Circuits and Systems
Mekelweg 4,
2628 CD Delft
The Netherlands
https://sps.ewi.tudelft.nl/

SPS-2025-00

M.Sc. Thesis

Dynamic Graph Topology Identification: A Kalman Filtering Approach

Anja Kroon B.Eng.

Abstract

Complex systems and networks, including financial, brain, transport, and social networks, can be modeled as graphs. Learning their connectivity is valuable because structure drives dynamics, enabling prediction, monitoring, and control. Applications include understanding diseases such as Alzheimer's, epilepsy, and other neurological disorders. Past work has emphasized static graphs with fixed connectivity, while more recent efforts address dynamic graphs that better reflect real-world networks. To accommodate time variation, online and recursive methods update connectivity estimates as new node signals arrive.

An online algorithm for graph topology identification, *Kalman Dynamic Online GTI* (KalDO GTI), has been developed where the adjacency is the hidden state of a state–space model and is updated via a recursive Kalman filter. Beyond state tracking, the state transition matrix is also estimated online through an optimization loop and is refined with Adam-based updates under sparsity, masking, and structural priors. This two-tier structure, sequential Kalman prediction–correction for adjacency tracking and batch optimization for state dynamics learning, yields a recursive yet adaptive framework suited for online, streaming data. In direct comparison with streaming LASSO and prediction–correction methods, the algorithm is competitive across regimes and particularly robust under high measurement noise. Experiments on synthetic networks with aggressive dynamic regimes validate the approach and show similar or better performance compared to existing methods.

Dynamic Graph Topology Identification: A Kalman Filtering Approach

Thesis

submitted in partial fulfillment of the
requirements for the degree of

Master of Science

in

Electrical Engineering

by

Anja Kroon B.Eng.
born in New Jersey, United States of America

This work was performed in:

Circuits and Systems Group
Department of Microelectronics
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Delft University of Technology

Delft University of Technology
Department of
Microelectronics

The undersigned hereby certify that they have read and recommend to the Faculty of Electrical Engineering, Mathematics and Computer Science for acceptance a thesis entitled **"Dynamic Graph Topology Identification: A Kalman Filtering Approach"** by **Anja Kroon B.Eng.** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: 29 September 2025

Chairman: _____
prof.dr.ir. Geert Leus

Committee Members: _____
prof.dr.ir. Huijuan Wang

_____
ir. Yongsheng Han

Abstract

Complex systems and networks, including financial, brain, transport, and social networks, can be modeled as graphs. Learning their connectivity is valuable because structure drives dynamics, enabling prediction, monitoring, and control. Applications include understanding diseases such as Alzheimer's, epilepsy, and other neurological disorders. Past work has emphasized static graphs with fixed connectivity, while more recent efforts address dynamic graphs that better reflect real-world networks. To accommodate time variation, online and recursive methods update connectivity estimates as new node signals arrive.

An online algorithm for graph topology identification, *Kalman Dynamic Online GTI* (KalDO GTI), has been developed where the adjacency is the hidden state of a state–space model and is updated via a recursive Kalman filter. Beyond state tracking, the state transition matrix is also estimated online through an optimization loop and is refined with Adam-based updates under sparsity, masking, and structural priors. This two-tier structure, sequential Kalman prediction–correction for adjacency tracking and batch optimization for state dynamics learning, yields a recursive yet adaptive framework suited for online, streaming data. In direct comparison with streaming LASSO and prediction–correction methods, the algorithm is competitive across regimes and particularly robust under high measurement noise. Experiments on synthetic networks with aggressive dynamic regimes validate the approach and show similar or better performance compared to existing methods.

Acknowledgments

This past year has been challenging and, in equal measure, rewarding. Alongside my first solo research endeavor came a difficult health confrontation. The MSc thesis is already a time of wandering and trying on research styles; becoming symptomatic, battling my own body, and cycling through treatments before finally finding one that worked made the year extra "character-building" as one might say. I can therefore confidently say that I would not have finished this thesis, or managed life around it, without a whole village. To my supervisor, colleagues, friends in EWI, friends and family in the US, the Netherlands, and Canada — thank you. Your patience with me this year is something I will always cherish.

