matrix $\boldsymbol{W} \in \mathbb{R}^{d(k+1)\times d}$ which forms the so-called window graph depicted with an example in Fig. 1. Note that the window graph is a DAG since the edges go only forward in time. The problem we aim to solve is to infer $\boldsymbol{W}$ from time-series data under the assumption that there are few significant structural shocks. To achieve this, our approach imposes a sparsity assumption on the input $\boldsymbol{s}_t$.

Example In the previous stock market example, the matrix $\boldsymbol{B}_0$ represents instantaneous influences within the same day, while the other matrices $\boldsymbol{B}_\tau$ capture influences across different days. Since stock markets typically react almost instantaneously to new information, one would expect most dependencies to be reflected in $\boldsymbol{B}_0$.

Sparse input We denote with $\boldsymbol{x}_{t,\text{past}} = (\boldsymbol{x}_t, ..., \boldsymbol{x}_{t-k})$, $t \in [T]$, the data at previous time steps of $\boldsymbol{x}_t$ with lag up to a chosen fixed k. Analogously, $\boldsymbol{X}_{\text{past}} \in \mathbb{R}^{T\times d(k+1)}$ contains as rows the vectors $\boldsymbol{x}_{t,\text{past}}$, $t \in [T]$ and $\boldsymbol{\mathcal{X}}_{\text{past}} \in \mathbb{R}^{N\times T\times d}$ contains N realizations of $\boldsymbol{X}_{\text{past}}$. With this notation, the SVAR (2) can be written in the following matrix format:

$$\boldsymbol{X} = \boldsymbol{X}_{\text{past}}\boldsymbol{W} + \boldsymbol{S} \Rightarrow \boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{X}}_{\text{past}}\boldsymbol{W} + \boldsymbol{\mathcal{S}}. \quad (3)$$

Intuitively, the non-zero values in $\boldsymbol{\mathcal{S}}$ represent unobserved events that propagate through space (according to $\boldsymbol{B}_0$) and also through time t (according to $\boldsymbol{B}_1, ..., \boldsymbol{B}_k$) to generate $\boldsymbol{X}$ via (3). In Fig. 1 we illustrate the data generation process (3). In the upper part, the significant structural shocks $\boldsymbol{\mathcal{S}}$ are denoted in color, whereas white nodes correspond to (approximately) zero values (noise).

[2]We provide a stability condition for (2) in App. A.1.

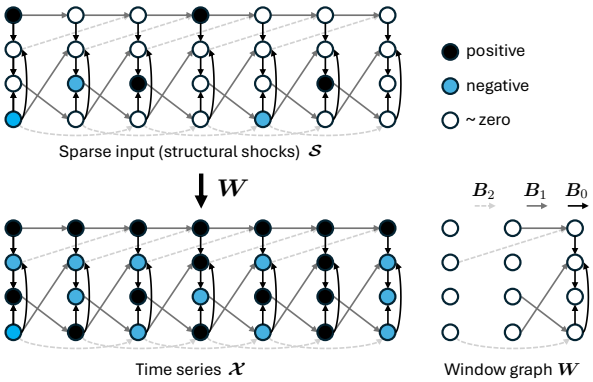


Figure 1: Visualizing an SVAR (3) with sparse input $\boldsymbol{\mathcal{S}}$. Out of 28 structural shocks in $\boldsymbol{\mathcal{S}}$ only seven are significant (positive or negative) and the rest are approximately zero. The window graph $\boldsymbol{W}$, composed of $\boldsymbol{B}_0, \boldsymbol{B}_1, \boldsymbol{B}_2$, generates the observed dense time series $\boldsymbol{\mathcal{X}}$ (bottom) via (3).

Example In our stock market example, the structural shocks $\boldsymbol{s}_t$ would represent significant events (big news) that trigger changes in the prices of the stocks at day t. Examples include unexpected quarterly results, administrative changes in the company, capital investment, launching a new product, etc. It is intuitive that significant events happen rarely and affect few stocks every day, and thus $\boldsymbol{\mathcal{S}}$ is sparse. Later, we confirm the sparse input assumption in experiments with real-world financial time series.

Laplace distribution In practical applications, input sparsity can only be approximately satisfied. Therefore, we consider a distribution for $\boldsymbol{\mathcal{S}}$ that encourages sparsity formation. A natural choice is the Laplace$(0, \beta)$ distribution, which is characterized by a sharp peak at 0 and heavy tails [Jing et al., 2015]. Tibshirani [1996] introduced the classical LASSO regression by adopting the Laplace prior, leading to the well-known L^1 regularizer that promotes sparsity. The Laplace prior has also been used in Bayesian linear regression [Castillo et al., 2015], compressive sensing [Babacan et al., 2009], sparse matrix factorization [Jing et al., 2015], and sparse principal component analysis (PCA) [Guan and Dy, 2009]. Based on this, we impose the following assumption on $\boldsymbol{S}$ and derive its probability density function f_S:

$$\boldsymbol{S}_{t,j} \sim \text{Laplace}(0, \beta) \Leftrightarrow f_S(\boldsymbol{S}_{t,j}|\beta) = \frac{1}{2\beta}e^{-\frac{|\boldsymbol{s}_{t,j}|}{\beta}}, \quad (4)$$

where the ground truth β parameter is denoted with β^*. With (4) we impose the assumption that the input terms $\boldsymbol{S}_{t,j}$ are i.i.d., which implies that there are no hidden confounders in the data. Furthermore, we assume $\boldsymbol{\mathcal{X}}$ contains N i.i.d. realizations of $\boldsymbol{X}$ via (3).

3 LEARNING THE SVAR

In this section, we establish the identifiability of our setting, derive the Laplacian MLE, prove its consistency, and formulate the proposed optimization framework, SpinSVAR.

Identifiability A fundamental question in causal discovery is whether the graph structure is identifiable from the data [Park, 2020]. Let $\boldsymbol{W}^*$ be the ground-truth window graph, and $f_X(\boldsymbol{\mathcal{X}}|\boldsymbol{W}, \beta)$ denote the probability density function of the data, parameterized by $(\boldsymbol{W}, \beta)$. Identifiability means that if $f_X(\boldsymbol{\mathcal{X}}|\boldsymbol{W}, \beta) = f_X(\boldsymbol{\mathcal{X}}|\boldsymbol{W}^*, \beta^*)$, then necessarily $\boldsymbol{W} = \boldsymbol{W}^*$. This ensures that the window graph $\boldsymbol{W}^*$ is uniquely determined by the data distribution. Theorem 3.1 establishes the identifiability of $\boldsymbol{W}$ and the parameter β, which is a necessary condition for our consistency result. Note that, the identification result for the window graph $\mathbf{W}$ can be directly derived from VARLiNGAM [Hyvärinen et al., 2010], even though our proof is slightly different. Moreover, the proof of the identifiability of the β parameter of the Laplacian distribution is new and specific to our setting.

Theorem 3.1. *Consider the time-series model (3) with $\boldsymbol{S}$ following a multivariate Laplace distribution (4) with $\beta^* > 1/NTd$. Then the adjacency matrices $\boldsymbol{B}_0, \boldsymbol{B}_1, ..., \boldsymbol{B}_k \in \mathbb{R}^{d\times d}$ and β are identifiable from the time-series data $\boldsymbol{\mathcal{X}}$.*

Proof sketch. We unroll $\boldsymbol{W}$ over time into a DAG and rewrite (3) as a linear SEM, as explained in [Misiakos et al., 2024]. The identifiability then follows from LiNGAM [Shimizu et al., 2006], since $\boldsymbol{\mathcal{S}}$ follows a Laplacian distribution. The window graph is identified by extracting $\boldsymbol{B}_0, \boldsymbol{B}_1, \ldots, \boldsymbol{B}_k$ from the unrolled DAG. The parameter β is identified using the monotonicity of the probability density function. A complete proof is provided in App. A.2. □

Laplacian MLE The MLE is a fundamental statistical method for estimating model parameters by maximizing the likelihood function $f_X(\boldsymbol{\mathcal{X}}|\boldsymbol{W}, \beta)$ given the observed data $\boldsymbol{\mathcal{X}}$. Under the Laplacian noise model (4), the probability density function of $\boldsymbol{X}$ is given by (see App. A.3 for details):

$$f_X(\boldsymbol{\mathcal{X}}|\boldsymbol{W}, \beta) = \frac{|\det(\boldsymbol{I} - \boldsymbol{B}_0)|^{NT}}{(2\beta)^{NdT}} e^{-\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}_{\text{past}}\boldsymbol{W}\|_1/\beta}. \tag{5}$$

The MLE seeks to find the optimal parameters by maximizing the likelihood function. Equivalently, we maximize the log-likelihood function $\mathcal{L}(\boldsymbol{W}, \beta; \boldsymbol{\mathcal{X}}) = \log f_X(\boldsymbol{\mathcal{X}}|\boldsymbol{W}, \beta)$:

$$\mathcal{L}(\boldsymbol{W}, \beta; \boldsymbol{\mathcal{X}}) = NT \log|\det(\boldsymbol{I} - \boldsymbol{B}_0)| - NTd\log(2\beta) - \frac{1}{\beta}\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}_{\text{past}}\boldsymbol{W}\|_1. \tag{6}$$

Thus, the MLE estimate $\widehat{\boldsymbol{W}}$ is given by:

$$\widehat{\boldsymbol{W}} = \underset{\boldsymbol{W}\in\mathcal{W}}{\arg\max}\, \mathcal{L}(\boldsymbol{W}, \beta; \boldsymbol{\mathcal{X}}). \tag{7}$$

A desirable property of the MLE is that $\widehat{\boldsymbol{W}} \to \boldsymbol{W}^*$ as $N \to \infty$. In the infinite sample regime, the log-likelihood function $\mathcal{L}(\boldsymbol{W}, \beta; \boldsymbol{\mathcal{X}})$ corresponds to the population log-likelihood defined as:

$$L(\boldsymbol{W}, \beta) = \mathbb{E}_{\boldsymbol{W}^*, \beta^*}[\mathcal{L}(\boldsymbol{W}, \beta; \boldsymbol{\mathcal{X}})], \tag{8}$$

where $L(\boldsymbol{W}, \beta)$ represents the expected value of $\mathcal{L}(\boldsymbol{W}, \beta; \boldsymbol{\mathcal{X}})$ computed under the ground truth probability density $f_X(\boldsymbol{\mathcal{X}}|\boldsymbol{W}^*, \beta^*)$. Under the assumption of identifiability, the maximizer $\widehat{\boldsymbol{W}}$ of $L(\boldsymbol{W}, \beta)$ satisfies $\widehat{\boldsymbol{W}} = \boldsymbol{W}^*$. The following lemma formalizes this property, with a proof provided in App. A.4.

Lemma 3.2. *Assume that the ground truth parameters $(\boldsymbol{W}^*, \beta^*)$ are identifiable from the data distribution $f_X(\boldsymbol{X}|\boldsymbol{W}^*, \beta^*)$. Then, the population likelihood $L(\boldsymbol{W}, \beta)$ has a unique maximum at $(\boldsymbol{W}^*, \beta^*)$.*

Lemma 3.2 implies that with infinite data, the log-likelihood has a unique global maximizer at the ground truth $\boldsymbol{W}^*$. However, since we only have a finite dataset, we require a stronger result for the empirical log-likelihood $\mathcal{L}(\boldsymbol{W}, \beta; \boldsymbol{\mathcal{X}})$.

Consistency of MLE We prove the consistency of the MLE, which states that as the amount of data increases, $\widehat{\boldsymbol{W}}$ converges in probability to $\boldsymbol{W}^*$. Formally, we show the following result.

Theorem 3.3. *The maximum log-likelihood estimator (7) satisfies the conditions of Theorem 2.5 of Newey and McFadden [1994] and thus is consistent under the following assumptions:*

- *The space of window graphs is $\mathcal{W} \subseteq [-1, 1]^{d(k+1)\times d}$ and $\boldsymbol{B}_0$ acyclic.*
- *$\beta \in [a, b]$ is bounded, with a lower bound $a > 1/NTd$.*
- *The N time-series samples $\boldsymbol{\mathcal{X}}_i$ are i.i.d..*

Proof sketch. The proof requires a compact search space for $\boldsymbol{W}$, which is satisfied by the given bounds on $\boldsymbol{W}$ and β. Additionally, the set of acyclic matrices is closed, as it can be expressed as the pre-image $h^{-1}(\{0\})$, where h is a continuous function characterizing acyclicity [Zheng et al., 2018]. Identifiability of $\boldsymbol{W}$ and β is ensured by Theorem 3.1. Finally, the log-likelihood is continuous, and it can be shown that $\sup_{\boldsymbol{W}\in\mathcal{W}}|\mathcal{L}(\boldsymbol{W}; \boldsymbol{\mathcal{X}})|$ has finite expectation. Under these requirements, Theorem 2.5 of Newey and McFadden [1994] then utilizes the uniform law of large numbers to show that $\widehat{\boldsymbol{W}}$ converges in probability to $\boldsymbol{W}^*$. A complete proof is provided in App. A.5. □

Our method SpinSVAR Theorem 3.3 implies that $\widehat{\boldsymbol{W}}$ in (7) converges in probability to $\boldsymbol{W}^*$ as $N \to \infty$. Since the parameter β is fixed but unknown, we estimate it by maximizing the log-likelihood function (6). Following Ng et al. [2020], we compute an estimate $\widehat{\beta}$ by solving:

$$\frac{\partial \mathcal{L}}{\partial \beta} = 0 \Leftrightarrow \widehat{\beta} = \frac{1}{NTd} \left\| \boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}_{\text{past}} \boldsymbol{W} \right\|_1 . \tag{9}$$

This estimate is consistent in expectation. Indeed, it is true that $\mathbb{E}\left[\left\| \boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}_{\text{past}} \boldsymbol{W} \right\|_1\right] = \mathbb{E}\left[\left\| \boldsymbol{\mathcal{S}} \right\|_1\right] = NTd\beta^*$. Thus, the log-likelihood maximization problem for approximating $\boldsymbol{W}$ reduces to (see App. A.6 for details):

$$\begin{aligned} \widehat{\boldsymbol{W}} &= \underset{\boldsymbol{W} \in \mathcal{W}}{\arg\max}\, \mathcal{L}\left(\boldsymbol{W}, \widehat{\beta}; \boldsymbol{\mathcal{X}}\right) \\ &= \underset{\boldsymbol{W} \in \mathcal{W}}{\arg\min} \log \left\| \boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}_{\text{past}} \boldsymbol{W} \right\|_1 - \frac{1}{d} \log \left| \det \left(\boldsymbol{I} - \boldsymbol{B}_0 \right) \right| . \end{aligned} \tag{10}$$

However, directly minimizing (10) over the space of DAGs is computationally inefficient. This would require enforcing a hard DAG constraint to restrict $\boldsymbol{W} \in \mathcal{W}$, as in [Zheng et al., 2018], where it is implemented via the augmented Lagrangian method. Such an approach demands careful fine-tuning and can lead to numerical instabilities, as demonstrated by Ng et al. [2020]. To overcome these challenges, following Ng et al. [2020], we relax the hard acyclicity constraint and introduce a soft regularizer. This approach maintains strong performance while improving efficiency, as demonstrated in our experiments. The final optimization problem for SpinSVAR is formulated as:

$$\begin{aligned} \widetilde{\boldsymbol{W}} = \underset{\boldsymbol{W} \in \mathbb{R}^{d(k+1) \times d}}{\arg\min} & \log \left\| \boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}_{\text{past}} \boldsymbol{W} \right\|_1 \\ & - \frac{1}{d} \log \left| \det \left(\boldsymbol{I} - \boldsymbol{B}_0 \right) \right| + \lambda_1 \|\boldsymbol{W}\|_1 + \lambda_2 \cdot h\left(\boldsymbol{B}_0\right) . \end{aligned} \tag{11}$$

The first term in (11) promotes sparsity in the structural shocks, while the remaining terms encourage sparsity in the window graph $\boldsymbol{W}$ and enforce acyclicity in $\boldsymbol{B}_0$, respectively. The acyclicity regularizer $h\left(\boldsymbol{B}_0\right) = e^{\boldsymbol{A} \odot \boldsymbol{A}} - d$, introduced by Zheng et al. [2018], ensures that $\boldsymbol{B}_0$ satisfies the DAG constraint. Notably, (11) is well-suited for GPU acceleration using tensor operations, making it highly efficient in practice. In our implementation, we represent $\boldsymbol{W}$ as the parameter matrix of a (PyTorch) linear layer with $(k+1)d$ inputs and d outputs. The precomputed $\boldsymbol{\mathcal{X}}_{\text{past}}$ serves as input, and the linear layer's output is subtracted from the observed data $\boldsymbol{\mathcal{X}}$. The objective in (11) is then computed and optimized using the Adam optimizer [Kingma and Ba, 2014]. More implementation details can be found in App. C.

Since the proposed objective function is non-convex, it may have multiple local optima, and there is no guarantee of convergence to the global maximum. However, in practice, our method performs well and often even recovers the edges of $\boldsymbol{W}^*$ without error. This phenomenon, also observed in GOLEM [Ng et al., 2020], motivates further theoretical investigation.

Once $\widehat{\boldsymbol{W}}$ is obtained via (11), we approximate the input $\widehat{\boldsymbol{\mathcal{S}}}$:

$$\widehat{\boldsymbol{\mathcal{S}}} = \boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}_{\text{past}} \widehat{\boldsymbol{W}} . \tag{12}$$

In recovering $\boldsymbol{\mathcal{S}}$ from $\widehat{\boldsymbol{\mathcal{S}}}$, we are particularly interested in identifying significant structural shocks. To this end, we apply thresholding to filter out insignificant values in $\widehat{\boldsymbol{\mathcal{S}}}$. In our experiments, this threshold is selected based on the synthetic data generation process.

4 RELATED WORK

Time-series causal discovery Our work falls within the category of continuous optimization methods but differs in its assumption of sparsity in the input of the SVAR. Closely related approaches include functional causal model-based methods such as VARLiNGAM [Hyvärinen et al., 2010], which estimates an SVAR, as well as TiMINO [Peters et al., 2013] and NBCB [Assaad et al., 2021], which recover only the summary graph that disregards time delays [Gong et al., 2023]. In contrast, our method learns the full window graph. Other continuous optimization methods include DYNOTEARS [Pamfil et al., 2020], NTS-NOTEARS [Sun et al., 2023] for non-linear data, and iDYNO [Gao et al., 2022] for interventional data. These methods optimize the mean square error loss and do not impose sparsity on the SVAR input. In our experiments, we compare against these methods, as well as others that learn the window graph from observational time-series data, selecting both methodologically relevant approaches and representative alternatives.

Different from our approach, constraint-based methods infer edges using conditional independence tests. Examples include PCMCI [Runge et al., 2019], tsFCI [Entner and Hoyer, 2010], PCMCI+[Runge, 2020], LPCMCI [Gerhardus and Runge, 2020], PC-GCE [Assaad et al., 2022a], and SVAR-FCI [Malinsky and Spirtes, 2018]. Methods based on Granger causality typically recover only the summary graph. Notable examples include neural Granger causality [Tank et al., 2021], eSRU [Khanna and Tan, 2019], GVAR [Marcinkevičs and Vogt, 2020], and convergent cross mapping [Sugihara et al., 2012]. Another line of work leveraging neural networks includes TCDF [Nauta et al., 2019], SCGL [Xu et al., 2019], neural graphical modeling [Bellot et al., 2022], and amortized learning [Löwe et al., 2022].

Maximum likelihood estimator By modeling sparsity with a Laplacian distribution, we derive an MLE objective based on least absolute error loss, unlike prior causal discovery methods [Ng et al., 2020, Pamfil et al., 2020, Nauta et al., 2019], which use mean-square loss suited for Gaussian noise. Peters and Bühlmann [2014] provide consistency guarantees of the MLE for a linear SEM with equivariant Gaussian errors and GranDAG [Lachapelle et al., 2019] applies it to nonlinear additive noise models. However, these methods neither support time-series data nor enforce input

sparsity. For SVAR estimation, Hyvärinen et al. [2010] propose a generic MLE approach for non-Gaussian noise but do not explicitly integrate it into their methodology. Other MLE-based methods for SVAR [Lanne et al., 2017a, Fiorentini and Sentana, 2023, Maekawa and Nakanishi, 2023] also remain generic and are not tailored to Laplacian or sparse inputs. In particular, Lanne et al. [2017b] establish MLE consistency under a different set of assumptions: their model allows for a potentially cyclic $\boldsymbol{B}_0$ and does not incorporate regularization for sparsity or acyclicity. In contrast, our approach explicitly enforces acyclicity to ensure identifiability of the SVAR parameters—a key requirement for our consistency result. Therefore, our theoretical analysis is distinct, specifically designed for linear SVAR models with Laplacian-distributed inputs.

Least absolute error and sparsity The least absolute error (LAE) loss arises as an MLE when assuming that the SVAR input follows a Laplacian distribution Chai et al. [2019], Li and Arce [2004], enforcing sparsity in the model. LAE has been widely used as a regression objective across various fields, including dynamical systems [Jiang et al., 2023, He and Sun, 2024], due to its robustness against outliers compared to mean square error (MSE) loss [Pollard, 1991, Bassett Jr and Koenker, 1978, Kumar and Singh, 2015, Narula et al., 1999]. Despite this, the only method that employs LAE regression to enforce sparsity in the input of a linear SEM is SparseRC, proposed by Misiakos et al. [2023]. Misiakos et al. [2024] extended SparseRC to time-series graph learning by unrolling the window graph into a DAG, requiring the estimation of $(dT)^2$ parameters—rendering it computationally infeasible for our experiments. Our method advances over SparseRC by formulating a Laplacian MLE to enforce sparse input, providing both consistency guarantees and improved computational efficiency in practice.

5 EXPERIMENTS

We compare SpinSVAR to prior state-of-the-art work on learning the window graph $\boldsymbol{W}$ from time-series data. Our experiments in this section cover synthetic and real data. Additional experiments are in Appendix E.

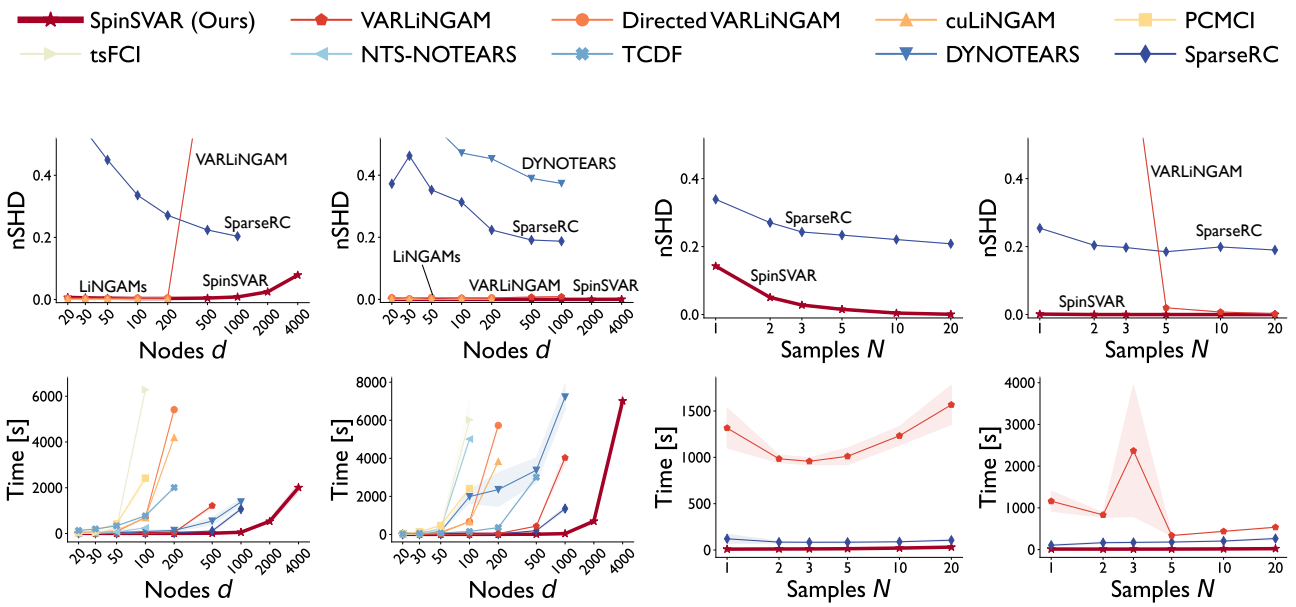
Baselines We compare against functional causal model methods VARLiNGAM, Directed VARLiNGAM [Hyvärinen et al., 2010], and the GPU-accelerated cuLiNGAM [Akinwande and Kolter, 2024], continuous optimization methods DYNOTEARS [Pamfil et al., 2020] and SparseRC [Misiakos et al., 2023], non-linear approaches NTS-NOTEARS [Sun et al., 2023] and TCDF [Nauta et al., 2019] and constraint-based methods tsFCI [Entner and Hoyer, 2010] and PCMCI [Runge et al., 2019]. Among these, LiNGAM-based methods assume non-Gaussian SVAR input, which yields the most competitive performance but at the cost of higher computational complexity. SparseRC enforces input sparsity but times out; thus, we modify its setup to a smaller unrolled DAG (details in App. B). The other baselines do not enforce input sparsity. We compare the optimization objective and computational complexity of the baselines and SpinSVAR in App. C.1. For the implementations we use public repositories (App. E.12), with hyperparameters tuned via grid search (App. E.10).

Metrics We evaluate the unweighted approximation of $\boldsymbol{W}$ using the normalized Structural Hamming Distance (nSHD). The Structural Hamming Distance (SHD), reported in Appendix E.2, counts the number of edge insertions, deletions, and reversals required to transform the estimated graph into the ground truth. The nSHD is then obtained via normalization: dividing the SHD by the total number of edges in the ground truth $\boldsymbol{W}$. An nSHD above 0.5 is considered a failure and is therefore not reported. The structural intervention distance (SID) [Peters and Bühlmann, 2015] is omitted as it times-out for DAGs with thousands of nodes. Additional results in App. E.2 include precision (PREC), recall (REC), area under ROC curve (AUROC), F1 score, and normalized MSE (NMSE) for the weighted approximation of $\boldsymbol{W}$. We also assess the detection of significant input values $\boldsymbol{\mathcal{S}}$ using SHD and NMSE for $\widehat{\boldsymbol{\mathcal{S}}}$. For all metrics, we report the mean and standard deviation (shown as shade) in Fig. 2 over five experiment repetitions. In the real-world stock market dataset, where the ground truth is unknown, the evaluation is purely empirical.

5.1 SYNTHETIC EXPERIMENTS

Data generation We generate data using the SVAR model (3), following settings similar to [Pamfil et al., 2020] for the SVAR window graph $\boldsymbol{W}$ and to [Misiakos et al., 2023] for the sparse SVAR input $\boldsymbol{\mathcal{S}}$. First, we set the number of nodes d, the length T of the time series, the number of realizations N, and the maximum lag k of the SVAR (2). For the window graph $\boldsymbol{W}$, we generate directed random Erdös-Renyi graphs for $\boldsymbol{B}_0, \boldsymbol{B}_1, \ldots, \boldsymbol{B}_k$, where $\boldsymbol{B}_0$ is a DAG with an average degree of 5, and $\boldsymbol{B}_1, \ldots, \boldsymbol{B}_k$ have an average degree of 2. We consider a default time lag of $k = 2$ and include an additional version with $k = 5$ in App. E.4. The edges of $\boldsymbol{W}$ are assigned uniform random weights from $[-0.5, -0.1] \cup [0.1, 0.5]$. The upper bound of 0.5 ensures that (3) is stable, and the generated data $\boldsymbol{\mathcal{X}}$ remain bounded in most cases (we discard $\boldsymbol{\mathcal{X}}$ if its entries become excessively large; see App. E.1 for details).

To impose sparsity in $\boldsymbol{\mathcal{S}}$, we consider two scenarios, using a threshold of 0.1 to distinguish significant values from approximately zero values in $\boldsymbol{\mathcal{S}}$. First, we use the Laplacian distribution (4) with $\beta = \frac{1}{3}$, where in expectation only 5% of values are significant (magnitudes greater than 0.1; see App. D). Second, we use a Bernoulli distribution to control the percentage of significant entries in $\boldsymbol{\mathcal{S}}$ [Kalisch and



(a) $N = 10$, Laplace (b) $N = 10$, Bernoulli (c) $d = 500$, Laplace (d) $d = 500$, Bernoulli

Figure 2: Synthetic experiments. The first row shows nSHD (lower is better), the second row runtime. (a), (b) consider $N = 10$ samples of time-series with $T = 1000$ and varying number d of nodes for both input distributions. (c), (d) consider $d = 500$ nodes and varying number of samples N of time-series of length $T = 1000$. The label LiNGAMs refers to VARLiNGAM and its two variations Directed VARLiNGAM and cuLiNGAM. Any non-reported point implies a time-out (execution time $> 10,000$s $\approx$ 2:45h).

Bühlman, 2007, Misiakos et al., 2023]: each entry is non-zero with probability $p = 5\%$ (assigned uniform weights from $[-1, -0.1] \cup [0.1, 1]$) or zero otherwise. To create approximate sparsity, we add zero-mean Gaussian noise with a standard deviation of 0.01 to $\boldsymbol{\mathcal{S}}$. We refer to this distribution as Bernoulli-uniform or simply Bernoulli. In Appendix E.3, we examine a third sparsity scenario, referred to as Gaussian subsampling, in which the non-zero entries of the Bernoulli variables are assigned values drawn from a normal distribution rather than uniform ones.

Results Fig. 2 presents the results of our synthetic experiments for both sparsity scenarios of $\boldsymbol{\mathcal{S}}$ (Laplace and Bernoulli). Figs. 2a, 2b correspond to a fixed number of samples, $N = 10$, with the number of nodes d ranging from 20 (180 edges) to 4000 (36,000 edges). In Figs. 2c, 2d, we fix $d = 500$ and vary the samples N from 1 to 20. In all cases, the time-series length is $T = 1000$. Baselines that are omitted either perform worse or time out.

In Figs. 2a,2b, SpinSVAR achieves the best performance, recovering $\boldsymbol{W}$ nearly perfectly for Bernoulli inputs and handling up to 2000 nodes for Laplacian inputs while maintaining the best runtime. Its computational complexity is superior to SparseRC, VARLiNGAM, and its variants, and comparable to DYNOTEARS (see App. C.1 for details). This efficiency stems from leveraging the sparse input assumption, enabling faster convergence with fewer iterations. Baseline methods such as PCMCI, tsFCI, TCDF, NTS-NOTEARS, and DYNOTEARS perform poorly even on small graphs. The latter three rely on MSE loss, which is better suited for Gaussian inputs. SparseRC, which enforces sparsity via an LAE loss, exhibits slight improvements but struggles with larger graphs, timing out beyond 1000 nodes. The strongest baseline methods are VARLiNGAM and its variants (jointly labelled as LiNGAMs in Fig. 2). VARLiNGAM scales better but performs slightly worse, running up to 1000 nodes before timing out. The other LiNGAMs, namely Directed VARLiNGAM and cuLiNGAM, also yield strong results but already time out beyond 200 nodes. For large graphs with Bernoulli input (Fig. 2b), VARLiNGAM remains competitive but is approximately 100 times slower for $d = 1000$.

For varying N (Figs. 2c, 2d), SpinSVAR consistently excels in the Bernoulli case and improves as N increases in the Laplace case. SparseRC performs poorly in both setups. VARLiNGAM struggles with Laplacian input and requires more samples than SpinSVAR in the Bernoulli setup. Additional results for varying d at fixed $N = 1$ in App. E.2 confirm these trends.

Note that all baselines except DYNOTEARS are designed for single time series input. For these methods, we concate-

Table 1: Normalized SHD for large DAGs ($T = 1000$).

SpinSVAR $N =$	1	2	4	8	16
$d = 1000, \boldsymbol{\mathcal{S}} \sim$ Laplace	0.927	0.118	0.041	0.012	**0.003**
$d = 1000, \boldsymbol{\mathcal{S}} \sim$ Bernoulli	0.000	**0.000**	**0.000**	**0.000**	**0.000**
$d = 2000, \boldsymbol{\mathcal{S}} \sim$ Laplace	1.000	0.958	0.116	0.036	0.010
$d = 2000, \boldsymbol{\mathcal{S}} \sim$ Bernoulli	0.001	**0.000**	**0.000**	**0.000**	**0.000**
$d = 4000, \boldsymbol{\mathcal{S}} \sim$ Laplace	1.010	0.995	0.908	0.125	0.034
$d = 4000, \boldsymbol{\mathcal{S}} \sim$ Bernoulli	0.005	0.001	0.000	**0.000**	**0.000**
VARLiNGAM $N =$	1	2	4	8	16
$d = 1000, \boldsymbol{\mathcal{S}} \sim$ Laplace	–	–	–	–	–
$d = 1000, \boldsymbol{\mathcal{S}} \sim$ Bernoulli	–	–	–	0.013	0.003

nate the N samples into one long sequence. We acknowledge that this preprocessing step may affect their performance—particularly for methods like VARLiNGAM. In contrast, our method is explicitly designed to handle multiple time series jointly, leveraging the assumption that we observe N i.i.d. time series samples. To enable a fairer comparison, we include additional experiments (App.E.2) where $N = 1$ and $T = 10000$, matching the total number of observations in Figs.2a and 2b, as well as experiments with $N = 1$ and $T = 1000$. These results further support our current conclusions.

Larger graphs In Table 1 we evaluate VARLiNGAM and SpinSVAR on graphs with up to $d = 4000$ nodes, varying the number of samples. These were the only methods that maintained reasonable performance without timing out at $d = 1000$. VARLiNGAM struggles with increasing graph sizes, timing out beyond $d = 1000$, and requiring significantly more samples for reasonable nSHD. In contrast, SpinSVAR achieves strong results with fewer samples, particularly in the Bernoulli case. For Laplacian input, it requires slightly more samples to match that performance. Remarkably, SpinSVAR can nearly perfectly recover a window graph with 3×4000 nodes (including time lags) and 16×1000 time points in 6759s for Bernoulli input. In Appendix (E.6), we further report the nSHD performance of SparseRC and the SHD in Table 4.

Time lag k In App. E.5, we present additional experiments on the sensitivity of the time lag k, showing that SpinSVAR performance remains unaffected as long as it parametrizes a large enough time lag. In real-world datasets, where the true value of k is unknown, we choose a large enough k such that $\boldsymbol{B}_k$ is approximately zero, making it highly unlikely that meaningful dependencies exist at even higher lags.

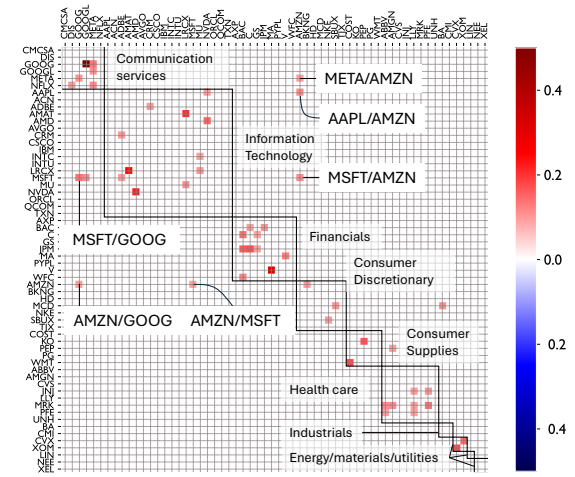
5.2 APPLICATION: S&P 500 STOCK DATA

Dataset We consider stock values from the Standard and Poor's (S&P) 500 market index. We gather data from March 1st, 2019, to March 1st, 2024, focusing only on stocks present in the index throughout this period, leaving $d = 410$ stocks as nodes. We collect daily closing values for each stock, resulting in 1259 time points per stock. The data values are computed as normalized log-returns [Pamfil et al., 2020], defined for stock i at day t as $x_{t,i} = \log(y_{t+1,i}/y_{t,i})$, where $y_{t,i}$ is the closing value. We partition the time series into shorter intervals of 50 days length to obtain time-series data $\boldsymbol{\mathcal{X}}$ of shape $25 \times 50 \times 410$. Using these data, we learn a window graph $\widehat{\boldsymbol{W}}$ that captures temporary relations between stocks and the underlying input $\widehat{\boldsymbol{\mathcal{S}}}$ that generates the data.

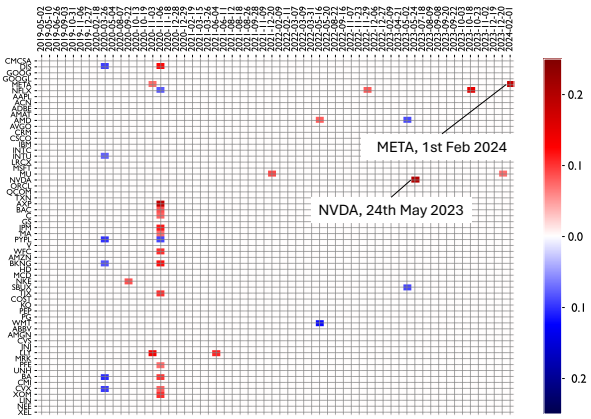
Learning stock relations We execute all baselines with hyperparameters set according to a simulated experiment shown in App. E.7. Fig. 3a shows the SpinSVAR estimate for $\widehat{\boldsymbol{B}}_0$, representing instantaneous relations between stocks. A similar figure is discovered by SparseRC, but other baselines did not yield reasonable results with our chosen hyperparameters or those from the published papers (see App. E.9). Below, we analyze this result and argue that the sparse input assumption yields interpretable results for financial data.

For better visualization, we focus on the 45 highest-weighted stocks in the S&P 500 index. In the execution of SpinSVAR, we set a maximum time lag of $k = 2$, but the method discovered that only $\boldsymbol{B}_0$ was significant. This aligns with the efficient market hypothesis [Fama, 1970], which states that stock prices fully reflect all available information, making past data redundant. Fig. 3a can be interpreted well: the edges of $\widehat{\boldsymbol{B}}_0$ roughly cluster stocks according to their economic sectors. A few outliers arise due to major IT companies being spread across multiple sectors. For example: (i) MSFT influences GOOG and AMZN, (ii) META, AAPL, and MSFT influence AMZN, and (iii) AMZN influences GOOG and MSFT. Notably, the weights of $\widehat{\boldsymbol{B}}_0$ are positive, indicating that these stocks positively influence each other: when one increases or decreases, the others do so as well.

Learning the input From the window graph approximation $\widehat{\boldsymbol{W}}$, we can estimate the input $\widehat{\boldsymbol{\mathcal{S}}}$ using (12). Fig. 3b presents this estimation for the same 45 stocks across 60 randomly chosen dates. As expected, significant input values (structural shocks) correspond to substantial changes in stock prices. To investigate this further, we evaluated all input values based on their alignment with stock price changes. We say that the input $s_{t,i}$ *aligns* with the change in data if $s_{t,i}\,(x_{t+1,i} - (1 + s_{t,i}/2)x_{t,i}) > 0$. For example, if $s_{t,i} = 0.1$ aligns with the data change, then $x_{t+1,i}$ is at least 1.05 times $x_{t,i}$. Considering the most significant $\approx 1\%$ of the $NTd = 512{,}500$ input entries of $\widehat{\boldsymbol{\mathcal{S}}}$ results in a threshold of 0.07 and amounts to 4,656 significant structural shocks, out of which 99.5% align with stock value changes. Thus,



(a) SpinSVAR estimate for $\widehat{\boldsymbol{B}}_0$



(b) SpinSVAR estimate for $\widehat{\boldsymbol{S}}$

Figure 3: Real experiment on the S&P 500 stock market index. (a) Discovered instantaneous relations $\widehat{\boldsymbol{B}}_0$ between the 45 highest weighted stocks within S&P 500, grouped by sectors (squares), and (b) the associated discovered structural shocks $\widehat{\boldsymbol{S}}$ for 60 days. In (a) the direction of influence is from row to column.

whenever a structural shock occurs at day t, the stock price at day $t + 1$[3] will increase if the value is positive (red) or decrease if the value is negative (blue).

News and dividends We conjecture that structural shocks primarily capture significant unexpected events. For instance, META had a positive structural shock of $+0.18$ on February 1, 2024 (Fig. 3b) and the same day it announced that it would pay dividends for the first time [Reuters, 2024]. Similarly, NVDA experienced a $+0.20$ structural shock on May 24, 2023, coinciding with an upward sales forecast revision due to rising AI demand [Reuters, 2023]. In contrast, significant, but expected, stock value changes like dividends deducted on the ex-dividend date are unlikely to generate structural shocks. Our dataset contains 3,796 dividend payments, yet only 36 coincided with a negative structural shock, supporting this conjecture.

[3]The structural shock effect happens on the next day as the data we consider are the log returns of stock prices.

6 LIMITATIONS

SpinSVAR inherits limitations of structure learning based on SVAR, which assumes a linear and causally stationary model. The directed edges found are not necessarily true causal relations; establishing those would require further assumptions. We implicitly assume no undersampling: the measurement frequency is at least as high as the causal effect frequency. This may affect the stock market experiment, where we used daily measurements despite stock market effects occurring within split seconds. In addition, we assume there are no missing values in the data and that measurements on each node are taken at the same frequency. Also, while we can learn DAGs with up to thousands of nodes, very large graphs beyond that remain out of reach. In our theoretical results, we assume that all structural shocks are i.i.d. Laplacian distributed. Extending our theory to allow for non-identical or dependent shocks is left for future work. While this is a limitation, our method remains applicable more broadly to sparse inputs as we demonstrated for Bernoulli inputs. Finally, our work is designed specifically for sparse SVAR input. In App. E.7, E.8, we include experiments on a simulated financial dataset and the Dream3 gene expression dataset. While our method performs competitively, it is not the best—potentially because the sparse input assumption (or even linearity) is violated.

7 CONCLUSION

We proposed SpinSVAR, a novel method for estimating SVARs from time-series data under the assumption of sparse input. By modeling the input as i.i.d. Laplacian variables, SpinSVAR is formulated as a maximum likelihood estimator based on least absolute error regression. Our method is supported by theoretical consistency guarantees and demonstrates superior performance over the state-of-the-art in experiments with synthetic and real-world financial datasets. The results highlight the utility of the sparse input assumption in uncovering interpretable structures and identifying significant events in real-world time-series data. This work opens avenues for future research in leveraging sparse input SVARs for causal discovery in time series.

References

Victor Akinwande and J Zico Kolter. AcceleratedLiNGAM: Learning Causal DAGs at the speed of GPUs. *arXiv preprint arXiv:2403.03772*, 2024.

Charles K. Assaad, Emilie Devijver, Eric Gaussier, and Ali Ait-Bachir. A Mixed Noise and Constraint-Based Approach to Causal Inference in Time Series. In *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part I 21*, pages 453–468. Springer, 2021.

Charles K. Assaad, Emilie Devijver, and Eric Gaussier. Discovery of extended summary graphs in time series. In *Uncertainty in Artificial Intelligence*, pages 96–106. PMLR, 2022a.

Charles K. Assaad, Emilie Devijver, and Eric Gaussier. Survey and Evaluation of Causal Discovery Methods for Time Series. *Journal of Artificial Intelligence Research*, 73:767–819, 2022b.

S Derin Babacan, Rafael Molina, and Aggelos K Katsaggelos. Bayesian compressive sensing using laplace priors. *IEEE Transactions on image processing*, 19(1):53–63, 2009.

Gilbert Bassett Jr and Roger Koenker. Asymptotic theory of least absolute error regression. *Journal of the American Statistical Association*, 73(363):618–622, 1978.

Alexis Bellot, Kim Branson, and Mihaela van der Schaar. Neural graphical modelling in continuous-time: consistency guarantees and algorithms. In *International Conference on Learning Representations*, 2022.

Bart Bussmann, Jannes Nys, and Steven Latré. Neural Additive Vector Autoregression Models for Causal Discovery in Time Series. In *Discovery Science: 24th International Conference, DS 2021, Halifax, NS, Canada, October 11–13, 2021, Proceedings 24*, pages 446–460. Springer, 2021.

Ismaël Castillo, Johannes Schmidt-Hieber, and Aad Van der Vaart. Bayesian linear regression with sparse priors. *The Annals of Statistics*, pages 1986–2018, 2015.

Li Chai, Jun Du, Qing-Feng Liu, and Chin-Hui Lee. Using generalized gaussian distributions to improve regression error modeling for deep learning-based speech enhancement. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(12):1919–1931, 2019.

Yuxiao Cheng, Lianglong Li, Tingxiong Xiao, Zongren Li, Jinli Suo, Kunlun He, and Qionghai Dai. CUTS+: High-dimensional Causal Discovery from Irregular Time-series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 11525–11533, 2024.

Max Chickering, David Heckerman, and Chris Meek. Large-Sample Learning of Bayesian Networks is NP-Hard. *Journal of Machine Learning Research*, 5:1287–1330, 2004.

Torbjørn Eltoft, Taesu Kim, and Te-Won Lee. On the multivariate laplace distribution. *IEEE Signal Processing Letters*, 13(5):300–303, 2006.

Doris Entner and Patrik O Hoyer. On Causal Discovery from Time Series Data using FCI. *Probabilistic graphical models*, pages 121–128, 2010.

Eugene F Fama. Efficient Capital Markets: A Review of Theory and Empirical Work. *Journal of finance*, 25(2): 383–417, 1970.

Gabriele Fiorentini and Enrique Sentana. Discrete mixtures of normals pseudo maximum likelihood estimators of structural vector autoregressions. *Journal of Econometrics*, 235(2):643–665, 2023.

Tian Gao, Debarun Bhattacharjya, Elliot Nelson, Miao Liu, and Yue Yu. IDYNO: Learning Nonparametric DAGs from Interventional Dynamic Data. In *International Conference on Machine Learning*, pages 6988–7001. PMLR, 2022.

Andreas Gerhardus and Jakob Runge. High-recall causal discovery for autocorrelated time series with latent confounders. *Advances in Neural Information Processing Systems*, 33:12615–12625, 2020.

Chang Gong, Di Yao, Chuzhe Zhang, Wenbin Li, Jingping Bi, Lun Du, and Jin Wang. Causal Discovery from Temporal Data: An Overview and New Perspectives. In *Association for Computing Machinery*, KDD '23, page 5803–5804, 2023.

Mingming Gong, Kun Zhang, Bernhard Schoelkopf, Dacheng Tao, and Philipp Geiger. Discovering Temporal Causal Relations from Subsampled Data . In *International Conference on Machine Learning*, pages 1898–1906. PMLR, 2015.

Wenbo Gong, Joel Jennings, Cheng Zhang, and Nick Pawlowski. Rhino: Deep causal temporal relationship learning with history-dependent noise. *arXiv preprint arXiv:2210.14706*, 2022.

Yue Guan and Jennifer Dy. Sparse probabilistic principal component analysis. In *Artificial Intelligence and Statistics*, pages 185–192. PMLR, 2009.

Uzma Hasan, Emam Hossain, and Md Osman Gani. A Survey on Causal Discovery Methods for I.I.D. and Time Series Data . *Transactions on Machine Learning Research*, 2023.

Xin He and ZhongKui Sun. Sparse identification of dynamical systems by reweighted l1-regularized least absolute deviation regression. *Communications in Nonlinear Science and Numerical Simulation*, 131:107813, 2024.

Roger A. Horn and Charles R. Johnson. *Matrix analysis*. Cambridge university press, 2012.

Aapo Hyvärinen, Kun Zhang, Shohei Shimizu, and Patrik O Hoyer. Estimation of a Structural Vector Autoregression Model Using Non-Gaussianity. *Journal of Machine Learning Research*, 11(5), 2010.

Feng Jiang, Lin Du, Fan Yang, and Zi-Chen Deng. Regularized least absolute deviation-based sparse identification of dynamical systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 33(1), 2023.

Yi Jiang and Shohei Shimizu. Linkages among the Foreign Exchange, Stock, and Bond Markets in Japan and the United States. In *Causal Analysis Workshop Series*, pages 1–19. PMLR, 2023.

Liping Jing, Peng Wang, and Liu Yang. Sparse probabilistic matrix factorization by laplace distribution for collaborative filtering. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

Markus Kalisch and Peter Bühlman. Estimating high-dimensional directed acyclic graphs with the pc-algorithm. *Journal of Machine Learning Research*, 8 (3), 2007.

Saurabh Khanna and Vincent YF Tan. Economy Statistical Recurrent Units For Inferring Nonlinear Granger Causality. In *International Conference on Learning Representations*, 2019.

Lutz Kilian. Structural Vector Autoregressions. In *Handbook of research methods and applications in empirical macroeconomics*, pages 515–554. Edward Elgar Publishing, 2013.

Hyoungshick Kim and Ross Anderson. Temporal node centrality in complex networks. *Physical Review E*, 85 (2):026107, 2012.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Samantha Kleinberg. *Causality, Probability, and Time*. Cambridge University Press, 2013.

Pranesh Kumar and Jai Narain Singh. Regression model estimation using least absolute deviations, least squares deviations and minimax absolute deviations criteria. *IJCSEE*, 3(4):2320–4028, 2015.

Sébastien Lachapelle, Philippe Brouillard, Tristan Deleu, and Simon Lacoste-Julien. Gradient-based neural dag learning. In *International Conference on Learning Representations*, 2019.

Markku Lanne, Mika Meitz, and Pentti Saikkonen. Identification and estimation of non-gaussian structural vector autoregressions. *Journal of Econometrics*, 196(2):288–304, 2017a.

Markku Lanne, Mika Meitz, and Pentti Saikkonen. Identification and estimation of non-Gaussian structural vector autoregressions. *Journal of Econometrics*, 196(2):288–304, 2017b.

Yinbo Li and Gonzalo R Arce. A fast maximum likelihood estimation approach to lad regression. In *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages ii–889. IEEE, 2004.

Sindy Löwe, David Madras, Richard Zemel, and Max Welling. Amortized Causal Discovery: Learning to Infer Causal Graphs from Time-Series Data. In *Conference on Causal Learning and Reasoning*, pages 509–525. PMLR, 2022.

Helmut Lütkepohl. *New Introduction to Multiple Time Series Analysis*. Springer Science & Business Media, 2005.

Koichi Maekawa and Tadashi Nakanishi. Estimation of non-gaussian svar models: a pseudo-log-likelihood function approach. *Journal of Statistical Computation and Simulation*, 93(11):1830–1850, 2023.

Daniel Malinsky and Peter Spirtes. Causal Structure Learning from Multivariate Time Series in Settings with Unmeasured Confounding. In *Proceedings of 2018 ACM SIGKDD workshop on causal discovery*, pages 23–47. PMLR, 2018.

Daniel Marbach, Thomas Schaffter, Claudio Mattiussi, and Dario Floreano. Generating realistic in silico gene networks for performance assessment of reverse engineering methods. *Journal of computational biology*, 16(2):229–239, 2009.

Ričards Marcinkevičs and Julia E Vogt. Interpretable Models for Granger Causality Using Self-explaining Neural Networks. In *International Conference on Learning Representations*, 2020.

Panagiotis Misiakos, Chris Wendler, and Markus Püschel. Learning DAGs from Data with Few Root Causes. *Advances in Neural Information Processing Systems*, 36, 2023.

Panagiotis Misiakos, Vedran Mihal, and Markus Püschel. Learning Signals and Graphs from Time-Series Graph Data with Few Causes. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 9681–9685, 2024.

Subhash C Narula, Paulo HN Saldiva, Carmen DS Andre, Silvia N Elian, Aurea Favero Ferreira, and Vera Capelozzi. The minimum sum of absolute errors regression: a robust alternative to the least squares regression. *Statistics in medicine*, 18(11):1401–1417, 1999.

Meike Nauta, Doina Bucur, and Christin Seifert. Causal Discovery with Attention-Based Convolutional Neural Networks. *Machine Learning and Knowledge Extraction*, 1(1):19, 2019.

Whitney K Newey and Daniel McFadden. Large sample estimation and hypothesis testing. *Handbook of econometrics*, 4:2111–2245, 1994.

Ignavier Ng, AmirEmad Ghassami, and Kun Zhang. On the Role of Sparsity and DAG Constraints for Learning Linear DAGs. *Advances in Neural Information Processing Systems*, 33:17943–17954, 2020.

Roxana Pamfil, Nisara Sriwattanaworachai, Shaan Desai, Philip Pilgerstorfer, Konstantinos Georgatzis, Paul Beaumont, and Bryon Aragam. DYNOTEARS: Structure Learning from Time-Series Data. In *International Conference on Artificial Intelligence and Statistics*, pages 1595–1605. PMLR, 2020.

Gunwoong Park. Identifiability of Additive Noise Models Using Conditional Variances. *J. Mach. Learn. Res.*, 21 (75):1–34, 2020.

Judea Pearl. *Causality*. Cambridge university press, 2009.

Jonas Peters and Peter Bühlmann. Identifiability of Gaussian structural equation models with equal error variances. *Biometrika*, 101(1):219–228, 2014.

Jonas Peters and Peter Bühlmann. Structural intervention distance for evaluating causal graphs. *Neural computation*, 27(3):771–799, 2015.

Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. Causal Inference on Time Series using Structural Equation Models. *Advances in neural information processing systems*, 26, 2013.

Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference: foundations and learning algorithms*. The MIT Press, 2017.

David Pollard. Asymptotics for least absolute deviation regression estimators. *Econometric Theory*, 7(2):186–199, 1991.

Robert J Prill, Daniel Marbach, Julio Saez-Rodriguez, Peter K Sorger, Leonidas G Alexopoulos, Xiaowei Xue, Neil D Clarke, Gregoire Altan-Bonnet, and Gustavo Stolovitzky. Towards a rigorous assessment of systems biology models: the dream3 challenges. *PloS one*, 5(2): e9202, 2010.

Reuters. Nvidia shares soar nearly 30% as sales forecast jumps and ai booms. `https://www.reuters.com/technology/nvidia-forecasts-second-quarter-revenue-above-` 2023. Accessed: 2024-05-21.

Reuters. Facebook parent meta declares first dividend, shares soar. `https://www.reuters.com/technology/facebook-parent-meta-declares-first-ever-divid` 2024. Accessed: 2024-05-21.

Jakob Runge. Discovering contemporaneous and lagged causal relations in autocorrelated nonlinear time series datasets. In *Conference on Uncertainty in Artificial Intelligence*, pages 1388–1397. PMLR, 2020.

Jakob Runge, Peer Nowack, Marlene Kretschmer, Seth Flaxman, and Dino Sejdinovic. Detecting and quantifying causal associations in large nonlinear time series datasets. *Science advances*, 5(11):eaau4996, 2019.

Pentti Saikkonen. *Stability results for nonlinear vector autoregressions with an application to a nonlinear error correction model*. Humboldt-Universität zu Berlin, Wirtschaftswissenschaftliche Fakultät, 2001.

Bastian Seifert, Chris Wendler, and Markus Püschel. Causal Fourier Analysis on Directed Acyclic Graphs and Posets. *IEEE Trans. Signal Process.*, 71:3805–3820, 2023. doi: 10.1109/TSP.2023.3324988.

Shohei Shimizu, Patrik O. Hoyer, Aapo Hyvärinen, and Antti Kerminen. A Linear Non-Gaussian Acyclic Model for Causal Discovery. *Journal of Machine Learning Research*, 7(72):2003–2030, 2006. URL `http://jmlr.org/papers/v7/shimizu06a.html`.

Shohei Shimizu, Takanori Inazumi, Yasuhiro Sogawa, Aapo Hyvarinen, Yoshinobu Kawahara, Takashi Washio, Patrik O Hoyer, Kenneth Bollen, and Patrik Hoyer. DirectLiNGAM: A direct method for learning a linear non-Gaussian structural equation model. *Journal of Machine Learning Research-JMLR*, 12(Apr):1225–1248, 2011.

Christopher A Sims. Comparison of Interwar and Postwar Business Cycles: Monetarism Reconsidered, 1980.

Sigurd Skogestad and Ian Postlethwaite. *Multivariable Feedback Control: Analysis and Design*. john Wiley & sons, 2005.

Stephen M Smith, Karla L Miller, Gholamreza Salimi-Khorshidi, Matthew Webster, Christian F Beckmann, Thomas E Nichols, Joseph D Ramsey, and Mark W Woolrich. Network modelling methods for FMRI. *Neuroimage*, 54(2):875–891, 2011.

George Sugihara, Robert May, Hao Ye, Chih-hao Hsieh, Ethan Deyle, Michael Fogarty, and Stephan Munch. Detecting Causality in Complex Ecosystems. *science*, 338 (6106):496–500, 2012.

Xiangyu Sun, Oliver Schulte, Guiliang Liu, and Pascal Poupart. NTS-NOTEARS: Learning Nonparametric DBNs With Prior Knowledge. In *International Conference on Artificial Intelligence and Statistics*, pages 1942–1964. PMLR, 2023.

Wilson A Sutherland. *Introduction to metric and topological spaces*. Oxford University Press, 2009.

Alex Tank, Ian Covert, Nicholas Foti, Ali Shojaie, and Emily B Fox. Neural Granger Causality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44 (8):4267–4279, 2021.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288, 1996.

Matthew J Vowels, Necati Cihan Camgoz, and Richard Bowden. D'ya like DAGs? A survey on structure learning and causal discovery. *ACM Computing Surveys (CSUR)*, 2021.

Chenxiao Xu, Hao Huang, and Shinjae Yoo. Scalable Causal Graph Learning through a Deep Neural Network. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1853–1862, 2019.

Xueli Yang, Zhi-Hua Wang, Chenghao Wang, and Ying-Cheng Lai. Detecting the causal influence of thermal environments among climate regions in the United States. *Journal of Environmental Management*, 322: 116001, 2022.

Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing. DAGs with NO TEARS: Continuous Optimization for Structure Learning. *Advances in Neural Information Processing Systems*, 31, 2018.

SpinSVAR: Estimating Structural Vector Autoregression Assuming Sparse Input (Supplementary material)

Panagiotis Misiakos[1] **Markus Püschel**[1]

[1]Department of Computer Science, ETH Zurich, Zürich, Switzerland

ETHICS

SpinSVAR inherits the broader impact of other DAG learning methods from time series. From an ethical viewpoint, the methodology is generic and poses no specific potential risk.

REPRODUCIBILITY

We acknowledge the importance of reproducibility and here we explain the actions that we took towards a more effortless reproduction of our results.

Code We provide our code written in Python 3.9 as supplementary material and will make it available on github upon acceptance. In the README.md file, we explain the Python environment installation, how the code can be executed, and provide a Jupyter notebook demonstrating a synthetic experiment. More importantly, our code not only provides an implementation of our method but rather the whole experimental pipeline, showing how the data are generated and how the baselines are applied.

Data The sparse input SVAR data generation can be executed using our code or reproduced according to the parameters explained in the experimental section of the main text and the details in Appendix E.1. For the simulated financial and the S&P 500 data we provide in Appendix E.13 the sources to download them.

Methods We have explained in great detail in the main text the optimization problem solved by SpinSVAR and the adapted version of SparseRC that we use for fair comparison, also explained in Appendix B. For the execution of all baselines, we use publicly available repositories listed in E.12 with hyperparameters set as shown in E.10. Competitor methods can also be executed using the provided code.

A MATHEMATICAL PROOFS AND COMPUTATIONS

In this section we provide all the proofs of technical results used in the manuscript.

A.1 SVAR STABILITY

Whenever a measurement can be taken in a system, stability in the measured data holds by definition. For example, temperature measurements or stock price markets are never unbounded. To ensure that the same happens for synthetic data, one needs to guarantee the stability of the data generation process. A few prior works mention stability [Gong et al., 2015, Khanna and Tan, 2019, Bellot et al., 2022, Malinsky and Spirtes, 2018], and here we want to acknowledge its importance.

Equation (3) can be viewed as a discrete-time multi-input multi-output (MIMO) system [Skogestad and Postlethwaite, 2005],

in which the input is the structural shocks $\boldsymbol{S}$ and the output is the time-series data $\boldsymbol{X}$. As the time-series length T in (2) increases, the values of $\boldsymbol{X}$ can get arbitrarily large. We desire to find a range of weights for the matrices $\boldsymbol{B}_0, \boldsymbol{B}_1, ..., \boldsymbol{B}_k$ that guarantees that our time-series data are bounded. In particular, we require a condition for the bounded-input bounded-output (BIBO) stability of this system. This has been already considered by Lütkepohl [2005] (linear case, for non-linear refer to [Saikkonen, 2001]). The proposed condition requires the roots of the reverse characteristic polynomial to have a modulus less than 1. Here, we prove a practical and intuitive condition for stability as a derivation of the [Lütkepohl, 2005] result.

Transitive closure To begin, we introduce the definition of the weighted transitive closure of the unrolled DAG (43).

$$\widetilde{\boldsymbol{X}} = \widetilde{\boldsymbol{X}}\boldsymbol{A} + \widetilde{\boldsymbol{S}} \Leftrightarrow \widetilde{\boldsymbol{X}} = \widetilde{\boldsymbol{S}}\left(\boldsymbol{I} - \boldsymbol{A}\right)^{-1} = \widetilde{\boldsymbol{S}}\left(\boldsymbol{I} + \overline{\boldsymbol{A}}\right), \tag{13}$$

On the right hand (13) $\overline{\boldsymbol{A}} = \boldsymbol{A} + ... + \boldsymbol{A}^{dT-1}$ is the weighted transitive closure [Seifert et al., 2023] of the unrolled DAG $\boldsymbol{A}$.

Stability of model (3) We will now prove Theorem A.1 that we are interested in. This provides a sufficient condition under which the model (3) is BIBO stable. BIBO stability here means that if the input $\boldsymbol{S}$ is bounded, then so are the output measurements $\boldsymbol{X}$.

Theorem A.1. *The model (3) is BIBO stable if for some (sub-multiplicative) matrix norm $\|\cdot\|$:*

$$\|\boldsymbol{W}\| < 1$$

Proof. If $\|\boldsymbol{W}\| = \lambda < 1$ then from the structure of $\boldsymbol{A}$ also $\|\boldsymbol{A}\| = \|\boldsymbol{W}\| = \lambda < 1$. Therefore:

$$\left\|\boldsymbol{I} + \overline{\boldsymbol{A}}\right\| = \left\|\boldsymbol{I} + \boldsymbol{A} + ... + \boldsymbol{A}^{dT-1}\right\| \leq \sum_{t=0}^{dT-1} \|\boldsymbol{A}\|^t \leq \sum_{t=0}^{dT-1} \lambda^t \leq \sum_{t=0}^{\infty} \lambda^t = \frac{1}{1-\lambda} = M$$

Thus

$$\begin{aligned}
\lim_{T\to\infty} \|\boldsymbol{X}\| &= \lim_{T\to\infty} \left\|\left(\boldsymbol{I} + \overline{\boldsymbol{A}}\right)\boldsymbol{S}\right\| \\
&\leq \lim_{T\to\infty} \left\|\boldsymbol{I} + \overline{\boldsymbol{A}}\right\| \|\boldsymbol{S}\| \\
&\leq M \|\boldsymbol{S}\|
\end{aligned}$$

This implies that $\|\boldsymbol{X}\|$ is bounded for all T and the model (3) is BIBO stable. □

Example Consider the induced $L^\infty-$norm as $\|\boldsymbol{A}\|_\infty = \max_j \sum_{i=1}^d |a_{ij}|$. The induced $L^\infty-$norm is sub-multiplicative and thus Theorem A.1 can be utilized. In fact it can be proved that any induced vector norm is sub-multiplicative (Theorem 5.6.2 in [Horn and Johnson, 2012]). Then, condition $\|\boldsymbol{W}\|_\infty < 1$ translates to all outcoming weights (rows of the window graph matrix) having the sum of absolute values less than 1.

For the sake of completeness, we provide a proof of the submultiplicativity property of the $L^\infty-$norm in Lemma A.2.

Lemma A.2. *The induced $L^\infty-$norm is submultiplicative.*

Proof. Consider any two square matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{d\times d}$. We need to show that $\|\boldsymbol{AB}\| \leq \|\boldsymbol{A}\| \|\boldsymbol{B}\|$. Indeed,

$$\begin{aligned}
\|\boldsymbol{AB}\| &= \max_i \sum_{j=1}^d \left|\sum_{k=1}^d a_{ik} b_{kj}\right| \\
&\leq \max_i \sum_{j=1}^d \sum_{k=1}^d |a_{ik} b_{kj}| \\
&= \max_i \sum_{k=1}^d \sum_{j=1}^d |a_{ik}|\,|b_{kj}|
\end{aligned}$$

$$
\begin{aligned}
&= \max_i \sum_{k=1}^{d} |a_{ik}| \left(\sum_{j=1}^{d} |b_{kj}| \right) \\
&\leq \max_i \sum_{k=1}^{d} |a_{ik}| \left(\max_k \sum_{j=1}^{d} |b_{kj}| \right) \\
&= \max_i \sum_{k=1}^{d} |a_{ik}| \, \|\boldsymbol{B}\| \\
&\leq \|\boldsymbol{A}\| \, \|\boldsymbol{B}\|
\end{aligned}
$$

□

Example The $L^\infty-$norm is particularly interesting for our scenario as the condition of Theorem A.1 provides an intuitive interpretation for the weights. Consider our stock market example. Then the condition in A.1 means that for every stock that affects a set of other stocks, each with some factor < 1, the total sum should be less than 1. Of course, this is only a sufficient condition for the data to be bounded, but we believe that it is meaningful to consider that the influences between stocks are of this form in reality. To understand better why the condition in A.1 provides bounded data, we can think about it in the following way. When the $L^\infty-$norm is bounded, the total effect of a stock is divided into individual fractions that affect other stocks and doesn't get iteratively increased (which could be the case with sum $L^\infty-$norm > 1). Bounding the sum of outcoming weights to 1 has also been considered in [Seifert et al., 2023, Misiakos et al., 2023] in the scenario of pollution propagation in a river network.

We further include another submultiplicative property, that we later use on our proofs.

Lemma A.3. *The L^1-norm, defined as sum of absolut values of the entries of a matrix is submultiplicative.*

Proof. Consider any two square matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{d \times d}$. We want to show that $\|\boldsymbol{AB}\|_1 \leq \|\boldsymbol{A}\|_1 \, \|\boldsymbol{B}\|_1$. Indeed,

$$
\begin{aligned}
\|\boldsymbol{AB}\|_1 &= \sum_{i=1}^{d} \sum_{j=1}^{d} \left| \sum_{k=1}^{d} a_{ik} b_{kj} \right| \\
&\leq \sum_{i=1}^{d} \sum_{j=1}^{d} \sum_{k=1}^{d} |a_{ik} b_{kj}| \\
&= \sum_{i=1}^{d} \sum_{k=1}^{d} \sum_{j=1}^{d} |a_{ik}| \, |b_{kj}| \\
&= \sum_{i=1}^{d} \sum_{k=1}^{d} \sum_{j=1}^{d} \sum_{l=1}^{d} |a_{ik}| \, \mathbb{1}_{k=l} \, |b_{lj}| \\
&\leq \sum_{i=1}^{d} \sum_{k=1}^{d} \sum_{j=1}^{d} \sum_{l=1}^{d} |a_{ik}| \, |b_{lj}| \\
&= \left(\sum_{i=1}^{d} \sum_{k=1}^{d} |a_{ik}| \right) \left(\sum_{j=1}^{d} \sum_{l=1}^{d} |b_{lj}| \right) \\
&\leq \|\boldsymbol{A}\|_1 \, \|\boldsymbol{B}\|_1
\end{aligned}
$$

□

A.2 IDENTIFIABILITY

Theorem A.4. *Consider the time-series model (3) with $\boldsymbol{S}$ following a multivariate Laplace distribution as in (4) with $\beta \in [a, b]$ and $a > \frac{1}{NTd}$. Then the matrices $\boldsymbol{B}_0, \boldsymbol{B}_1, ..., \boldsymbol{B}_k \in \mathbb{R}^{d\times d}$ and β are identifiable from the time-series data $\boldsymbol{\mathcal{X}}$.*

Proof. Recall that the SVAR model in (2) is:

$$\boldsymbol{x}_t = \boldsymbol{x}_t\boldsymbol{B}_0 + \boldsymbol{x}_{t-1}\boldsymbol{B}_1 + \cdots + \boldsymbol{x}_{t-k}\boldsymbol{B}_k + \mathbf{s}_t$$

As we explain later in Appendix B we can rewrite the SVAR (2) as a linear SEM. We collect all observations $\boldsymbol{x}_t$ for $t = 0, 1, \ldots, T-1$ into a single row vector (one long time series) $\tilde{\boldsymbol{x}} = \begin{pmatrix}\boldsymbol{x}_0 & \boldsymbol{x}_1 & \ldots & \boldsymbol{x}_{T-1}\end{pmatrix} \in \mathbb{R}^{1\times dT}$, which gives rise to the "unrolled DAG equation":

$$\begin{pmatrix}\boldsymbol{x}_0 & \boldsymbol{x}_1 & \ldots & \boldsymbol{x}_{T-1}\end{pmatrix} = \begin{pmatrix}\boldsymbol{x}_0 & \boldsymbol{x}_1 & \ldots & \boldsymbol{x}_{T-1}\end{pmatrix} \begin{pmatrix} \boldsymbol{B}_0 & \boldsymbol{B}_1 & \ldots & \boldsymbol{B}_k & \ldots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{B}_0 & \boldsymbol{B}_1 & & \ddots & \boldsymbol{B}_1 \\ \vdots & \mathbf{0} & \boldsymbol{B}_0 & \ddots & & \boldsymbol{B}_k \\ & \ddots & \mathbf{0} & \ddots & \boldsymbol{B}_1 & \vdots \\ \vdots & & \ddots & \ddots & \boldsymbol{B}_0 & \boldsymbol{B}_1 \\ \mathbf{0} & \ldots & & \ldots & \mathbf{0} & \boldsymbol{B}_0 \end{pmatrix} + \begin{pmatrix}\mathbf{s}_0 & \mathbf{s}_1 & \ldots & \mathbf{s}_{T-1}\end{pmatrix}$$

Equivalently, we can write:

$$\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{x}}\mathbf{A} + \tilde{\mathbf{s}} \tag{14}$$

Here, $\mathbf{A}$ is a directed acyclic graph matrix, since it is upper (block) triangular, and $\boldsymbol{B}_0$ is assumed to be acyclic. This is the unrolled DAG representation (43).

Because $\tilde{\mathbf{s}}$ contains i.i.d. and Laplace-distributed (i.e., non-Gaussian) components, the model 14 satisfies the assumptions of LiNGAM [Shimizu et al., 2006]. Therefore, the matrix $\mathbf{A}$ is identifiable from the distribution of $\tilde{\boldsymbol{x}}$. Moreover, identifiability on $\boldsymbol{A}$ implies identifiability for the parameters $\boldsymbol{B}_0, \boldsymbol{B}_1, ..., \boldsymbol{B}_k$ of the window graph $\boldsymbol{W}$, as desired.

Note that this result is not affected if our dataset contains more than 1 sample ($N > 1$), which only benefits the distribution estimation. For N samples we have:

$$\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{X}}_{\text{past}}\boldsymbol{W} + \boldsymbol{\mathcal{S}} \Leftrightarrow \widetilde{\boldsymbol{X}} = \widetilde{\boldsymbol{X}}\boldsymbol{A} + \widetilde{\boldsymbol{S}}. \tag{15}$$

where $\widetilde{\boldsymbol{X}}, \widetilde{\boldsymbol{S}} \in \mathbb{R}^{N\times dT}$ and $\boldsymbol{A} \in \mathbb{R}^{dT\times dT}$.

We will now establish identifiability of β using the monotonicity of the Laplacian probability distribution. Notice, identifiability on $\boldsymbol{W}$ means, that for any $\boldsymbol{W} \in \mathcal{W}$ and any $\beta \in [a, b]$, the equation $f_X(\boldsymbol{\mathcal{X}}|\boldsymbol{W}, \beta) = f_X(\boldsymbol{\mathcal{X}}|\boldsymbol{W}^*, \beta^*)$ gives $\boldsymbol{W} = \boldsymbol{W}^*$. This in turn implies that the parameter β is identifiable. Indeed:

$$f_X(\boldsymbol{\mathcal{X}}|\boldsymbol{W}^*, \beta) = |\det(\boldsymbol{I} - \boldsymbol{B}_0)|^{NT} \frac{1}{(2\beta)^{NTd}} e^{-\frac{\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}_{\text{past}}\boldsymbol{W}^*\|_1}{\beta}} \tag{16}$$

The derivative with respect to β is:

$$\frac{\partial f_X}{\partial \beta} = |\det(\boldsymbol{I} - \boldsymbol{B}_0)|^{NT} \left(\frac{1}{\beta} - NTd\right) \frac{\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}_{\text{past}}\boldsymbol{W}^*\|_1}{2^{NTd}\beta^{NTd+1}} e^{-\frac{\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}_{\text{past}}\boldsymbol{W}^*\|_1}{\beta}} < 0 \tag{17}$$

Therefore, f_X is monotonically decreasing and thus bijective for $\beta > \frac{1}{NTd}$. Therefore:

$$f_X(\boldsymbol{\mathcal{X}}|\boldsymbol{W}, \beta) = f_X(\boldsymbol{\mathcal{X}}|\boldsymbol{W}^*, \beta^*) \xRightarrow{\text{LiNGAM}} \boldsymbol{W} = \boldsymbol{W}^* \text{ and } f_X(\boldsymbol{\mathcal{X}}|\boldsymbol{W}^*, \beta) = f_X(\boldsymbol{\mathcal{X}}|\boldsymbol{W}^*, \beta^*) \xRightarrow{\text{monotonicity}} \beta = \beta^* \tag{18}$$

Thus, $(\boldsymbol{W}, \beta)$ are identifiable from the data $\boldsymbol{\mathcal{X}}$. □

Remark A.5. In Section 5.1.1 in the VARLiNGAM paper [Hyvärinen et al., 2010], the way LiNGAM identifiability is invoked in Step 3 of their argument is fundamentally different from our approach in (14). Specifically, Hyvärinen et al. [2010] first perform standard autoregressive (AR) estimation, and then apply LiNGAM identifiability to the resulting residuals. In contrast, our proof directly constructs a system of SVAR equations, leading to an **unrolled DAG representation** over time, on which we then apply LiNGAM.

A.3 MLE COMPUTATION

Estimator computation Here we compute the MLE assuming that each entry of the structural shocks $\boldsymbol{S} \in \mathbb{R}^{d\times T}$ follows independently a Laplace distribution Laplace $(0, \beta)$. The multivariate probability density function of $\boldsymbol{S}$ is:

$$f_C(\boldsymbol{S}) = \prod_{\tau,j} \frac{1}{2\beta} e^{-\frac{|\boldsymbol{s}_{\tau,j}|}{\beta}} \tag{19}$$

Solving with respect to $\boldsymbol{X}$ equation (3) gives $\boldsymbol{X} = \boldsymbol{S}(\boldsymbol{I} - \boldsymbol{A})^{-1}$ where $\boldsymbol{A} \in \mathbb{R}^{dT\times dT}$ is the unrolled DAG matrix of $\boldsymbol{W}$ according to (43). Here, we didn't change our notation, but $\boldsymbol{X}$ and $\boldsymbol{S}$ are supposed to represent $1 \times dT$ dimensional vectors. For simplicity we will do this interchange in the following computations as it doesn't affect the probability distribution. Using this linear transformation the probability density function (pdf) of $\boldsymbol{X}$, or likelihood of the data, becomes

$$\begin{aligned} f_X(\boldsymbol{X}|\boldsymbol{W}, \beta) &= \frac{f_C\left(\boldsymbol{X}\left(\boldsymbol{I} - \boldsymbol{A}\right)\right)}{\left|\det\left(\left(\boldsymbol{I} - \boldsymbol{A}\right)^{-1}\right)\right|} \\ &= \left|\det\left(\boldsymbol{I} - \boldsymbol{A}\right)\right| \prod_{\tau,j} \frac{1}{2\beta} e^{-\frac{|\boldsymbol{x}_{\tau,j} - \boldsymbol{x}_{\text{past}\tau,:}\boldsymbol{W}_{:,j}|}{\beta}} \\ &= \left|\det\left(\boldsymbol{I} - \boldsymbol{B}_0\right)^T\right| \prod_{\tau,j} \frac{1}{2\beta} e^{-\frac{|\boldsymbol{x}_{\tau,j} - \boldsymbol{x}_{\text{past}\tau,:}\boldsymbol{W}_{:,j}|}{\beta}} \\ &= \left|\det\left(\boldsymbol{I} - \boldsymbol{B}_0\right)\right|^T \frac{1}{(2\beta)^{dT}} e^{-\frac{\|\boldsymbol{x} - \boldsymbol{x}_{\text{past}}\boldsymbol{W}\|_1}{\beta}} \end{aligned}$$

Therefore, for N realizations of $\boldsymbol{X}$ in the tensor $\boldsymbol{\mathcal{X}}$ we have that:

$$f_X(\boldsymbol{\mathcal{X}}|\boldsymbol{W}, \beta) = \left|\det\left(\boldsymbol{I} - \boldsymbol{B}_0\right)\right|^{NT} \frac{1}{(2\beta)^{NdT}} e^{-\frac{\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}_{\text{past}}\boldsymbol{\mathcal{W}}\|_1}{\beta}}, \tag{20}$$

which in turn gives the log-likelihood for the data:

$$\begin{aligned} \mathcal{L}\left(\boldsymbol{W}, \beta; \boldsymbol{\mathcal{X}}\right) &= \log f_X\left(\boldsymbol{\mathcal{X}}|\boldsymbol{W}, \beta\right) \\ &= NT \log\left|\det\left(\boldsymbol{I} - \boldsymbol{B}_0\right)\right| - NTd\log(2\beta) - \frac{1}{\beta}\left\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}_{\text{past}}\boldsymbol{W}\right\|_1. \end{aligned} \tag{21}$$

In what follows, for simplicity of notation we will skip the parameter β and will use $\mathcal{L}\left(\boldsymbol{W}, \beta; \boldsymbol{\mathcal{X}}\right)$ and $\mathcal{L}\left(\boldsymbol{W}; \boldsymbol{\mathcal{X}}\right)$ interchangeably.

A.4 MLE CONSISTENCY BACKGROUND

We proceed by analyzing the prior theorems that we will use to prove our results. First, denote with $L\left(\boldsymbol{W}\right)$ the population log-likelihood [Lachapelle et al., 2019, Newey and McFadden, 1994], defined as:

$$L\left(\boldsymbol{W}, \beta\right) = \mathbb{E}_{\boldsymbol{W}^*,\beta^*}\left[\mathcal{L}\left(\boldsymbol{W}, \beta; \boldsymbol{\mathcal{X}}\right)\right]. \tag{22}$$

Note that we use $\mathcal{L}\left(\boldsymbol{W}, \beta; \boldsymbol{\mathcal{X}}\right)$ and $\mathcal{L}\left(\boldsymbol{W}; \boldsymbol{\mathcal{X}}\right)$ interchangeably, as well as $L\left(\boldsymbol{W}, \beta\right)$ and $L\left(\boldsymbol{W}\right)$.

In essence, the population log-likelihood is the expected value of the log-likelihood function computed with the probability density $f_X(\boldsymbol{\mathcal{X}}|\boldsymbol{W}^*, \beta^*)$ with parameters the ground truth window graph $\boldsymbol{W}^*$ and parameter β^*. This expected value is computed over the distribution of $\mathbf{X}$, parametrized assuming ground truth parameters $\mathbf{W}^*, \beta^*$. Formally, we have:

$$L(\mathbf{W}, \beta) = \mathbb{E}_{\mathbf{W}^*,\beta^*}[\mathcal{L}(\mathbf{W}, \beta; \mathbf{X})] = \int_{\mathbf{X}\sim\mathbb{P}_{\mathbf{W}^*,\beta^*}(\mathbf{X})} \mathcal{L}(\mathbf{W}, \beta; \mathbf{X}) d\mathbf{X}$$

Lemma A.6. *Assume that the ground truth window graph $\boldsymbol{W}^*$ and parameter β^* are identifiable from the data distribution. This means, that for $(\boldsymbol{W}, \beta) \neq (\boldsymbol{W}^*, \beta^*)$ it is true that $f_X(\boldsymbol{\mathcal{X}}|\boldsymbol{W}, \beta) \neq f_X(\boldsymbol{\mathcal{X}}|\boldsymbol{W}^*, \beta^*)$. Then, the population likelihood $L(\boldsymbol{W}, \beta)$ has unique maximum at the true window graph $\boldsymbol{W}^*$ and true β^*.*

Proof. We show that $L(\boldsymbol{W}^*, \beta^*) > L(\boldsymbol{W}, \beta)$ for every $(\boldsymbol{W}, \beta) \neq (\boldsymbol{W}^*, \beta^*)$. By simplifying our notation we have:

$$\begin{aligned}
L(\boldsymbol{W}^*) - L(\boldsymbol{W}) &= \mathbb{E}_{\boldsymbol{W}^*}\left[\mathcal{L}(\boldsymbol{W}^*; \boldsymbol{\mathcal{X}}) - \mathcal{L}(\boldsymbol{W}; \boldsymbol{\mathcal{X}})\right] \\
&= \mathbb{E}_{\boldsymbol{W}^*}\left[-\log \frac{f_X(\boldsymbol{\mathcal{X}}|\boldsymbol{W})}{f_X(\boldsymbol{\mathcal{X}}|\boldsymbol{W}^*)}\right] \\
&> -\log \mathbb{E}_{\boldsymbol{W}^*}\left[\frac{f_X(\boldsymbol{\mathcal{X}}|\boldsymbol{W})}{f_X(\boldsymbol{\mathcal{X}}|\boldsymbol{W}^*)}\right] \\
&= -\log \int_{\boldsymbol{\mathcal{X}} \in \mathbb{R}^{N \times T \times d}} \frac{f_X(\boldsymbol{\mathcal{X}}|\boldsymbol{W})}{f_X(\boldsymbol{\mathcal{X}}|\boldsymbol{W}^*)} f_X(\boldsymbol{\mathcal{X}}|\boldsymbol{W}^*)\, d\boldsymbol{\mathcal{X}} \\
&= -\log \int_{\boldsymbol{\mathcal{X}} \in \mathbb{R}^{N \times T \times d}} f_X(\boldsymbol{\mathcal{X}}|\boldsymbol{W})\, d\boldsymbol{\mathcal{X}} \\
&= -\log 1 = 0
\end{aligned}$$

On the second line we used that $\frac{f_X(\boldsymbol{\mathcal{X}}|\boldsymbol{W})}{f_X(\boldsymbol{\mathcal{X}}|\boldsymbol{W}^*)}$ is non-constant, so we can apply the strict Jensen inequality [Newey and McFadden, 1994] $\mathbb{E}[a(\boldsymbol{Y})] > \mathbb{E}[a(\boldsymbol{Y})]$ for a convex function a and non-constant random variable $\boldsymbol{Y}$. □

As a next result for our toolset to prove the MLE consistency, we include the uniform law of large numbers as stated by Newey and McFadden [1994].

Lemma A.7 (Uniform Law of Large Numbers)**.** *Consider that the log-likelihood function $\mathcal{L}(\boldsymbol{W}; \boldsymbol{\mathcal{X}})$, $\boldsymbol{W} \in \mathcal{W}$ satisfy the following conditions.*

- *The data $\boldsymbol{\mathcal{X}}_i$ are independent and identically distributed.*
- *$\mathcal{W}$ is a compact space.*
- *$\mathcal{L}(\boldsymbol{\mathcal{X}}_i; \boldsymbol{W})$, $\boldsymbol{W} \in \mathcal{W}$ is continuous at each $\boldsymbol{W} \in \mathcal{W}$ with probability 1.*
- *There exists dominating function $D(\boldsymbol{W})$ such that $|\mathcal{L}(\boldsymbol{W}; \boldsymbol{\mathcal{X}})| \leq D(\boldsymbol{W})$ and $\mathbb{E}_{\boldsymbol{W}^*}[D(\boldsymbol{W})] < \infty$.*

Then the population likelihood $L(\boldsymbol{W})$ and the empirical average log-likelihood converges uniformly in probability to it:

$$\sup_{\boldsymbol{W} \in \mathcal{W}} \left| \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(\boldsymbol{\mathcal{X}}_i; \boldsymbol{W}) - L(\boldsymbol{W}) \right| \xrightarrow{p} 0 \tag{23}$$

We now present Theorem A.8, which establishes the consistency of the maximum likelihood estimator (MLE). This theorem is based on a set of sufficient assumptions for ensuring MLE consistency. For completeness, we include a detailed proof of Theorem A.8, leveraging the uniform law of large numbers.

Theorem A.8. *Consider that the average log-likelihood function $L_n(\boldsymbol{W})$ and population $L(\boldsymbol{W})$ satisfy the following conditions for $\boldsymbol{W} \in \mathcal{W}$:*

- *$\boldsymbol{W}^* = \arg\max_{\boldsymbol{W} \in \mathcal{W}} L(\boldsymbol{W})$ is identifiable from the data.*
- *$\mathcal{W}$ is a compact space.*
- *The data $\boldsymbol{\mathcal{X}}_i$ are independent and identically distributed.*
- *$\mathcal{L}(\boldsymbol{\mathcal{X}}_i; \boldsymbol{W})$ is continuous at each $\boldsymbol{W} \in \mathcal{W}$ with probability 1.*
- *$\mathbb{E}_{\boldsymbol{W}^*}[\sup_{\boldsymbol{W} \in \mathcal{W}} |\mathcal{L}(\boldsymbol{W}; \boldsymbol{\mathcal{X}})|] < \infty$.*

Then, if the maximum of $L_n(\boldsymbol{W}) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(\boldsymbol{\mathcal{X}}_i; \boldsymbol{W})$ is achieved at $\widehat{\boldsymbol{W}}_n$ then $\widehat{\boldsymbol{W}}_n$ converges uniformly to $\boldsymbol{W}^*$.

Proof. We repeat the proof of (Theorem 2.1, Newey and McFadden [1994]) for our scenario. From the identifiability assumption, Lemma A.6 implies that $\boldsymbol{W}^*$ is the unique and global maximizer of $L(\boldsymbol{W})$. Also, if we set $D(\boldsymbol{W}) = \sup_{\boldsymbol{W}\in\mathcal{W}} |\mathcal{L}(\boldsymbol{W};\boldsymbol{\mathcal{X}})|$, then the conditions of Lemma A.7 are satisfied and therefore $L_n(\boldsymbol{W})$ converges uniformly in probability to $L(\boldsymbol{W})$. We will leverage the compactness of the space $\mathcal{W}$ to show that their maxima satisfy

$$\widehat{\boldsymbol{W}}_n \xrightarrow{p} \boldsymbol{W}^* \tag{24}$$

From the uniform convergence it follows that with probability approaching 1 for any ϵ (or $\epsilon/3$ as we use next):

$$|L_n(\boldsymbol{W}) - L(\boldsymbol{W})| < \epsilon \Leftrightarrow L(\boldsymbol{W}) - \epsilon < L_n(\boldsymbol{W}) < L(\boldsymbol{W}) + \epsilon,\ \forall \boldsymbol{W} \in \mathcal{W}. \tag{25}$$

Since by definition $L_n(\boldsymbol{W})$ is a continuous function and $\mathcal{W}$ is compact it takes a maximum value at point $\widehat{\boldsymbol{W}}_n$. Since $L_n\left(\widehat{\boldsymbol{W}}_n\right) \geq L_n(\boldsymbol{W}^*)$ the maximum would satisfy for any $\epsilon > 0$

$$L_n\left(\widehat{\boldsymbol{W}}_n\right) > L_n(\boldsymbol{W}^*) - \epsilon/3. \tag{26}$$

This in combination with (25) would imply

$$L\left(\widehat{\boldsymbol{W}}_n\right) > L_n\left(\widehat{\boldsymbol{W}}_n\right) - \epsilon/3 > L_n(\boldsymbol{W}^*) - 2\epsilon/3 > L(\boldsymbol{W}^*) - \epsilon. \tag{27}$$

In essence we have proved that $L\left(\widehat{\boldsymbol{W}}_n\right)$ can get arbitrarily close to $L(\boldsymbol{W}^*)$. This in turn gives that $\widehat{\boldsymbol{W}}_n$ approaches $\boldsymbol{W}^*$ with probability 1 as $n \to \infty$. Indeed, if we consider any open interval $\mathcal{I}$ containing $\boldsymbol{W}^*$, then $\mathcal{W} \cap \mathcal{I}^c$ is compact and we can compute

$$M = \sup_{\boldsymbol{W}\in\mathcal{W}\cap\mathcal{I}^c} L(\boldsymbol{W}) < L(\boldsymbol{W}^*) \tag{28}$$

Note that by Lemma A.7 $L(\boldsymbol{W})$ is continuous, so the supremum is a finite value. If we choose $\epsilon = L(\boldsymbol{W}^*) - M$ then:

$$L\left(\widehat{\boldsymbol{W}}_n\right) > L(\boldsymbol{W}^*) - \epsilon = M \tag{29}$$

Thus $\widehat{\boldsymbol{W}}_n \in \mathcal{I}$ which concludes the proof. □

A.5 MLE CONSISTENCY FOR DAGS

We will now show that the MLE computed at (21) satisfies the requirements of Theorem A.8 for consistency. Practically, this result implies that as the amount of available data $\boldsymbol{\mathcal{X}}$ increases, the maximizer $\widehat{\boldsymbol{W}}$ of the log-likelihood function $\mathcal{L}(\boldsymbol{W}, \beta; \boldsymbol{\mathcal{X}})$ converges to the maximizer $\boldsymbol{W}^*$ of the population likelihood $L(\boldsymbol{W}, \beta)$. To begin with we introduce the following useful lemma. In essence, using the continuous characterization of acyclicity [Zheng et al., 2018] we show that the space of bounded DAGs is also closed and thus compact.

Lemma A.9. *The set of acyclic matrices* $\mathcal{A} = \left\{\boldsymbol{A} \in [-1,1]^{d\times d} | \boldsymbol{A} \text{ is acyclic}\right\}$ *is compact.*

Proof. Note that Zheng et al. [2018] proved that

$$\boldsymbol{A} \text{ is acyclic} \Leftrightarrow h(\boldsymbol{A}) = 0, \tag{30}$$

where $h(\boldsymbol{A}) = e^{\boldsymbol{A}\odot\boldsymbol{A}} - d$ is a continuous function. We proceed by showing that $\mathcal{A}$ is closed and bounded.

- **Closed:** $[-1,1]^{d\times d}$ is closed and since $h(\boldsymbol{A})$ is continuous and $\{\boldsymbol{A} \text{ is acyclic}\} = h^{-1}(\{0\})$ implies that $\mathcal{A}$ is closed [Sutherland, 2009].
- **Bounded:** $\mathcal{A}$ is bounded because $\mathcal{A} \subset [-1,1]^{d\times d}$ which is bounded.

Therefore, since $\mathcal{A} \subset \mathbb{R}^{d\times d}$ is closed and bounded, $\mathcal{A}$ is compact [Sutherland, 2009]. □

Now using this Lemma we are ready to prove our consistency result.

Theorem A.10. *The maximum log-likelihood estimator of (21) satisfies the conditions of Theorem A.8 and thus is consistent under the following assumptions:*

- *The space of window graphs $\mathcal{W} \subseteq [-1,1]^{d(k+1)\times d}$ is bounded and $\boldsymbol{B}_0$ is acyclic.*
- *The Laplacian parameter $\beta \in [a,b]$ is bounded with lower bound $a > \frac{1}{dT}$*[1].
- *The time-series samples $\boldsymbol{\mathcal{X}}_i$ are independent and identically distributed.*

Proof. We check one-by-one the requirements of Theorem A.8.

First, the identifiability of the ground truth $\boldsymbol{W}^*$ and β^* follows from Theorem A.2.

Also, $(\boldsymbol{W}, \beta) \in \mathcal{W} \times [a,b] = \mathcal{A} \times [-1,1]^{dk\times d} \times [a,b]$ which is compact because the space $\mathcal{A}$ of acyclic graphs $\boldsymbol{B}_0$ is compact from Lemma A.9 and $[-1,1]^{dk\times d}$ and $[a,b]$ are both closed and bounded and thus compact according to Sutherland [2009].

Moreover, the log-likelihood

$$\mathcal{L}(\boldsymbol{W}, \beta; \boldsymbol{\mathcal{X}}) = NT \log|\det(\boldsymbol{I} - \boldsymbol{B}_0)| - NTd\log(2\beta) - \frac{1}{\beta}\left\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}_{\text{past}}\boldsymbol{W}\right\|_1 \tag{31}$$

is continuous at $(\boldsymbol{W}, \beta)$.

Finally, we need to show that $\mathbb{E}\left[\sup_{\boldsymbol{W}\in\mathcal{W}} |\mathcal{L}(\boldsymbol{W}; \boldsymbol{\mathcal{X}})|\right] < \infty$. For this we compute:

$$\begin{aligned}
|\mathcal{L}(\boldsymbol{W}; \boldsymbol{\mathcal{X}})| &= |\log f_X(\boldsymbol{\mathcal{X}}|\boldsymbol{W}, \beta)| \\
&= \left| NT \log|\det(\boldsymbol{I} - \boldsymbol{B}_0)| - NTd\log(2\beta) - \frac{1}{\beta}\left\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}_{\text{past}}\boldsymbol{W}\right\|_1 \right| \\
&= \left| -NTd\log(2\beta) - \frac{1}{\beta}\left\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}_{\text{past}}\boldsymbol{W}\right\|_1 \right| \\
&\leq |NTd\log(2b)| + \left| \frac{1}{\beta}\left\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}_{\text{past}}\boldsymbol{W}\right\|_1 \right| \\
&\leq C_1 + \frac{1}{a}\left\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}_{\text{past}}\boldsymbol{W}\right\|_1 \\
&\leq C_1 + C_2\left\|\boldsymbol{\mathcal{X}}\right\|_1
\end{aligned}$$

Here we used that $\boldsymbol{B}_0$ is acyclic and thus $NT\log|\det(\boldsymbol{I} - \boldsymbol{B}_0)| = 0$. We assumed that $\beta \in [a,b]$ is bounded. Also we used that the (τ, j) entry of $\boldsymbol{X} - \boldsymbol{X}_{\text{past}}\boldsymbol{W}$ is $\boldsymbol{X}_{\tau,j} - \boldsymbol{X}_{\text{past}\tau,:}\boldsymbol{W}_{:,j}$ and

$$\begin{aligned}
&|\boldsymbol{X}_{\tau,j} - \boldsymbol{X}_{\text{past}\tau,:}\boldsymbol{W}_{:,j}| < |\boldsymbol{X}_{\tau,j}| + \left\|\boldsymbol{X}_{\text{past}\tau,:}\right\|_1 \Rightarrow \\
&\sum_{\tau,j} |\boldsymbol{X}_{\tau,j} - \boldsymbol{X}_{\text{past}\tau,:}\boldsymbol{W}_{:,j}| < \sum_{\tau,j}((k+1)d+1)\,|\boldsymbol{X}_{\tau,j}| = ((k+1)d+1)\left\|\boldsymbol{X}\right\|_1,
\end{aligned}$$

which furthermore implies

$$\left\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}_{\text{past}}\boldsymbol{W}\right\|_1 = \sum_i |\boldsymbol{\mathcal{X}}_i - \boldsymbol{\mathcal{X}}_{i,\text{past}}\boldsymbol{W}| < \sum_i ((k+1)d+1)\left\|\boldsymbol{\mathcal{X}}_i\right\|_1 = ((k+1)d+1)\left\|\boldsymbol{\mathcal{X}}\right\|_1 = C_2\left\|\boldsymbol{\mathcal{X}}\right\|_1, \tag{32}$$

[1] This value in our experiment is at most $\frac{1}{2\cdot 10^4}$, so this is a mild assumption.

for some constant C_2. Therefore:

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{W}^*}\left[\left|\mathcal{L}\left(\boldsymbol{W};\boldsymbol{\mathcal{X}}\right)\right|\right] &= \int_{\boldsymbol{\mathcal{X}}\in\mathbb{R}^{T\times d}} \left|\mathcal{L}\left(\boldsymbol{W};\boldsymbol{\mathcal{X}}\right)\right| f_X\left(\boldsymbol{\mathcal{X}}|\boldsymbol{W}^*,\beta^*\right) d\boldsymbol{\mathcal{X}} \\
&< \int_{\boldsymbol{\mathcal{X}}\in\mathbb{R}^{N\times T\times d}} \left(C_1 + C_2\left\|\boldsymbol{\mathcal{X}}\right\|_1\right) f_X\left(\boldsymbol{\mathcal{X}}|\boldsymbol{W}^*,\beta^*\right) d\boldsymbol{\mathcal{X}} \\
&= \int_{\boldsymbol{\mathcal{X}}\in\mathbb{R}^{N\times T\times d}} \left(C_1 + C_2\left\|\boldsymbol{\mathcal{X}}\right\|_1\right) \left|\det\left(\boldsymbol{I}-\boldsymbol{B}_0^*\right)\right|^{NT} \frac{1}{\left(2\beta^*\right)^{NdT}} e^{-\frac{\left\|\boldsymbol{\mathcal{X}}-\boldsymbol{\mathcal{X}}_{\text{past}}\boldsymbol{W}^*\right\|_1}{\beta^*}} d\boldsymbol{\mathcal{X}} \\
&= \int_{\boldsymbol{\mathcal{X}}\in\mathbb{R}^{N\times T\times d}} \left(C_1 + C_2\left\|\boldsymbol{\mathcal{X}}\right\|_1\right) \frac{1}{\left(2\beta^*\right)^{NdT}} e^{-\frac{\left\|\boldsymbol{\mathcal{X}}-\boldsymbol{\mathcal{X}}_{\text{past}}\boldsymbol{W}^*\right\|_1}{\beta^*}} \left|\det\left(\boldsymbol{I}-\boldsymbol{A}^*\right)\right| d\boldsymbol{\mathcal{X}} \\
&= \int_{\boldsymbol{\mathcal{S}}\in\mathbb{R}^{N\times T\times d}} \left(C_1 + C_2\left\|\boldsymbol{\mathcal{X}}\right\|_1\right) \frac{1}{\left(2\beta^*\right)^{NdT}} e^{-\frac{\left\|\boldsymbol{\mathcal{S}}\right\|_1}{\beta^*}} d\boldsymbol{\mathcal{S}} \\
&= C_1 + C_2 \int_{\boldsymbol{\mathcal{S}}\in\mathbb{R}^{N\times T\times d}} \left\|\boldsymbol{\mathcal{X}}\right\|_1 \frac{1}{\left(2\beta^*\right)^{NdT}} e^{-\frac{\left\|\boldsymbol{\mathcal{S}}\right\|_1}{\beta^*}} d\boldsymbol{\mathcal{S}} \\
&= C_1 + C_2 \int_{\boldsymbol{\mathcal{S}}\in\mathbb{R}^{N\times T\times d}} \left\|\boldsymbol{\mathcal{S}}\left(\boldsymbol{I}-\boldsymbol{A}^*\right)^{-1}\right\|_1 \frac{1}{\left(2\beta^*\right)^{NdT}} e^{-\frac{\left\|\boldsymbol{\mathcal{S}}\right\|_1}{\beta^*}} d\boldsymbol{\mathcal{S}}
\end{aligned}$$

Note that, $\left\|\boldsymbol{\mathcal{S}}\left(\boldsymbol{I}-\boldsymbol{A}^*\right)^{-1}\right\|_1 = \left\|\boldsymbol{\mathcal{S}}\left(\boldsymbol{I}+\boldsymbol{A}^*+...+\left(\boldsymbol{A}^*\right)^{dT}\right)\right\|_1$. From Lemma A.3 we have that

$$\left\|\boldsymbol{\mathcal{S}}\left(\boldsymbol{I}-\boldsymbol{A}^*\right)^{-1}\right\|_1 = \left\|\boldsymbol{\mathcal{S}}\left(\boldsymbol{I}+\boldsymbol{A}^*+...+\left(\boldsymbol{A}^*\right)^{dT}\right)\right\|_1 \leq \left\|\boldsymbol{\mathcal{S}}\right\|_1 \left(dT + \left\|\boldsymbol{A}^*\right\|_1 + ... + \left\|\left(\boldsymbol{A}^*\right)\right\|_1^{dT}\right) \leq \left\|\boldsymbol{\mathcal{S}}\right\|_1 \cdot C_3 \quad (33)$$

Thus:

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{W}^*}\left[\left|\mathcal{L}\left(\boldsymbol{W};\boldsymbol{\mathcal{X}}\right)\right|\right] &< C_1 + C_2 \int_{\boldsymbol{\mathcal{S}}\in\mathbb{R}^{N\times T\times d}} \left\|\boldsymbol{\mathcal{S}}\left(\boldsymbol{I}-\boldsymbol{A}^*\right)^{-1}\right\|_1 \frac{1}{\left(2\beta^*\right)^{NdT}} e^{-\frac{\left\|\boldsymbol{\mathcal{S}}\right\|_1}{\beta^*}} d\boldsymbol{\mathcal{S}} \\
&< C_1 + C_2 C_3 \int_{\boldsymbol{\mathcal{S}}\in\mathbb{R}^{N\times T\times d}} \left\|\boldsymbol{\mathcal{S}}\right\|_1 \frac{1}{\left(2\beta^*\right)^{NdT}} e^{-\frac{\left\|\boldsymbol{\mathcal{S}}\right\|_1}{\beta^*}} d\boldsymbol{\mathcal{S}} \\
&= C_1 + C_2 C_3 \sum_{i,\tau,j} \int_{\boldsymbol{\mathcal{S}}\in\mathbb{R}^{N\times T\times d}} \left|\boldsymbol{\mathcal{S}}_{i,\tau,j}\right| \frac{1}{\left(2\beta^*\right)^{NdT}} e^{-\frac{\left\|\boldsymbol{\mathcal{S}}\right\|_1}{\beta^*}} d\boldsymbol{\mathcal{S}} \\
&= C_1 + C_2 C_3 \sum_{i,\tau,j} \int_{\mathbb{R}} \left|\boldsymbol{\mathcal{S}}_{i,\tau,j}\right| \frac{1}{\left(2\beta^*\right)} e^{-\frac{\left|\boldsymbol{\mathcal{S}}_{i,\tau,j}\right|}{\beta^*}} d\boldsymbol{\mathcal{S}}_{i,\tau,j} \\
&= C_1 + 2C_2 C_3 \sum_{i,\tau,j} \int_{\mathbb{R}_{\geq 0}} \left|\boldsymbol{\mathcal{S}}_{i,\tau,j}\right| \frac{1}{\left(2\beta^*\right)} e^{-\frac{\left|\boldsymbol{\mathcal{S}}_{i,\tau,j}\right|}{\beta^*}} d\boldsymbol{\mathcal{S}}_{i,\tau,j} \\
&= C_1 + 2C_2 C_3 \sum_{i,\tau,j} \int_{\mathbb{R}_{\geq 0}} \boldsymbol{\mathcal{S}}_{i,\tau,j} \frac{1}{\left(2\beta^*\right)} e^{-\frac{\boldsymbol{\mathcal{S}}_{i,\tau,j}}{\beta^*}} d\boldsymbol{\mathcal{S}}_{i,\tau,j} \\
&= C_1 + 2C_2 C_3 \sum_{i,\tau,j} \left\{ -\frac{\boldsymbol{\mathcal{S}}_{i,\tau,j}}{2} e^{-\frac{\boldsymbol{\mathcal{S}}_{i,\tau,j}}{\beta^*}}\Big|_0^\infty - \int_{\mathbb{R}_{\geq 0}} -\frac{1}{2} e^{-\frac{\boldsymbol{\mathcal{S}}_{i,\tau,j}}{\beta^*}} d\boldsymbol{\mathcal{S}}_{i,\tau,j} \right\} \\
&= const < \infty.
\end{aligned}$$

□

Remark A.11. Note that LiNGAM identifiability is true in the entire space of real matrices $\mathbb{R}^{d(k+1)\times d}$ [Shimizu et al., 2006, Ng et al., 2020]. In other words for any $\boldsymbol{W} \in \mathbb{R}^{d(k+1)\times d}$ different from the ground truth DAG $\boldsymbol{W}^*$ the distribution $f_X(\boldsymbol{X}|\boldsymbol{W},\beta)$ induced by $\boldsymbol{W}$ is different from that of $\boldsymbol{W}^*$, namely $f_X(\boldsymbol{X}|\boldsymbol{W}^*,\beta)$. The reason we restrict our search space to be a DAG is to constrain the magnitude of the terms of the MLE that contain $\boldsymbol{B}_0$.

A.6 SPINSVAR OPTIMIZATION DERIVATION

Here we derive the optimization objective of SpinSVAR for approximating the ground truth window graph parameters of the SVAR of (3), given that the SVAR input $\boldsymbol{\mathcal{S}}$ entries are distributed independently according to Laplace$(0,\beta^*)$. We consider

N realization of time series $\boldsymbol{X}$ collected in a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{N\times T\times d}$. According to (21) the log-likelihood of the data $\boldsymbol{\mathcal{X}}$ is

$$\mathcal{L}\left(\boldsymbol{W},\beta;\boldsymbol{\mathcal{X}}\right) = \log f_X\left(\boldsymbol{\mathcal{X}}|\boldsymbol{W},\beta\right) = \log\prod f_X\left(\boldsymbol{X}_i|\boldsymbol{W},\beta\right) \tag{34}$$

$$= \sum_{i=1}^{N}\log f_X\left(\boldsymbol{X}_i|\boldsymbol{W},\beta\right) \tag{35}$$

$$= NT\log\left|\det\left(\boldsymbol{I}-\boldsymbol{B}_0\right)\right| - NTd\log(2\beta) - \frac{1}{\beta}\left\|\boldsymbol{\mathcal{X}}-\boldsymbol{\mathcal{X}}_{\text{past}}\boldsymbol{W}\right\|_1 \tag{36}$$

To maximize the log-likelihood with respect to β we solve:

$$\frac{\partial\mathcal{L}}{\partial\beta} = 0 \Leftrightarrow -\frac{NTd}{\beta} + \frac{1}{\beta^2}\left\|\boldsymbol{\mathcal{X}}-\boldsymbol{\mathcal{X}}_{\text{past}}\boldsymbol{W}\right\|_1 = 0 \Leftrightarrow \beta = \frac{1}{NTd}\left\|\boldsymbol{\mathcal{X}}-\boldsymbol{\mathcal{X}}_{\text{past}}\boldsymbol{W}\right\|_1. \tag{37}$$

Note that if $\boldsymbol{W} = \boldsymbol{W}^*$ this is a reasonable value for β as on expectation

$$\mathbb{E}_{\beta^*}\left[\left\|\boldsymbol{\mathcal{X}}-\boldsymbol{\mathcal{X}}_{\text{past}}\boldsymbol{W}\right\|_1\right] = \mathbb{E}_{\beta^*}\left[\left\|\boldsymbol{\mathcal{S}}\right\|_1\right] = \sum_{i,\tau,j}\mathbb{E}_{\beta^*}\left[\left\|\boldsymbol{\mathcal{S}}_{i,\tau,j}\right\|_1\right] = NTd\beta^*. \tag{38}$$

Moreover,

$$\frac{\partial^2\mathcal{L}}{\partial\beta^2} = 0 \Leftrightarrow \frac{NTd}{\beta^2} - \frac{2}{\beta^3}\left\|\boldsymbol{\mathcal{X}}-\boldsymbol{\mathcal{X}}_{\text{past}}\boldsymbol{W}\right\|_1 = \frac{NTd}{\beta^3}\left(\beta - \frac{2}{NTd}\left\|\boldsymbol{\mathcal{X}}-\boldsymbol{\mathcal{X}}_{\text{past}}\boldsymbol{W}\right\|_1\right) < 0. \tag{39}$$

So, $\mathcal{L}\left(\boldsymbol{W},\beta;\boldsymbol{\mathcal{X}}\right)$ is locally concave at $\beta = \frac{1}{NTd}\left\|\boldsymbol{\mathcal{X}}-\boldsymbol{\mathcal{X}}_{\text{past}}\boldsymbol{W}\right\|_1$, which gives a local maximum. Similarly to Ng et al. [2020], we profile out the parameter β using its approximation $\widehat{\beta} = \frac{1}{NTd}\left\|\boldsymbol{\mathcal{X}}-\boldsymbol{\mathcal{X}}_{\text{past}}\boldsymbol{W}\right\|_1$ to formulate a log-likelihood maximization problem for approximating $\boldsymbol{W}$:

$$\mathcal{L}\left(\boldsymbol{W},\widehat{\beta};\boldsymbol{\mathcal{X}}\right) = NT\log\left|\det\left(\boldsymbol{I}-\boldsymbol{B}_0\right)\right| - NTd\log\left(\left\|\boldsymbol{\mathcal{X}}-\boldsymbol{\mathcal{X}}_{\text{past}}\boldsymbol{W}\right\|_1\right) + \text{const.}$$

The window graph $\boldsymbol{W}^*$ is then be approximated as:

$$\widehat{\boldsymbol{W}} = \underset{\boldsymbol{W}\in\mathcal{W}}{\arg\max}\,\mathcal{L}\left(\boldsymbol{W};\boldsymbol{\mathcal{X}}\right) = \underset{\boldsymbol{W}\in\mathcal{W}}{\arg\max}\left\{NT\log\left|\det\left(\boldsymbol{I}-\boldsymbol{B}_0\right)\right| - NTd\log\left(\left\|\boldsymbol{\mathcal{X}}-\boldsymbol{\mathcal{X}}_{\text{past}}\boldsymbol{W}\right\|_1\right) + \text{const}\right\}$$

$$= \underset{\boldsymbol{W}\in\mathcal{W}}{\arg\min}\left\{d\log\left(\left\|\boldsymbol{\mathcal{X}}-\boldsymbol{\mathcal{X}}_{\text{past}}\boldsymbol{W}\right\|_1\right) - \log\left|\det\left(\boldsymbol{I}-\boldsymbol{B}_0\right)\right|\right\}$$

$$= \underset{\boldsymbol{W}\in\mathcal{W}}{\arg\min}\log\left\|\boldsymbol{\mathcal{X}}-\boldsymbol{\mathcal{X}}_{\text{past}}\boldsymbol{W}\right\|_1 - \frac{1}{d}\log\left|\det\left(\boldsymbol{I}-\boldsymbol{B}_0\right)\right| \tag{40}$$

In practice, searching for the minimum of (40) over the space of DAGs is computationally inefficient. Following Ng et al. [2020] we use the acyclicity as a soft constraint, i.e. a regularizer. This simplifies the optimization algorithm without compromising performance, as will be shown in our experiments. The final optimization of SpinSVAR is the following.

$$\widetilde{\boldsymbol{W}} = \underset{\boldsymbol{W}\in\mathbb{R}^{d(k+1)\times d}}{\arg\min}\log\left\|\boldsymbol{\mathcal{X}}-\boldsymbol{\mathcal{X}}_{\text{past}}\boldsymbol{W}\right\|_1 - \frac{1}{d}\log\left|\det\left(\boldsymbol{I}-\boldsymbol{B}_0\right)\right| + \lambda_1\cdot\left\|\boldsymbol{W}\right\|_1 + \lambda_2\cdot h\left(\boldsymbol{B}_0\right). \tag{41}$$

B APPLYING SPARSERC TO TIME-SERIES DATA

SparseRC [Misiakos et al., 2023] is designed to learn a DAG from static data. Misiakos et al. [2024] applied SparseRC to learn graphs from time-series data by exploiting the structure of the unrolled DAG corresponding to the time series. For long time series, such a formulation creates a huge DAG to be learned - ranging from 20 thousand to 1 million nodes in our experiments. However, SparseRC can only be executed for ≈ 5000 nodes at maximum to terminate in a reasonable time [Misiakos et al., 2023]. Thus it is impossible to be applied in our scenario in its prior form. For this reason, we propose an alternative way to apply SparseRC, which however, comes with a cost in approximation performance.

SVAR as a Linear SEM To start with we show how an SVAR can be written as a linear structural equation model (SEM), which is the analogous model for generating linear static DAG data. We consider a time series $\boldsymbol{X}$ generated with the SVAR in (3) (noiseless for simplicity). Consider the single-row vector $\boldsymbol{x} = (\boldsymbol{x}_0, \boldsymbol{x}_1, ..., \boldsymbol{x}_{T-1}) \in \mathbb{R}^{1\times dT}$ consisting of the concatenation of the time-series vectors $\boldsymbol{x}_0, \boldsymbol{x}_1, ..., \boldsymbol{x}_{T-1}$ along the first dimension. Then (3) can also be encoded as:

$$\boldsymbol{x} = \boldsymbol{x}\boldsymbol{A} + \boldsymbol{s}, \tag{42}$$

where the structural shocks $\boldsymbol{s}$ here also have dimension $1 \times dT$. The matrix $\boldsymbol{A}$ is the adjacency matrix of a DAG with a special structure called the *unrolled DAG* [Kim and Anderson, 2012], which occurs by repeating the window graph corresponding to (2) for every time step $t \in [T]$:

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{B}_0 & \boldsymbol{B}_1 & \dots & \boldsymbol{B}_k & \dots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{B}_0 & \boldsymbol{B}_1 & & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & & \boldsymbol{B}_k \\ & & & \ddots & \ddots & \vdots \\ \boldsymbol{0} & & \dots & \boldsymbol{0} & \boldsymbol{B}_0 & \boldsymbol{B}_1 \\ \boldsymbol{0} & \boldsymbol{0} & & \dots & \boldsymbol{0} & \boldsymbol{B}_0 \end{pmatrix}. \tag{43}$$

This allows us to rewrite (3) as a linear structural equation model (SEM) [Shimizu et al., 2006]:

$$\widetilde{\boldsymbol{X}} = \widetilde{\boldsymbol{X}}\boldsymbol{A} + \widetilde{\boldsymbol{S}}, \tag{44}$$

where $\widetilde{\boldsymbol{X}} \in \mathbb{R}^{N\times dT}$ consists of the N time series as rows and $\widetilde{\boldsymbol{S}}$ is defined similarly for the structural shocks. Since $\boldsymbol{A}$ is a DAG, (44) represents a linear SEM.

Original SparseRC We now explain how SparseRC can be applied to learn the window graph from time series according to Misiakos et al. [2024]. SparseRC can be used to learn $\boldsymbol{A}$ from (many samples of) $\boldsymbol{x}$ stacked as a matrix $\widetilde{\boldsymbol{X}} \in \mathbb{R}^{N\times dT}$, generated from a linear SEM (44). Its optimization objective aims to minimize the number of approximated non-zero structural shocks $\widetilde{\boldsymbol{S}}$ in (44). This is expressed with the following discrete optimization problem

$$\widehat{\boldsymbol{A}} = \underset{\boldsymbol{A}\in\mathbb{R}^{dT\times dT}}{\arg\min} \left\| \widetilde{\boldsymbol{X}} - \widetilde{\boldsymbol{X}}\boldsymbol{A} \right\|_0, \quad \text{s.t. } \boldsymbol{A} \text{ is acyclic.} \tag{45}$$

The window graph $\widehat{\boldsymbol{W}}$ can be then extracted from the first row of the approximated $\widehat{\boldsymbol{A}}$. SparseRC in practice uses a continuous relaxation to solve optimization problem (45), but here we keep the discrete formulation for simplicity.

It can be seen that the DAG $\boldsymbol{A}$ consists of dT nodes. In our smallest experiment this equals to $20 \times 1000 = 20000$ nodes, which is already out of reach for SparseRC. In contrast, SpinSVAR requires to learn only $(k+1)\times$ DAGs with d nodes each. Thus, we necessarily need to formulate SparseRC differently to be able to compare against it.

Modified SparseRC The idea is to reduce the size of $\boldsymbol{A}$ by getting rid of the $\boldsymbol{0}'$s in (43). Specifically, instead of feeding SparseRC $\widetilde{\boldsymbol{X}}$ we feed as input $\boldsymbol{\mathcal{X}}_{\text{past}}$. The resulting algorithm aims to find an $\widehat{\boldsymbol{A}}$ according to:

$$\widehat{\boldsymbol{A}} = \underset{\boldsymbol{A}\in\mathbb{R}^{(k+1)d\times(k+1)d}}{\arg\min} \left\| \boldsymbol{\mathcal{X}}_{\text{past}} - \boldsymbol{\mathcal{X}}_{\text{past}}\boldsymbol{A} \right\|_0, \quad \text{s.t. } \boldsymbol{A} \text{ is acyclic.} \tag{46}$$

To be compatible with the data-generating process, the following structure is assumed for $\boldsymbol{A}$:

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{B}_0 & \boldsymbol{0} & ... & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{B}_1 & \boldsymbol{B}_0 & \ddots & & \boldsymbol{0} \\ \vdots & \boldsymbol{B}_1 & \ddots & \ddots & \vdots \\ \boldsymbol{B}_{k-1} & & \ddots & \boldsymbol{B}_0 & \boldsymbol{0} \\ \boldsymbol{B}_k & \boldsymbol{B}_{k-1} & \dots & \boldsymbol{B}_1 & \boldsymbol{B}_0, \end{pmatrix} \tag{47}$$

The optimization objective (46) is different from $\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}_{\text{past}}\boldsymbol{W}\|_0$ used from SpinSVAR and promotes a different convention in the data generating process. In particular by setting $\widetilde{\boldsymbol{\mathcal{S}}} = \boldsymbol{\mathcal{X}}_{\text{past}} - \boldsymbol{\mathcal{X}}_{\text{past}}\boldsymbol{A}$ the structural shock $\widetilde{\boldsymbol{s}}_{t-j}$ corresponding to the position j of row t of $\boldsymbol{x}_{t,\text{past}} = (\boldsymbol{x}_t, \boldsymbol{x}_{t-1}, ..., \boldsymbol{x}_{t-j}, ..., \boldsymbol{x}_{t-k})$ of a sample i of $\boldsymbol{\mathcal{X}}_{\text{past}}$ would be:

$$\widetilde{\boldsymbol{s}}_{t-j} = \boldsymbol{x}_{t-j} - \boldsymbol{x}_{t-j}\boldsymbol{B}_0 + \boldsymbol{x}_{t-j-1}\boldsymbol{B}_1 + ... + \boldsymbol{x}_{t-k}\boldsymbol{B}_{k-j} \neq \boldsymbol{s}_{t-j}. \tag{48}$$

This implies that the approximation of the structural shocks is not consistent with the data generation in (3), except when $j = 0$. Thus, only the first column of $\boldsymbol{A}$ promotes the correct equations and the rest undermine the performance of SparseRC. Resolving this discrepancy and keeping only the first column as trainable parameters is among the technical contributions of our paper.

C SPINSVAR OPTIMIZATION AND COMPARISON WITH BASELINES

SpinSVAR Our implementation in PyTorch is outlined in Algorithm 1. It parametrizes the window graph matrix $\boldsymbol{W}$ using a single PyTorch linear layer and optimizes the objective function (11) with the Adam optimizer. The overall computational complexity of the algorithm is:

$$\mathcal{O}\left(M \cdot (NTd^2k + d^3)\right), \tag{49}$$

where M is the total number of epochs (up to 10^4).

The primary term in our objective, $N\left\{\log \|\boldsymbol{\mathcal{X}} - L(\boldsymbol{\mathcal{X}})\|_1 - \frac{1}{d}\log|\det(\boldsymbol{I} - \boldsymbol{B}_0)|\right\}$, represents a fundamental difference from prior work on causal discovery in time series. Methods such as VAR-based optimization approaches [Pamfil et al., 2020, Sun et al., 2023] typically rely on a mean-squared error loss supplemented by an L^1 penalty to promote sparsity in the DAG. In contrast, both the main term and the regularizer in our objective are L^1 norms, promoting sparsity not only in the DAG but also in the SVAR input. This design aligns with the assumption of sparse SVAR input. Potentially, L^2 leads to longer convergence times, which makes our algorithm terminate faster in the experiments.

Algorithm 1 SpinSVAR: DAG Learning from Time Series with Few structural shocks

Input: Time series data tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{N\times T\times d}$, λ_1, λ_2 regularization parameters and threshold ω.

Output: Weighted window graph $\widehat{\boldsymbol{W}} = \begin{pmatrix} \boldsymbol{B}_0 \\ \vdots \\ \boldsymbol{B}_k \end{pmatrix}$ and structural shocks $\widehat{\boldsymbol{\mathcal{S}}}$.

Initialize:
A single linear layer L(input: $d(k+1)$, output: d) in PyTorch that represents $\widehat{\boldsymbol{W}}$.
Tensor $\boldsymbol{\mathcal{X}}_{\text{past}} \in \mathbb{R}^{N\times T\times d(k+1)}$, where the (n,t) entry is the vector $\boldsymbol{x}_{t,\text{past}} = (\boldsymbol{x}_t,\, \boldsymbol{x}_{t-1},\, ...,\, \boldsymbol{x}_{t-k}) \in \mathbb{R}^{1\times d(k+1)}$.
Iterate:
for each training epoch up to $M = 10^4$ **do**
    Compute the loss:

$$N\left\{\log \|\boldsymbol{\mathcal{X}} - L(\boldsymbol{\mathcal{X}})\|_1 - \frac{1}{d}\log|\det(\boldsymbol{I} - \boldsymbol{B}_0)|\right\} + \lambda_1\|\boldsymbol{W}\|_1 + \lambda_2 h(\boldsymbol{B}_0),$$

    where $h(\boldsymbol{B}) = \text{tr}\left(e^{\boldsymbol{B}\odot\boldsymbol{B}}\right) - d$.
    Update the linear layer parameters $\widehat{\boldsymbol{W}}$ with Adam optimizer.
    Stop early if the loss doesn't improve for 40 epochs.
end for
Post-processing:
Set the entries w_{ij} of $\boldsymbol{W}$ with $|w_{ij}| < \omega$ to zero.
Compute the unweighted version $\boldsymbol{U} \in \{0,1\}^{d(k+1)\times d}$ of $\boldsymbol{W}$.
Compute the approximated structural shocks:

$$\widehat{\boldsymbol{\mathcal{S}}} = \boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}_{\text{past}}\widehat{\boldsymbol{W}}.$$

return $\widehat{\boldsymbol{W}}, \widehat{\boldsymbol{U}}, \widehat{\boldsymbol{\mathcal{S}}}$

C.1 COMPARISON WITH BASELINES

SparseRC As we explained in the main text, the method from Misiakos et al. [2023] is infeasible to execute for long time series data. In its original form, SparseRC has complexity $\mathcal{O}\left(M \cdot (Nd^2T^2 + d^3T^3)\right)$, where M is the total

number of iterations. SparseRC learns a $dT \times dT$ unrolled DAG, which for our smaller scenario, results in a DAG with $d \times T = 20 \times 1000 = 20000$ nodes that goes beyond its computational reach [Misiakos et al., 2023].

In Appendix B, we design a modified version of SparseRC that learns a $(k+1)d \times (k+1)d$ adjacency matrix, which ultimately leads to a complexity of $\mathcal{O}\left(M \cdot (NTd^2k^2 + d^3k^3)\right)$. This adaptation can be executed in most scenarios but comes at the cost of reduced model performance.

VARLiNGAM First, the method fits a VAR model to the data:

$$\boldsymbol{x}_t = \widetilde{\boldsymbol{B}}_1 \boldsymbol{x}_{t-1} + ... + \widetilde{\boldsymbol{B}}_k \boldsymbol{x}_{t-k} + \boldsymbol{n}_t, \tag{50}$$

and then performs Independent Component Analysis (ICA) to compute the self-dependencies matrix $\boldsymbol{B}_0$:

$$\boldsymbol{n}_t = (\boldsymbol{I} - \boldsymbol{B}_0)\boldsymbol{n}_t + \boldsymbol{s}_t. \tag{51}$$

The resulting matrices are calculated as

$$\boldsymbol{B}_\tau = (\boldsymbol{I} - \boldsymbol{B}_0)\widetilde{\boldsymbol{B}}_\tau.$$

The ICA step can be replaced with Direct LiNGAM [Shimizu et al., 2011], which guarantees convergence in a finite number of steps (under certain assumptions). This variation leads to the method Directed VARLiNGAM. However, both approaches have worse complexity compared to ours:

- For Direct LiNGAM: $\mathcal{O}\left(NTd^2k + NTd^3M^2 + d^4M^3\right)$, where M is the number of iterations of Direct LiNGAM.
- For ICA LiNGAM: $\mathcal{O}\left(NTd^2k + NTd^3 + d^4\right)$, which lacks convergence guarantees.

In the large-DAG regime, these algorithms are inevitably slower than ours.

cuLiNGAM Akinwande and Kolter [2024] accelerate Directed VARLiNGAM by implementing a parallelized version on GPUs. While this method is faster than Directed VARLiNGAM, our experiments show that it still times out, likely due to high convergence times.

DYNOTEARS Here, the mean-square error (MSE) is used, transforming the optimization into a quadratic problem:

$$\frac{1}{2NT}\left\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}_{\text{past}}\boldsymbol{W}\right\|_2 + \lambda_w\|\boldsymbol{W}\|_1 + \frac{\rho}{2}h(\boldsymbol{B}_0)^2 + ah(\boldsymbol{B}_0), \tag{52}$$

where the L^2 norm in the first term doesn't enforce sparsity on the structural shocks. As a result, this method experiences longer convergence times and produces a poor approximation of the ground truth window graph.

TCDF This method fits convolutional neural networks (CNNs) to predict the time series at each node, based on the time-series values of other nodes in previous time steps. The approximation is optimized using the MSE loss. However, both the non-linearity of CNNs and the MSE loss do not align with our data generation process, which limits the method's effectiveness for our specific task.

NTS-NOTEARS Similar to TCDF, this method also uses CNNs and MSE loss to approximate the window graph. In addition, the acyclicity regularizer from NOTEARS is applied. For similar reasons, we anticipate low performance in our experiments with this method as well, due to the mismatch between the assumptions of the method and the characteristics of our data.

tsFCI, PCMCI For the constraint-based baselines, there is no clear comparison in terms of optimization. These methods rely on statistical independence tests to infer causal dependencies between nodes at different time points. Empirically, however, these methods perform poorly, likely due to their inability to determine the causal direction for every edge they discover.

D SPARSITY PROPERTIES OF LAPLACE DISTRIBUTION

A random variable X follows a Laplace distribution [Eltoft et al., 2006], denoted as Laplace(μ, β), if its probability density function is given by:

$$f_X(x|\mu, \beta) = \frac{1}{2\beta}e^{-\frac{|x-\mu|}{\beta}}. \tag{53}$$

We now analyze why the Laplace distribution is better suited for modeling sparse vectors compared to the Gaussian distribution. Specifically, we consider Laplacian noise variables centered at zero, setting $\mu = 0$. Our motivation is that Laplace-distributed variables are more likely to produce large outliers, whereas Gaussian-distributed variables tend to be concentrated around zero.

To investigate sparsity, we consider three approaches for achieving approximately 5% sparsity. The first follows our experimental procedure described in Section 5.1, which combines Bernoulli and uniform distributions. The second uses a Gaussian distribution, and the third uses a Laplace distribution. Since strict sparsity cannot be achieved, 95% of the values will be approximately zero.

We compare these distributions in terms of their sparsity-inducing properties by addressing the following question: *How much more significant are the nonzero values compared to the approximately zero ones?* To do so, we define a threshold ω that classifies values above ω as significant and those below ω as approximately zero.

For each scenario, we generate a random vector $\boldsymbol{s}$ with d entries $(s_1, \ldots, s_d)$ and consider a threshold of $\omega = 0.1$.

Bernoulli & Uniform Each s_i is generated independently, and with probability $1 - p = 0.95$, it is set to zero. Otherwise, with probability $p = 0.05$, it takes a uniform random value from the range $[-0.4, -0.1] \cup [0.1, 0.4]$. The upper bound of 0.4 ensures that the maximum absolute value is comparable to that of the Laplace distribution, as described later. To each s_i, we then add Gaussian noise with a standard deviation of 0.03. Since 99% of the Gaussian noise values lie within $[-0.09, 0.09]$, this noise does not significantly affect the sparsity structure. Thus, given $\omega = 0.1$, approximately 95% of the entries in $\boldsymbol{s}$ will have absolute values below ω, effectively maintaining sparsity.

Gaussian For a Gaussian-distributed variable, it is known that approximately 95% of values lie within $[-2\sigma, 2\sigma]$. To achieve the required sparsity threshold $\omega = 0.1$, we set the standard deviation to $\sigma = 0.05$.

Laplace For a Laplace-distributed variable X, the probability that its absolute value does not exceed ω is given by:

$$\begin{aligned}
\mathbb{P}\left(|X| \leq \omega\right) &= \int_{-\omega}^{\omega} \frac{1}{2\beta} e^{-\frac{|x|}{\beta}} dx \\
&= 2\int_{0}^{\omega} \frac{1}{2\beta} e^{-\frac{x}{\beta}} dx \\
&= \int_{0}^{\omega} \frac{1}{\beta} e^{-\frac{x}{\beta}} dx \\
&= -e^{-\frac{x}{\beta}}\Big|_{0}^{\omega} = 1 - e^{-\frac{\omega}{\beta}}.
\end{aligned}$$

Setting $\beta = \omega/3$ ensures that $\mathbb{P}\left(|X| \leq \omega\right) \approx 0.95$, thereby achieving the desired sparsity.

Empirical Evaluation With the distribution parameters set, we empirically evaluate the sparsity patterns of the generated vectors. Our goal is to demonstrate that the Laplace distribution is better suited for generating sparse vectors compared to the Gaussian. For each distribution listed in Table 2, we generate a vector $\boldsymbol{s}$ with $d = 10^6$ entries and compute the following evaluation metrics:

- **Sparsity fraction**: The percentage of values with absolute values greater than ω.
- **Maximum absolute value**: $\max_i |c_i|$.
- **Contrast ratio**:

$$\text{Contrast ratio} = \frac{\frac{1}{M}\sum_{|c_i|>\omega} |c_i|}{\omega}, \tag{54}$$

  where M is the number of entries satisfying $|c_i| > \omega$.
- **Signal-to-noise ratio (SNR)**:

$$\text{SNR} = \frac{\sum_{|c_i|>\omega} |c_i|^2}{\sum_{|c_i|<\omega} |c_i|^2}. \tag{55}$$

The computational results are shown in Table 2. Given the 5% sparsity constraint, the best performance is achieved by the Bernoulli-Uniform method, which produces higher values with a maximum magnitude of 0.51 and exhibits a

superior contrast ratio and SNR, indicating better sparsity characteristics. The Laplace distribution achieves the second-best performance.

Table 2: Empirical sparsity evaluation for different distributions.

Method	Sparsity	Maximum Absolute Value	Contrast Ratio	SNR
Bernoulli & Uniform	5.0%	0.51	2.50	3.40
Gauss $\mathcal{N}(0, 0.051^2)$	5.0%	0.25	1.19	0.38
Laplace$\left(0, \frac{1}{3}\right)$	5.0%	0.53	1.33	0.73

E ADDITIONAL EXPERIMENTS

In this section we include additional synthetic experiments and additional results regarding the simulated and real-world financial datasets.

E.1 EMPIRICAL STABILITY OF TIME SERIES

According to Theorem A.1, the stability of the time-series data $\boldsymbol{X}$ requires that the weight matrices $\boldsymbol{B}_0, \boldsymbol{B}_1, \ldots, \boldsymbol{B}_k$ satisfy an upper bound w such that:

$$(5 + 2 + 2)w = 9w < 1, \quad \text{or equivalently,} \quad w < 0.11. \tag{56}$$

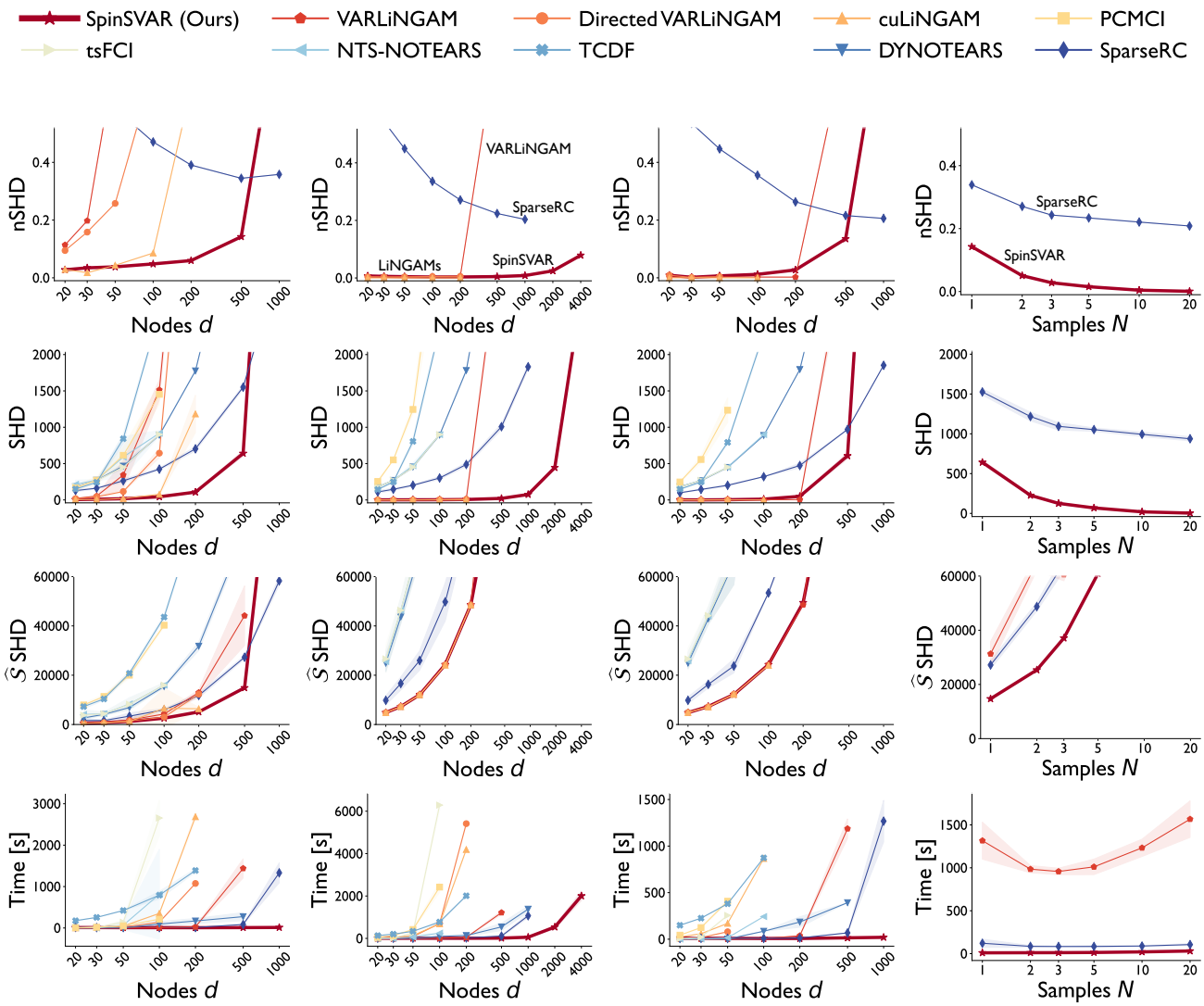
However, to allow for a greater variety of edge weights, we instead assign uniformly random weights from the range $[0.1, 0.5]$. In practice, $\boldsymbol{X}$ is typically observed to converge. If any generated dataset results in unbounded values—specifically, if the average value of $\boldsymbol{X}$ exceeds $10^6 \cdot NdT$—we discard the sample and repeat the data generation process.

E.2 ADDITIONAL SETUPS AND METRICS

In Figs. 4,5,6,7 we provide more experimental setups with the additional metrics SHD, precision (PREC), recall (REC), area under ROC curve (AUROC), F1-score, the normalized mean square error (NMSE) and the SHD and NMSE on the input $\widehat{\boldsymbol{\mathcal{S}}}$ approximation. To define formally the last two metrics, if $\widehat{\boldsymbol{W}}$ and $\widehat{\boldsymbol{\mathcal{S}}}$ are the approximations of the ground truth window graph $\boldsymbol{W}$ and structural shocks $\boldsymbol{\mathcal{S}}$ then:

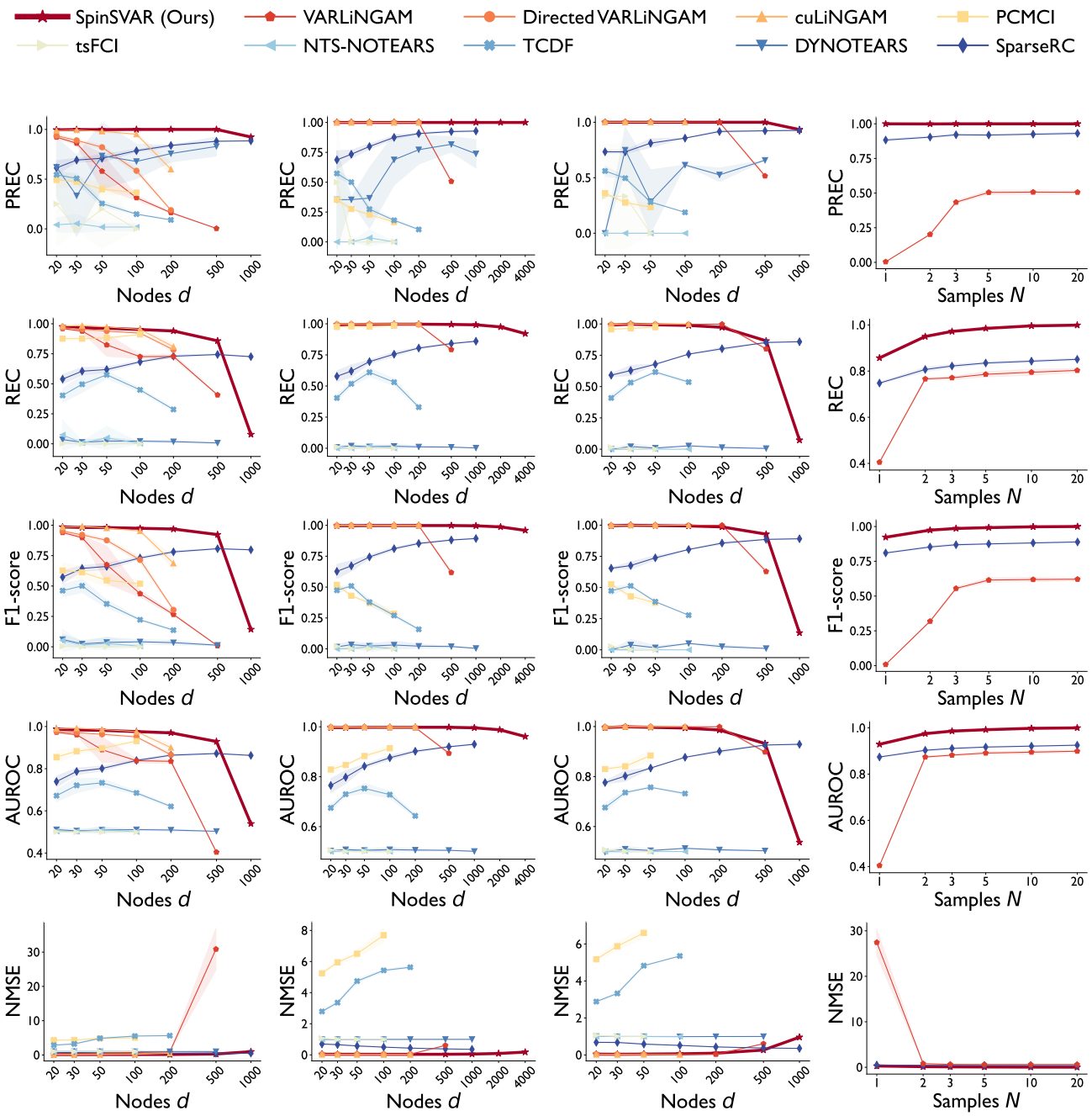
$$\text{NMSE} = \frac{\left\|\widehat{\boldsymbol{W}} - \boldsymbol{W}\right\|_2}{\|\boldsymbol{W}\|_2}, \quad \boldsymbol{\mathcal{S}}\ \text{NMSE} = \frac{\left\|\widehat{\boldsymbol{\mathcal{S}}} - \boldsymbol{\mathcal{S}}\right\|_2}{\|\boldsymbol{\mathcal{S}}\|_2}. \tag{57}$$

In Fig.4 the computation of $\widehat{\boldsymbol{\mathcal{S}}}$ NMSE is numerically unstable for all methods and is not reported.



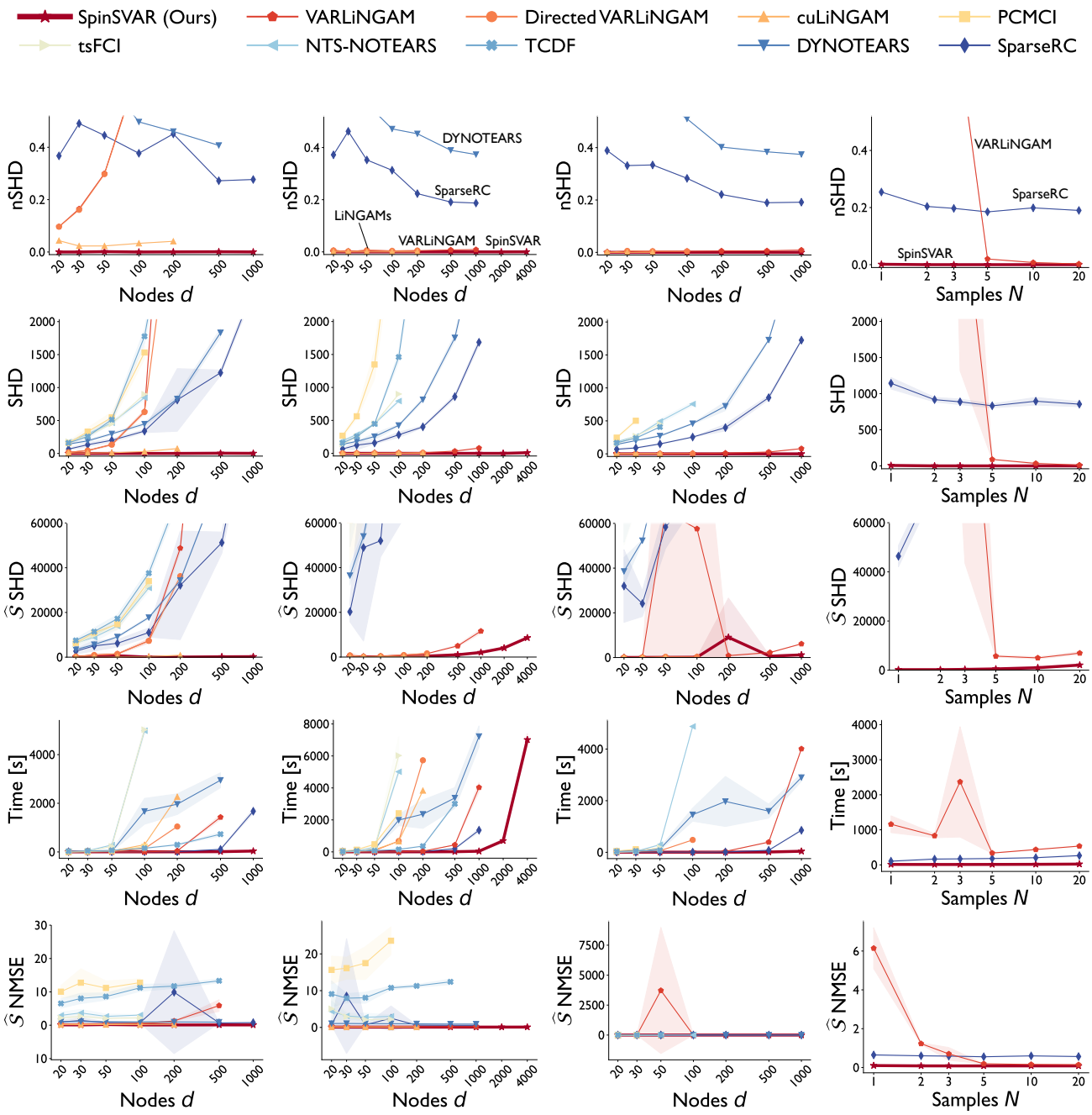
(a) $N = 1, T = 1000$ (b) $N = 10, T = 1000$ (c) $N = 1, T = 10000$ (d) $d = 500, T = 1000$

Figure 4: Performance on synthetic data (Laplacian distributed input): nSHD (↓), SHD (↓), structural shocks SHD (↓), runtime and structural shocks NMSE (↓).(a), (b) correspond to $N = 1$ and $N = 10$ samples of time-series with $T = 1000$ and varying number of nodes. (c) corresponds to $d = 500$ nodes and varying samples N of time-series of length $T = 1000$.



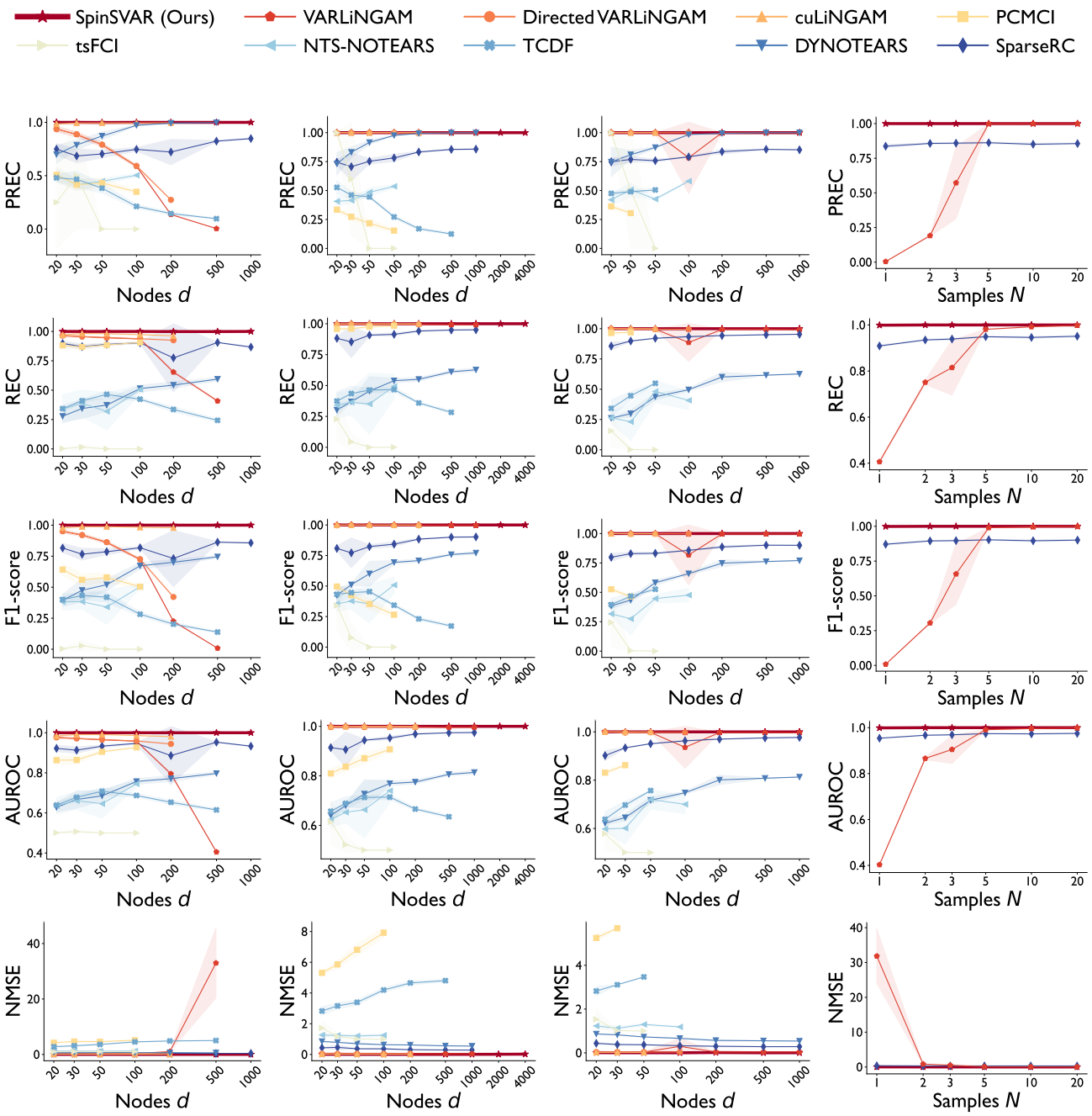
(a) $N = 1, T = 1000$ (b) $N = 10, T = 1000$ (c) $N = 1, T = 10000$ (d) $d = 500, T = 1000$

Figure 5: Performance on synthetic data (Laplacian distributed input): Precision (PREC) (↑), Recall (REC) (↑), F1-score (↑), AUROC (↑) and NMSE (↑). (a), (b) correspond to $N = 1$ and $N = 10$ samples of time-series with $T = 1000$ and varying number of nodes. (c) corresponds to $d = 500$ nodes and varying samples N of time-series of length $T = 1000$.



(a) $N = 1, T = 1000$ (b) $N = 10, T = 1000$ (c) $N = 1, T = 10000$ (d) $d = 500, T = 1000$

Figure 6: Performance on synthetic data (Bernoulli distributed input).



(a) $N = 1, T = 1000$ (b) $N = 10, T = 1000$ (c) $N = 1, T = 10000$ (d) $d = 500, T = 1000$

Figure 7: Performance on synthetic data (Bernoulli distributed input).

E.3 OTHER FORMS OF SPARSITY: GAUSSIAN SUBSAMPLING

Here we examine a third sparsity scenario, referred to as Gaussian subsampling, in which the non-zero entries of the Bernoulli variables are assigned values drawn from a normal distribution rather than uniform ones, as in the Bernoulli-uniform sparse data generation.

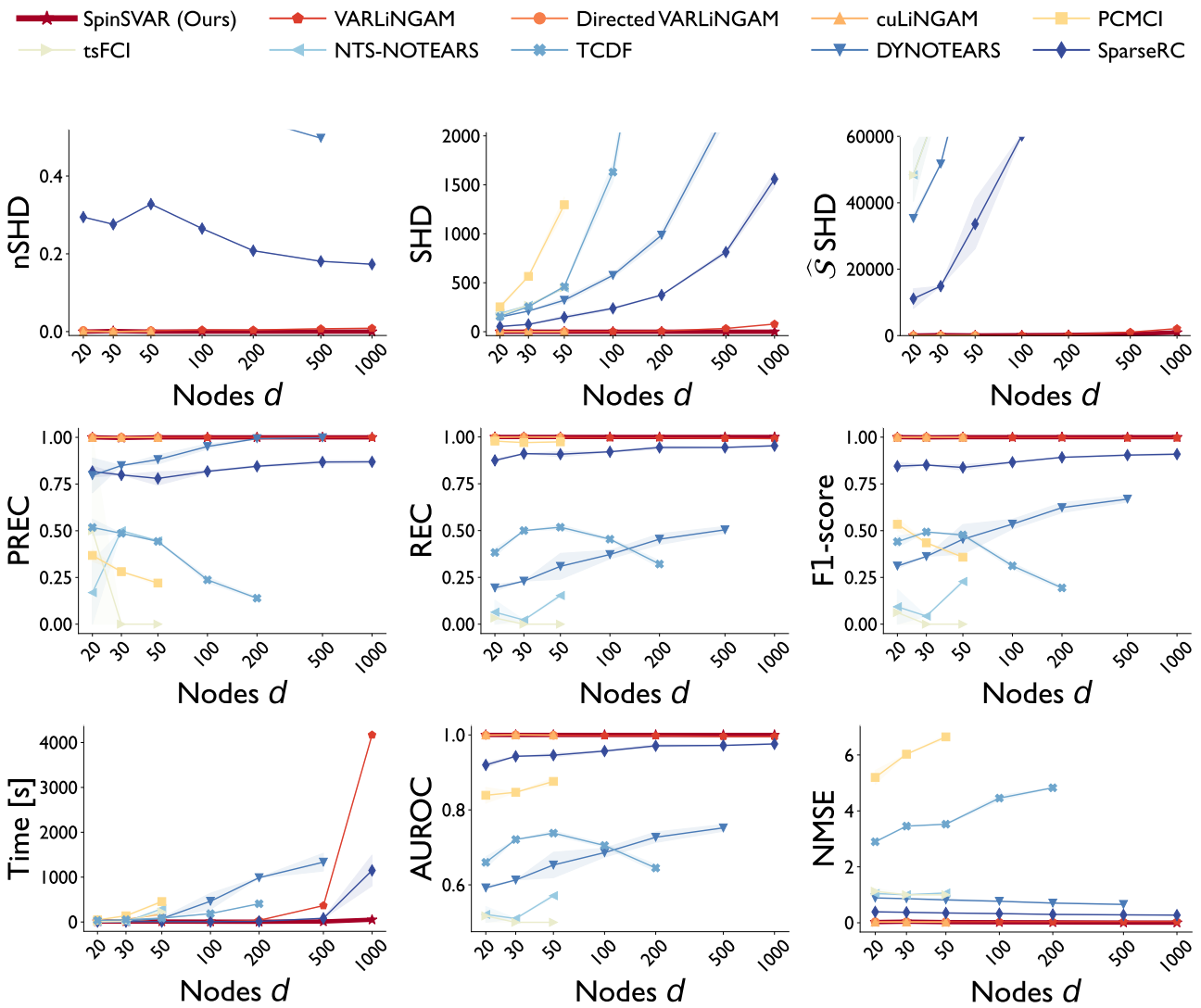


Figure 8: Synthetic data $\boldsymbol{\mathcal{X}}$ corresponding to input $\boldsymbol{\mathcal{S}}$ generated with Gaussian subsampling. $\boldsymbol{S}_{t,j} \sim \mathcal{N}(0.5, 0.1)$ with probability $p = 0.05$ and $\boldsymbol{S}_{t,j} = 0$ with probability $1 - p$. The number of samples is set to $N = 1$ and each time series sample has length $T = 1000$. The plots show performance for varying number of nodes.

E.4 LARGER TIME LAG

In Figs. 9,10, we present an experiment with a larger number of time lags, setting $k = 5$. This experiment considers $N = 10$ samples of time series, each of length $T = 1000$, while varying the number of nodes. All other experimental settings remain the same as in the main experiment, except for the weight bounds of $\boldsymbol{W}$, which are set to $[0.1, 0.2]$. This adjustment is necessary because a larger number of lags requires smaller weights to ensure bounded data, as dictated by Theorem A.1. The results are consistent with those in Fig. 2, with SpinSVAR performing better than the baselines.

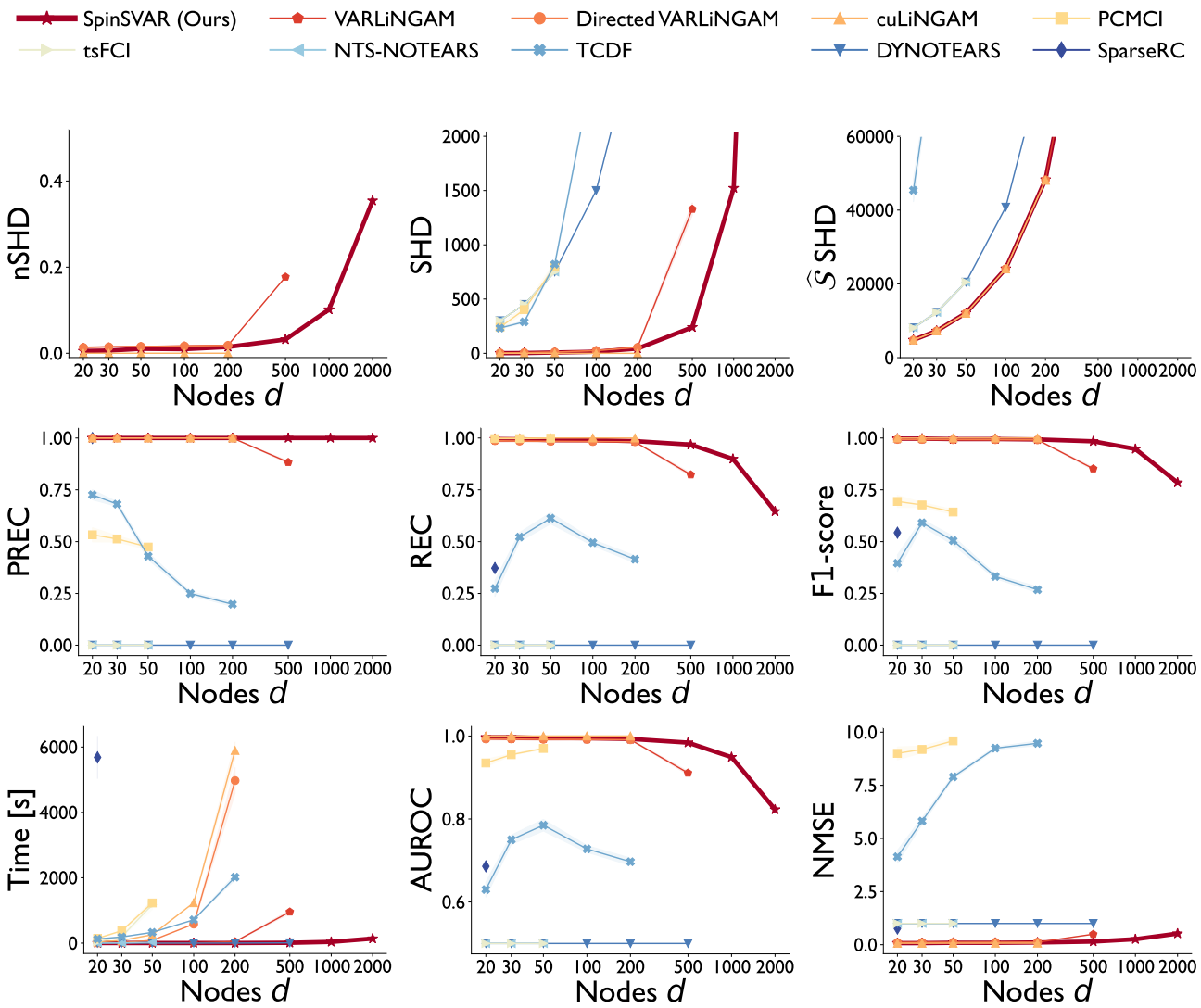


Figure 9: Synthetic experiment with with larger time lag $k = 5$, assuming input with Laplacian distribution. The number of samples is set to $N = 10$ and each time series sample has length $T = 1000$. The plots show performance for varying number of nodes.

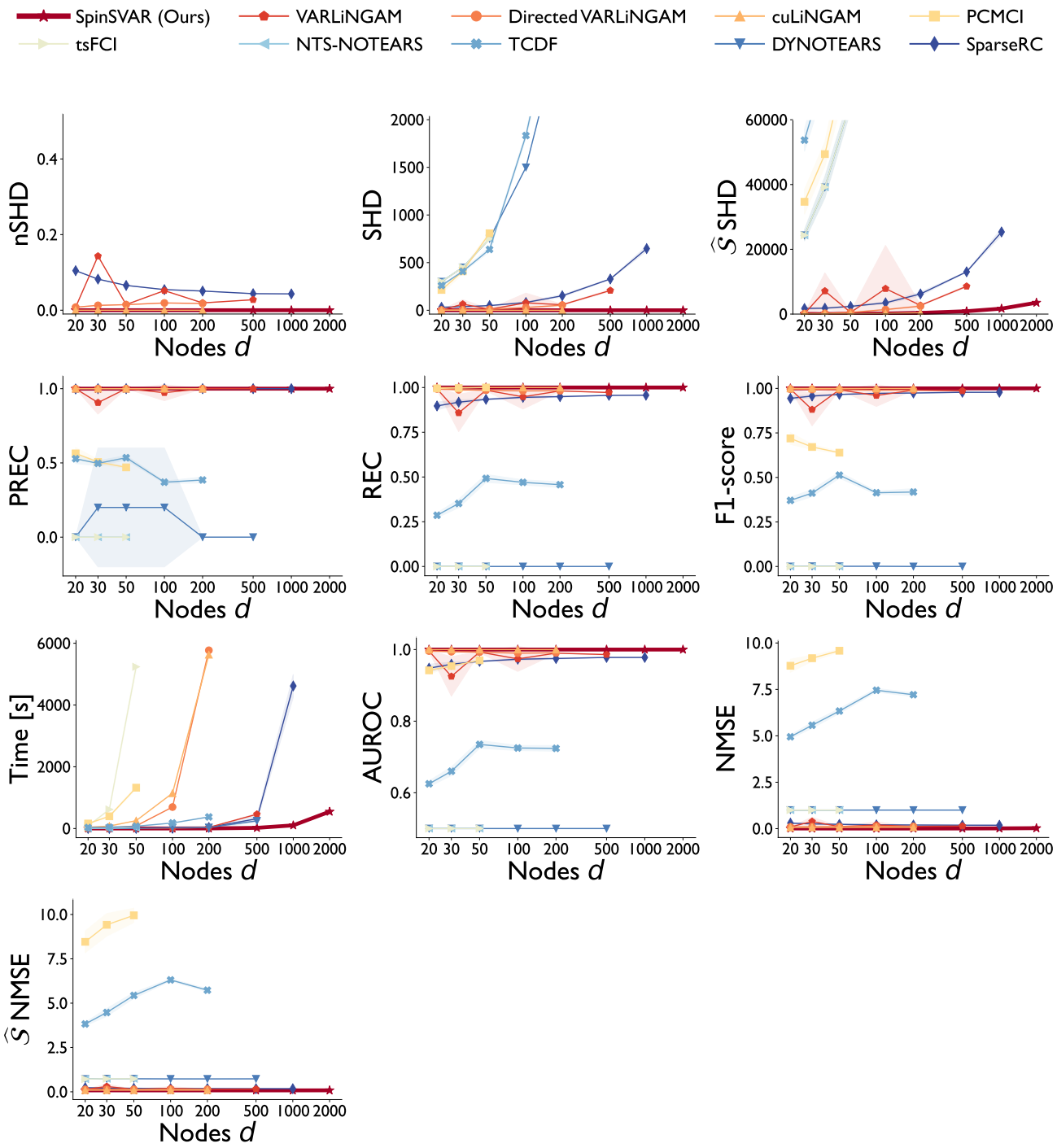


Figure 10: Synthetic experiment with larger time lag $k = 5$, assuming input with Bernoulli distribution.

E.5 SENSITIVITY OF TIME LAG

We examine the sensitivity of the time lag parameter in the algorithms using the experiment shown in Figs.11,12. This experiment follows standard synthetic settings with $d = 1000$, $T = 1000$, and a true time lag of $k = 3$.

Bernoulli-uniform input When SpinSVAR is provided with a time lag parameter $k' \geq k = 3$, its approximation remains optimal. This indicates that, as long as SpinSVAR is given a sufficiently large time lag, it can correctly identify the true maximum time lag k of the system. We observed a similar behavior in our real-world stock market experiment (Fig. 3), where SpinSVAR did not detect any time-lagged dependencies, as expected—the stock market typically reacts almost instantaneously. Conversely, if SpinSVAR is given a time lag $k' < 3$, its performance deteriorates significantly.

SparseRC performs well as long as $k' \geq k$, though its approximation remains worse than that of SpinSVAR. Additionally, SparseRC has a higher execution time and fails to complete (times out) when $k' = 6$. VARLiNGAM performs reasonably when provided with the exact time lag k, but also times out when $k' > 3$.

Other baseline methods either timed out or exhibited poor performance.

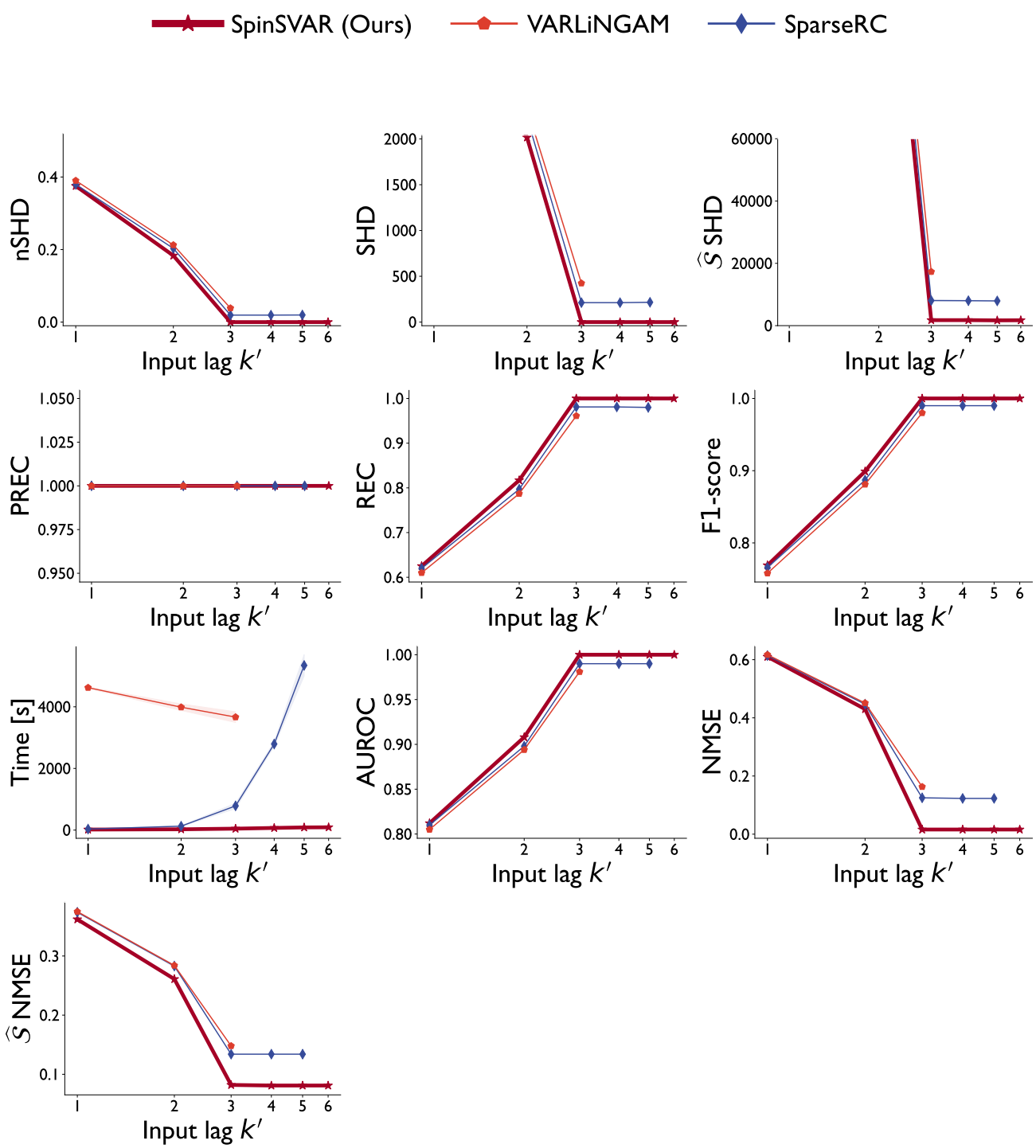


Figure 11: Evaluating the sensitivity of the time lag k in synthetic settings with original $k = 3$, $d = 1000$ nodes, $T = 1000$ and $N = 10$ samples and Bernoulli-uniform input. The algorithms have varying time lag from 1 to 6.

Laplacian input In this setting, the behavior differs slightly. When $k' \geq k$, SpinSVAR continues to perform well but is unable to recover the exact ground truth. This limitation explains the failure of the $\widehat{\mathcal{S}}$ metric. Nevertheless, SpinSVAR still outperforms the baseline methods. Notably, VARLiNGAM times out in this scenario.

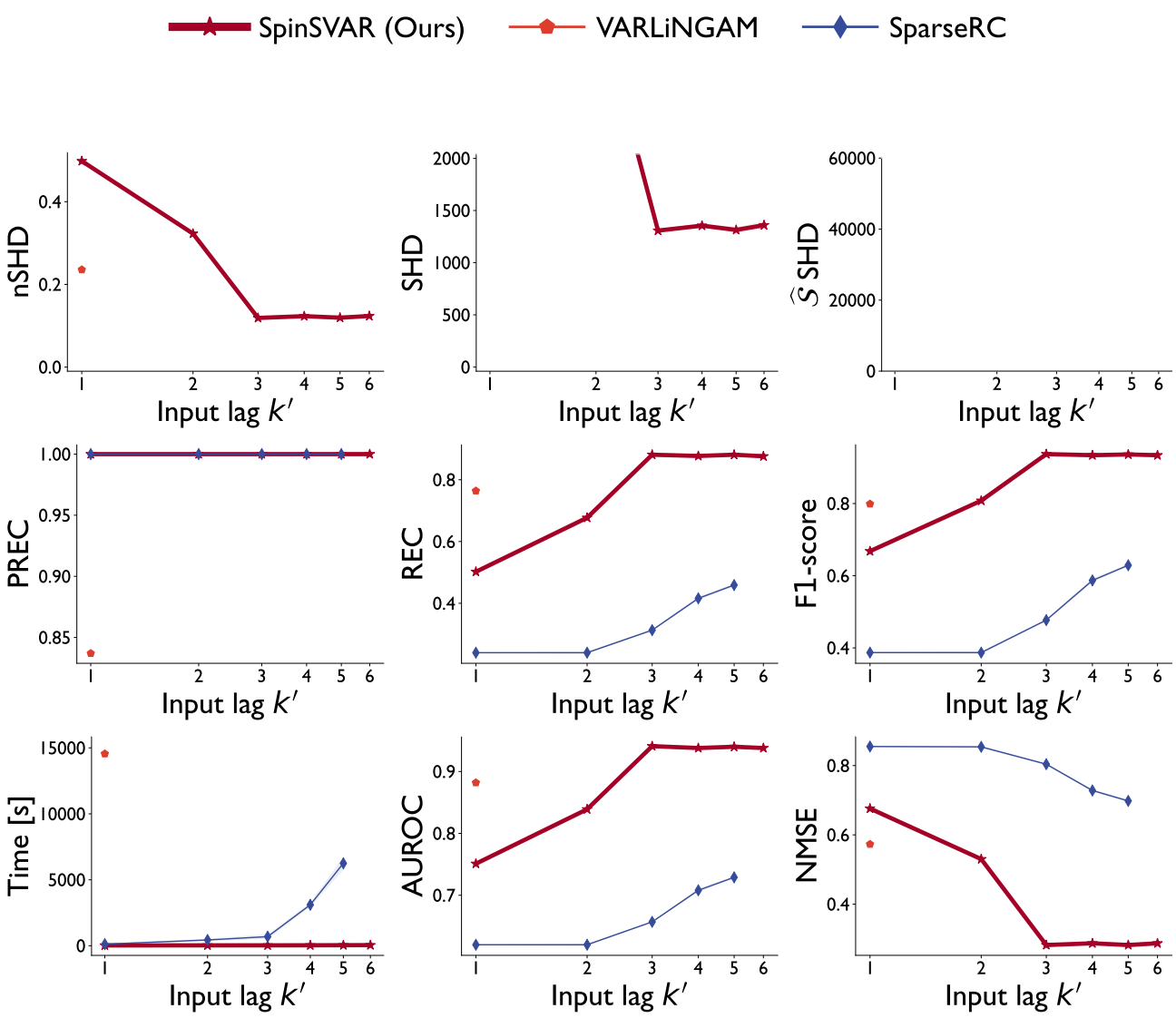


Figure 12: Evaluating the sensitivity of the time lag k in synthetic settings with original $k = 3$, $d = 1000$ nodes, $T = 1000$ and $N = 10$ samples and Laplacian input. The algorithms have varying time lag from 1 to 6.

E.6 LARGER DAGS

Here we include the time-outs of VARLiNGAM for $d = 2000$ and the performance of SparseRC which is poor compared to VARLiNGAM and SpinSVAR.

Table 3: Normalized SHD for large DAGs ($T = 1000$).

SpinSVAR $N =$	1	2	4	8	16
$d = 1000$, $\boldsymbol{\mathcal{S}} \sim$ Laplace	0.927	0.118	0.041	0.012	**0.003**
$d = 1000$, $\boldsymbol{\mathcal{S}} \sim$ Bernoulli	0.000	**0.000**	**0.000**	**0.000**	**0.000**
$d = 2000$, $\boldsymbol{\mathcal{S}} \sim$ Laplace	1.000	0.928	0.119	0.036	0.010
$d = 2000$, $\boldsymbol{\mathcal{S}} \sim$ Bernoulli	0.001	**0.000**	**0.000**	**0.000**	**0.000**
$d = 4000$, $\boldsymbol{\mathcal{S}} \sim$ Laplace	1.000	1.000	0.926	0.125	0.034
$d = 4000$, $\boldsymbol{\mathcal{S}} \sim$ Bernoulli	0.005	0.001	0.000	**0.000**	**0.000**
VARLiNGAM $N =$	1	2	4	8	16
$d = 1000$, $\boldsymbol{\mathcal{S}} \sim$ Laplace	–	–	–	–	–
$d = 1000$, $\boldsymbol{\mathcal{S}} \sim$ Bernoulli	–	–	–	0.013	0.003
$d = 2000$, $\boldsymbol{\mathcal{S}} \sim$ Laplace	–	–	–	–	–
$d = 2000$, $\boldsymbol{\mathcal{S}} \sim$ Bernoulli	–	–	–	–	–
SparseRC $N =$	1	2	4	8	16
$d = 1000$, $\boldsymbol{\mathcal{S}} \sim$ Laplace	0.365	0.247	0.219	0.203	0.202
$d = 1000$, $\boldsymbol{\mathcal{S}} \sim$ Bernoulli	0.287	0.192	0.175	0.181	0.186
$d = 2000$, $\boldsymbol{\mathcal{S}} \sim$ Laplace	–	–	–	–	–
$d = 2000$, $\boldsymbol{\mathcal{S}} \sim$ Bernoulli	–	–	–	–	–

Table 4: SHD report for large DAGs ($T = 1000$).

SpinSVAR $N =$	1	2	4	8	16
$d = 1000$, $\boldsymbol{\mathcal{S}} \sim$ Laplace	$8.3k$	$1k$	371	112	**27**
$d = 1000$, $\boldsymbol{\mathcal{S}} \sim$ Bernoulli	2	**0**	**0**	**0**	**0**
$d = 2000$, $\boldsymbol{\mathcal{S}} \sim$ Laplace	$18k$	$17k$	$2.1k$	645	183
$d = 2000$, $\boldsymbol{\mathcal{S}} \sim$ Bernoulli	12	**0**	**0**	**0**	**0**
$d = 4000$, $\boldsymbol{\mathcal{S}} \sim$ Laplace	$36k$	$36k$	$33k$	$4.5k$	$1.2k$
$d = 4000$, $\boldsymbol{\mathcal{S}} \sim$ Bernoulli	164	27	15	**7**	**9**
VARLiNGAM $N =$	1	2	4	8	16
$d = 1000$, $\boldsymbol{\mathcal{S}} \sim$ Laplace	–	–	–	–	–
$d = 1000$, $\boldsymbol{\mathcal{S}} \sim$ Bernoulli	–	–	–	115	29
$d = 2000$, $\boldsymbol{\mathcal{S}} \sim$ Laplace	–	–	–	–	–
$d = 2000$, $\boldsymbol{\mathcal{S}} \sim$ Bernoulli	–	–	–	–	–
SparseRC $N =$	1	2	4	8	16
$d = 1000$, $\boldsymbol{\mathcal{S}} \sim$ Laplace	$3.3k$	$2.2k$	$2k$	$1.8k$	$1.8k$
$d = 1000$, $\boldsymbol{\mathcal{S}} \sim$ Bernoulli	$2.6k$	$1.7k$	$1.6k$	$1.6k$	$1.7k$
$d = 2000$, $\boldsymbol{\mathcal{S}} \sim$ Laplace	–	–	–	–	–
$d = 2000$, $\boldsymbol{\mathcal{S}} \sim$ Bernoulli	–	–	–	–	–

E.7 SIMULATED FINANCIAL PORTFOLIOS

We evaluate our method on simulated financial time-series data from Kleinberg [2013], generated using the Fama-French three-factor model [Fama, 1970] (volatility, size, and value). The return $x_{i,t}$ of stock i at time t is computed as $x_{t,i} = \sum_j b_{ij} f_{t,i} + \epsilon_{t,i}$, where $f_{t,i}$ are the three factors, b_{ij} are their corresponding weights and $\epsilon_{t,i}$ are (correlated) idiosyncratic terms. We use 16 datasets from this benchmark, each incorporating time lags up to $k = 3$. The data consists of daily returns for $d = 25$ stocks, with ground truth DAGs containing an average of 22 edges. Each dataset provides a multivariate time series $\boldsymbol{X}$ with 4000 time steps, which we segment into non-overlapping windows of 50 time steps, yielding a dataset $\boldsymbol{\mathcal{X}}$ of shape $80 \times 50 \times 25$.

Table 5: Performance on the simulated financial dataset [Kleinberg, 2013].

Method	SHD (↓)	Time [s]
SpinSVAR (Ours)	12.89 ± 7.87	5.43 ± 0.65
SparseRC	$\mathbf{9.92 \pm 8.22}$	9.74 ± 1.21
VARLiNGAM	19.25 ± 10.64	$\mathbf{1.64 \pm 0.10}$
Directed VARLiNGAM	15.31 ± 9.38	4.85 ± 0.31
cuLiNGAM	15.22 ± 8.44	12.88 ± 0.42
TCDF	19.06 ± 10.18	33.56 ± 1.01
DYNOTEARS	33.92 ± 9.09	112.91 ± 29.59
NTS-NOTEARS	57.83 ± 37.22	16.40 ± 14.45
tsFCI	21.94 ± 9.52	17.50 ± 12.82
PCMCI	361.69 ± 67.80	16.23 ± 4.69

Table 5 reports the SHD and runtime for each method. Since the true structural shocks are unknown, we do not evaluate them in this setting. Hyperparameters were selected via grid search, as detailed in Appendix E.10.3. The best-performing methods are SpinSVAR and SparseRC, suggesting that assuming a sparse set of structural shocks is valid for financial data. SparseRC slightly outperforms SpinSVAR, likely due to the dataset's small scale—both in terms of time lags and number of nodes—though it remains slower. The fastest method, VARLiNGAM, exhibits weaker performance. The other baselines perform poorly in this dataset.

E.8 DREAM3 CHALLENGE DATASET

Table 6: AUROC report on the Dream3 challenge dataset [Marbach et al., 2009, Prill et al., 2010]. The methods are partitioned into non-linear and linear for a fair comparison. Best performances are marked with bold.

	Model	E.coli-1	E.coli-2	Yeast-1	Yeast-2	Yeast-3
Non-linear	MLP	0.644	0.568	0.585	0.506	0.528
	LSTM	0.629	0.609	0.579	0.519	0.555
	TCDF	0.614	0.647	0.581	0.556	**0.557**
	SRU	0.657	**0.666**	0.617	**0.575**	0.55
	eSRU	**0.66**	0.629	**0.627**	0.557	0.55
	PCMCI	0.594	0.545	0.498	0.491	0.508
	NTS-NOTEARS	0.592	0.471	0.551	0.551	0.507
	tsFCI	0.5	0.5	0.5	0.5	0.5
Linear	**SpinSVAR (Ours)**	0.547	0.525	0.551	0.508	**0.513**
	SparseRC	0.543	0.516	**0.554**	0.507	0.512
	VARLiNGAM	0.545	0.519	0.516	0.509	0.502
	Directed VARLiNGAM	0.504	0.501	0.514	0.501	0.510
	DYNOTEARS	**0.590**	**0.547**	0.527	**0.526**	0.510

In Table 6 we report the AUROC performance of our method compared to baselines. There, Component-wise MLP and LSTM are from [Tank et al., 2021] and SRU and eSRU from [Khanna and Tan, 2019]. while the rest of the methods are present in the main paper. The results of the first 5 rows are taken from [Khanna and Tan, 2019] and DYNOTEARS from [Gong et al., 2022]. The methods are partitioned into non-linear and linear for a fair comparison.

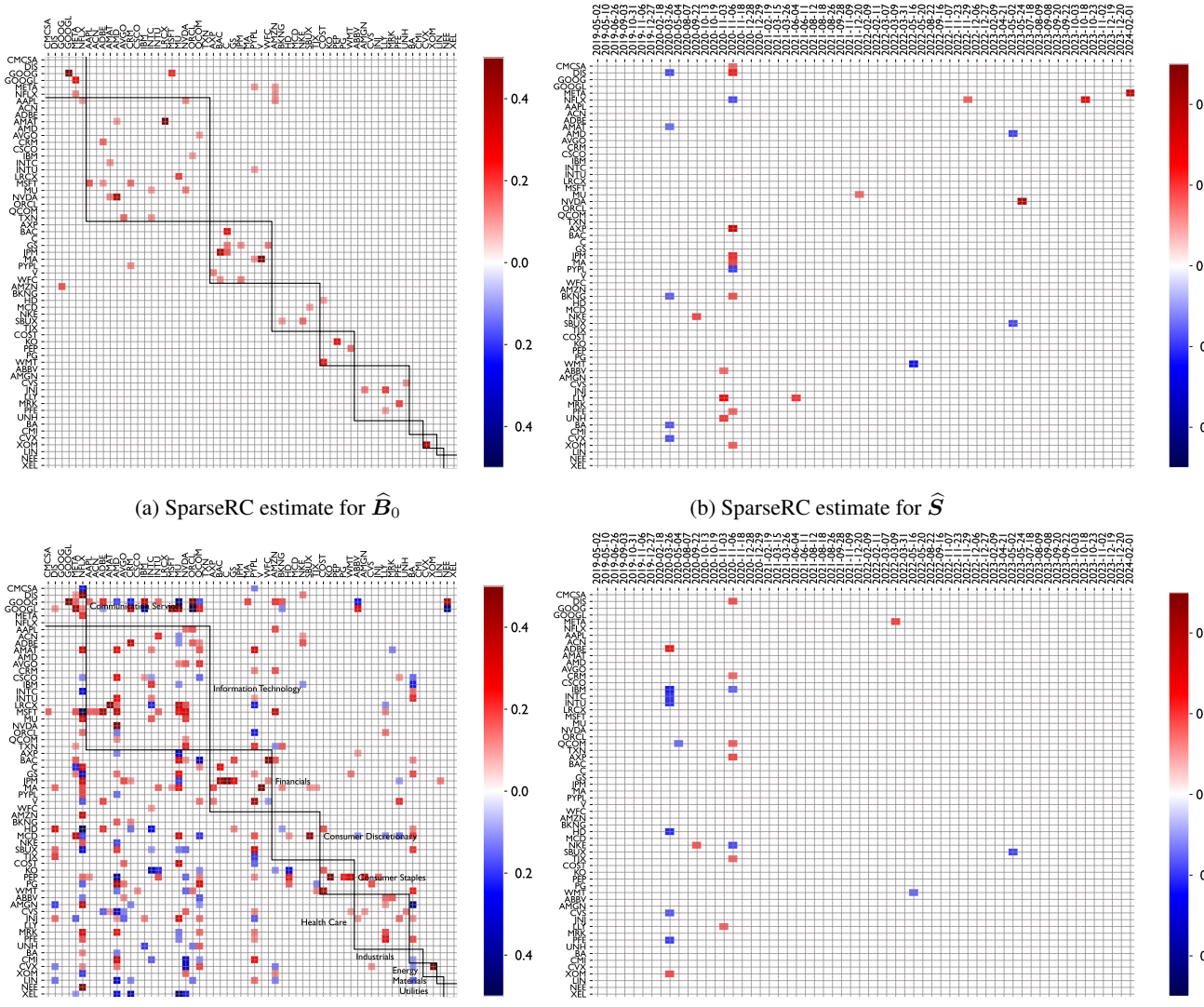
Our method is competitive to other linear-model baselines but worse than those assuming a nonlinear model. Apparently, one of the two assumptions, either the sparse SVAR input assumption or linearity of the data generation does not hold in this dataset and our method might not be the most appropriate.

E.9 S&P 500 REAL EXPERIMENT

In Figs. 13 and 14 we show the performance of SparseRC, VARLiNGAM, TCDF and PCMCI on the S&P 500 stock market index. As also mentioned in the main text, SparseRC approximates a DAG similar to SpinSVAR. This is due to the few

structural shock assumption that both methods use.

- **VARLiNGAM** seems to identify significant edges for any random stock combination, thus producing a poor result. Also, the approximated structural shocks $\widehat{\boldsymbol{S}}$ are less expressive than ours in the sense that out of the 4507 discovered structural shocks only 33.7% of them align with the data changes.
- **TCDF** produces a very sparse DAG with not enough information.
- **PCMCI** outputs a zero graph for time lag 0 and a not well-structured graph for time lag 1. As a consequence, we don't see a meaningful pattern in the structural shocks.
- **DYNOTEARS** had as output an empty graph and thus its performance is not reported. Regarding its hyperparameters, we minimized the weight threshold up to 0 (all weights included as edges) and we tried both $\lambda_w = \lambda_a = 0.01$ and $k = 2$, which were the optimal from our synthetic experiments and $\lambda_w = \lambda_a = 0.1$ which is the reported best in the S&P 100 experiment in [Pamfil et al., 2020].
- Directed VARLiNGAM, cuLiNGAM, tsFCI and NTS-NOTEARS had time-out in this experiment.



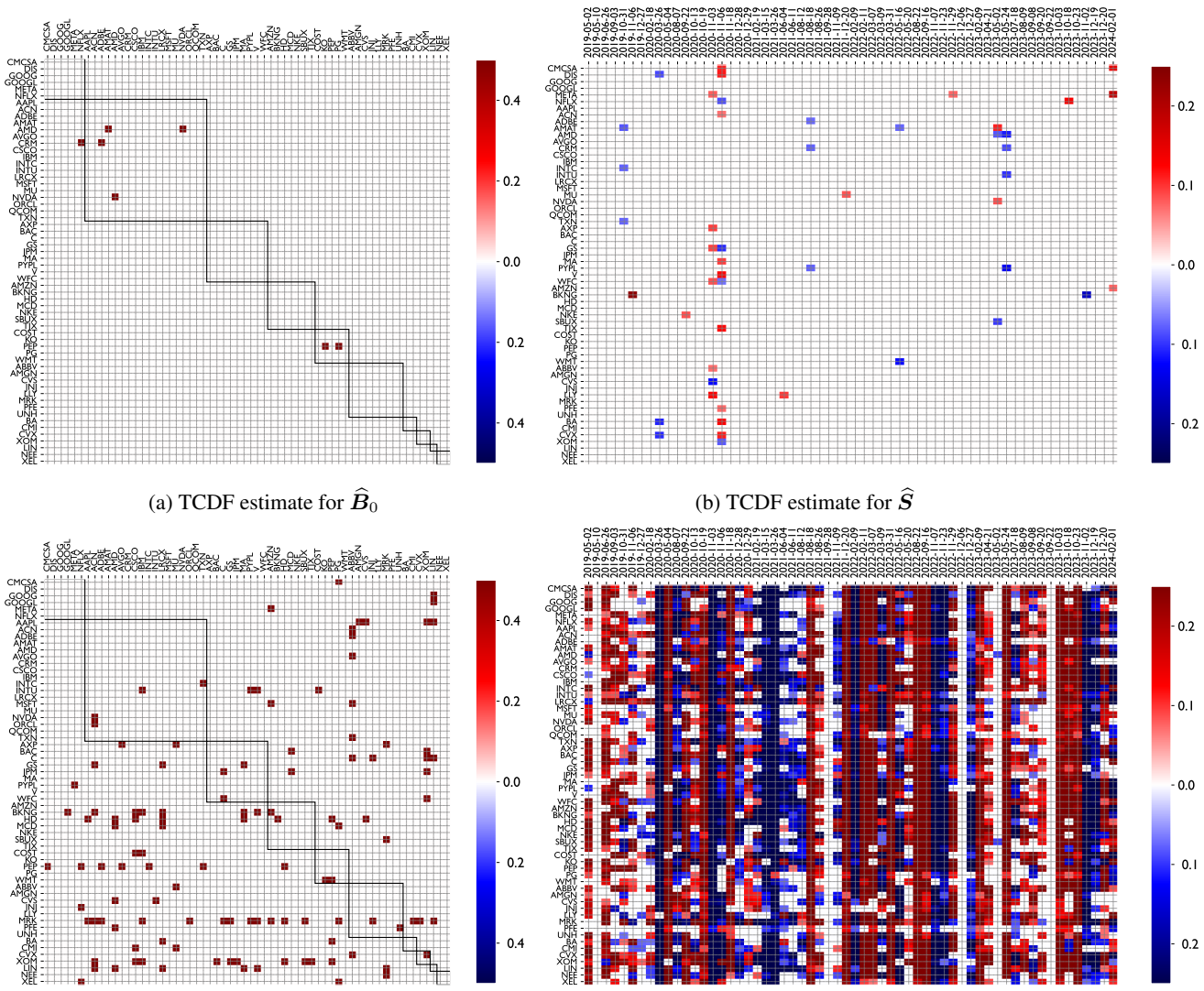
(a) SparseRC estimate for $\widehat{\boldsymbol{B}}_0$

(b) SparseRC estimate for $\widehat{\boldsymbol{S}}$

(c) VARLiNGAM estimate for $\widehat{\boldsymbol{B}}_0$

(d) VARLiNGAM estimate for $\widehat{\boldsymbol{S}}$

Figure 13: Evaluating baselines on the real experiment with S&P 500 stock market index. (a) Instantaneous relations between the 45 highest weighted stocks within S&P 500 and (b) the discovered structural shocks for 60 dates.



(a) TCDF estimate for $\widehat{\boldsymbol{B}}_0$

(b) TCDF estimate for $\widehat{\boldsymbol{S}}$

(c) PCMCI estimate for $\widehat{\boldsymbol{B}}_1$

(d) PCMCI estimate for $\widehat{\boldsymbol{S}}$

Figure 14: Evaluating PCMCI on the real experiment with S&P 500 stock market index. (a) Relations between the 45 highest weighted stocks within S&P 500 with time lag 1 and (b) the discovered structural shocks for 60 dates.

E.10 HYPERPARAMETER SEARCH

To find the most suitable hyperparameter selection for each method in our synthetic and simulated experiments we perform a grid search and choose the parameter combination that achieves the best nSHD performance.

E.10.1 Synthetic experiments with Laplacian input

For convenience we perform the grid search on small synthetic experimental settings ($N = 1$ sample, $T = 1000$ time steps, $d = 20$ nodes) where all methods have reasonable execution time. Note that for all methods we set their parameters regarding the number of lags correctly, to equal the ground truth lag (default $k = 2$). Any non-relevant hyperparameter that is not mentioned is set to its default value. The hyperparameter search gave the following optimal hyperparameters for each method:

SpinSVAR We set $\lambda_1 = 0.0005$, $\lambda_2 = 0.5$ the coefficients for the L^1 and acyclicity regularizer, respectively and $\omega = 0.09$. We let SpinSVAR run for 10000 epochs, although usually it terminates earlier as we have an early stopping activated when

for 40 consecutive epochs the loss didn't decrease.

SparseRC We set $\lambda_1 = 0.001$, $\lambda_2 = 1$, $\lambda_3 = 0.001$ the coefficients for the L^1, acyclicity and block-Toeplitz regularizers, respectively and $\omega = 0.09$. We similarly let SparseRC run for 10000 epochs, although usually it terminates earlier using early stopping as with SpinSVAR.

VARLiNGAM We may choose between ICA or Direct LiNGAM. In our experiments, we consider both cases (VAR-LiNGAM and Directed VARLiNGAM). The weight threshold is set to 0.09 both for VARLiNGAM and Directed VAR-LiNGAM but for cuLiNGAM is set to 0.05.

DYNOTEARS The resulting values are $\lambda_w = \lambda_a = 0.01$ and $\omega = 0.01$

NTS-NOTEARS The resulting values are $\lambda_1 = 0.002$, $\lambda_2 = 0.01$ and $\omega = 0.01$ The h_{tol} and the dimensions of the neural network were left to default.

tsFCI Significance level is set to 0.1 and $\omega = 0.01$. Note that the output of tsFCI is a partial ancestral graph (PAG), which we therefore need to interpret as a DAG. For this scope we follow the rules of DYNOTEARS [Pamfil et al., 2020], meaning that whenever there is ambiguity in the directionality of the discovered edge we assume that tsFCI made the correct choice (this favors and over-states the performance of tsFCI). In particular, we translate the edge between nodes i and j in the following ways (i) if $i \to$ we keep it, (ii) if $i \leftrightarrow j$ in the PAG we discard it, (iii) either $i \circ\!\!\to j$ or $i \circ\!\!-\!\!\circ j$ we assume tsFCI made the correct choice, by looking at the ground truth graph.

PCMCI The ParCorr conditional independence test was chosen. We do so because this test is suitable for linear additive noise models. Parameters are set as $pc_a = 0.1$, $a_{level} = 0.01$ and $\omega = 0.01$. The output can sometimes be ambiguous ($\circ - \circ$) because the algorithm can only find the graph up to the Markov equivalence class, or there be conflicts ($x - x$) in the conditional independence tests. In the former case, we assume that PCMCI made the correct choice and in the latter we disregard the edge.

TCDF Here the kernel size and the dilation coefficient are set as the number of lags $+1$ ($k + 1 = 3$). The other parameters are $significance = 1$ and $epochs = 1000$ and $\omega = 0.01$.

E.10.2 Synthetic experiments with Bernoulli-uniform input

Similarly for the Laplacian input, we perform hyperparameter search for $N = 1$ sample, $T = 1000$ time steps and $d = 20$ nodes. The hyperparameter search gave the following optimal hyperparameters for each method:

SpinSVAR We set $\lambda_1 = 0.0001$, $\lambda_2 = 0.1$ and $\omega = 0.09$. We let SpinSVAR run for 10000 epochs.

SparseRC We set $\lambda_1 = 0.001$, $\lambda_2 = 1$, $\lambda_3 = 0.001$ and $\omega = 0.09$. We similarly let SparseRC run for 10000.

VARLiNGAM In our experiments, we consider both cases (VARLiNGAM and Directed VARLiNGAM). The weight threshold is set to 0.09 both for VARLiNGAM and Directed VARLiNGAM but for cuLiNGAM is set to 0.05.

DYNOTEARS The resulting values are $\lambda_w = \lambda_a = 0.01$ and $\omega = 0.09$.

NTS-NOTEARS The resulting values are $\lambda_1 = 0.002$, $\lambda_2 = 0.01$ and $\omega = 0.09$. The h_{tol} and the dimensions of the neural network were left to default.

tsFCI Significance level is set to 0.1 and $\omega = 0.09$.

PCMCI Parameters are set as $pc_a = 0.1$, $a_{level} = 0.01$ and $\omega = 0.09$.

TCDF Here the kernel size and the dilation coefficient are set as the number of lags $+1$ ($k + 1 = 3$). The other parameters are $significance = 1$ and $epochs = 1000$ and $\omega = 0.09$.

E.10.3 Simulated financial data

Here we perform the grid search on the first available dataset of the simulated data (out of the 16 available) and choose the hyperparameters offering the best SHD performance. Here, we search for the most compatible weight threshold ω as the distribution of the ground truth weights is not known from the data generation. For all methods we set the number of maximum time lags at 3, which is the maximal ground truth lag. Any non-relevant hyperparameter that is not mentioned is set to its default value. The hyperparameter search gave the following optimal hyperparameters for each method:

SpinSVAR We set $\lambda_1 = 0.01$, $\lambda_2 = 1$, $\omega = 0.5$. We let SpinSVAR run for 10000 epochs at maximum.

SparseRC We set $\lambda_1 = 0.001$, $\lambda_2 = 1$, $\lambda_3 = 0.1$, $\omega = 0.3$. We similarly let SparseRC run for 10000 epochs at maximum.

VARLiNGAM The weight threshold is set to $\omega = 0.5$ for VARLiNGAM and $\omega = 0.6$ for Directed VARLiNGAM and cuLiNGAM.

DYNOTEARS The resulting values are $\lambda_w = 0.05$, $\lambda_a = 0.01$, $\omega = 0.3$.

NTS-NOTEARS The resulting values are $\lambda_1 = 0.001$, $\lambda_2 = 1$, $\omega = 0.1$. The h_{tol} and the dimensions of the neural network were left to default.

tsFCI Significance level is set to 0.001 and $\omega = 0.1$ As previously we favor tsFCI in case of ambiguity, using the ground truth.

PCMCI The ParCorr conditional independence test was chosen and parameters are set as $pc_a = 0.1$, $a_{level} = 0.01$, $\omega = 0.1$. In case of ambiguity, we assume PCMCI made the correct choice.

TCDF The kernel size and the dilation coefficient are set as number of lags $+1$ ($k + 1 = 4$). The other parameters are $significance = 0.8$, $epochs = 1000$, $\omega = 0.2$.

E.10.4 DREAM3 dataset

Here we perform the grid search on the first available dataset of the data (out of the 5 available) and choose the hyperparameters offering the best AUROC performance. We get the following results.

SpinSVAR $\lambda_1 = 0.001$, $\lambda_2 = 10$, $\omega = 0.2$. We let SpinSVAR run for 10000 epochs at maximum.

SparseRC $\lambda_1 = 0.01$, $\lambda_2 = 0.1$, $\lambda_3 = 0.1$, $\omega = 0.2$. We similarly let SparseRC run for 10000 epochs at maximum.

VARLiNGAM The weight threshold is set to $\omega = 0.2$ for VARLiNGAM, Directed VARLiNGAM and cuLiNGAM

DYNOTEARS The resulting values are $\lambda_w = 0.05$, $\lambda_a = 0.01$, $\omega = 0.3$.

NTS-NOTEARS The resulting values are $\lambda_1 = 0.001$, $\lambda_2 = 0.01$, $\omega = 0.2$. The h_{tol} and the dimensions of the neural network were left to default.

tsFCI Significance level is set to 0.001 and $\omega = 0.1$

PCMCI $pc_a = 0.1$, $a_{level} = 0.01$, $\omega = 0.1$.

TCDF The kernel size and the dilation coefficient are set as number of lags $+1$ ($k + 1 = 4$). The other parameters are $significance = 0.8$, $epochs = 1000$, $\omega = 0.2$.

E.11 COMPUTE RESOURCES

Our experiments were run on a single laptop machine (Dell Alienware x17 R2) with 8 core CPU with 32GB RAM and an NVIDIA GeForce RTX 3080 GPU. The execution of the synthetic experiments for the 5 repetitions amounts to approximately 1 week of full run. Of course, initially there were some failed experiments, and after debugging the experiments were executed for only 1 repetition to determine where each method has a time-out. We thus chose the time-out to 10000 to try to make our experiments with as little cost as possible.

E.12 CODE RESOURCES

For the implementation of the methods in our experiments we use the following publicly available repositories or websites. All github repositories are licensed under the Apache 2.0 or MIT license, except tigramite and TCDF which are under the GPL-3.0 license.

SparseRC SparseRC code https://github.com/pmisiakos/SparseRC/. (MIT license)

VARLiNGAM We use the official LiNGAM repo which we clone from github: https://github.com/cdt15/lingam. (MIT license)

cuLiNGAM Akinwande and Kolter [2024] provide the following github repo: https://github.com/aknvictor/culingam. (MIT license)

DYNOTEARS Code is available from the CausalNex library of QuantumBlack. The code is at https://github.com/mckinsey/causalnex/blob/develop/causalnex/structure/dynotears.py (Apache 2.0 license)

NTS-NOTEARS We use the github code https://github.com/xiangyu-sun-789/NTS-NOTEARS provided by Sun et al. [2023]. (Apache 2.0 license)

tsFCI We use the R implementation from Doris Entner website which in turn utilizes the https://www.cmu.edu/dietrich/philosophy/tetrad/. Tetrad is licensed under the GNU General Public License v2.0. We also used the repository https://github.com/ckassaad/causal_discovery_for_time_series corresponding to the causal time series survey [Assaad et al., 2022b] (no license available).

PCMCI We use the PCMCI implementation from [Runge et al., 2019] within the tigramite package. (GNU General Public License v3.0)

TCDF We use the repository https://github.com/M-Nauta/TCDF from Nauta et al. [2019]. (GNU General Public License v3.0)

eSRU We use the repository https://github.com/iancovert/Neural-GC from Khanna and Tan [2019]. (MIT License)

E.13 DATA RESOURCES

Simulated financial time series We take the data from http://www.skleinberg.org/data.html licensed under CC BY-NC 3.0

S&P 500 stock returns The data are downloaded using *yahoofinancials* python library.