To my supervisor, Geert, thank you chiefly for your patience. From working together, I have strengthened my mathematical analysis skills and significantly improved my research process. Week after week, you were available to meet with me taking the time to explain concepts and help me with my thesis. I am also very grateful for your understanding especially in the late fall and winter months when I was suffering with health issues and had not yet gleamed an idea of what may be going on.

To Yongsheng, I would like to thank you for your support. I always enjoyed our discussions and you provided me always with great advice. I am excited to see what direction you take your PhD research in and am already looking forward to reading your next work.

To my fellow peers in the SPS program, we did it! Looking back, it is easy to appreciate how far we have come but I think only we know the difficulties that growth truly entailed. I am honored to have been your classmate and even more so to be your friends. I will hold the memories dear of both the highs of our successes and the lows of the trenches especially during our coursework. It has already been so exciting to see everyone embark on future endeavors using the knowledge obtained to make an impact on the world.

Throughout my master's degree, I am also proud to have been involved with D.S.R.V. Laga, a student rowing association in Delft. I would like to particularly thank my first year teammates and friends Isa, Tessa, Saskia and Marta for their guidance as I navigated the very awkward process of learning to integrate into the Dutch culture. Your stories, laughter, and support is so very much appreciated and definitely shaped my master's experience. Thank you for providing me the privilege of community and a home away from home. I would also like to extend the same to my second year teammates ORC committee, and friends from Laga. For checking in on me, and making me smile after spending a long day, brow furrowed and hunched over a textbook or equations, in the EWI.

I have many reasons to be thankful to Laga but particularly for introducing me to my now boyfriend, Thom. As I navigated this difficult time both in research and with my health, you were there to support me and help me manage it all. Your kindness and consideration are inspirational and the the laughter and light you brought in to

soften my most stressful days will never be forgotten.

Thank you also to the Weitering Family - Lenie, Geert, Kirsten, Noud, Sacha, and Jeroen - for opening your home to me and making feel so very welcome here in the Netherlands. I appreciate earnestly your kindness and consideration and for giving me many fun memories and a 'thuis thuis' in the Netherlands.

Lastly, I would like to thank my parents. Regardless of where you were you always made time for me. To my dad, thank you for lending a safe listening ear to me on a daily basis as I shared life's mundane updates, had mental breakdowns, cried, and ranted. Even from another continent, you provided me unwavering support and I will forever be eternally grateful. To my mom, thank you for providing inspiration and helping me tackling life's challenges. Thank you for always being there regardless of the hour of the day and for lending a listening ear when I needed to talk. Your stories always made me laugh and talking to you often helped me place things in perspective.

This past year has taught me a lot about appreciating the often overlooked things in life – the small, seemingly insignificant wins and physical health. To my community, thank you, I am so very grateful. This win doesn't feel like mine alone – it feels like ours.

Anja Kroon B.Eng.
Delft, The Netherlands
29 September 2025

Contents

List of Figures

List of Tables

1 Introduction

Graphs are a powerful mathematical tool to represent complex systems and networks. Nodes in these graphs correspond to entities and edges to the relations among them. Examples of real world systems that graphs can represent include brain networks, financial markets, social connections, and transport systems to name a few [1, 2, 3]. As we move towards an increasingly online society with rich technological advancements, the data needed to create and analyze graphs is becoming increasingly abundant. In parallel, the potential applications of graph analysis methods are expanding in healthcare, politics, and industry. Therefore, graph learning and graph signal analysis has grown increasingly important in the last decade prompting the creation of the field, Graph Signal Processing (GSP). There, well-tested mathematical theory in filtering, estimation, and signal processing is applied and further developed to analyze increasingly complex and high-dimensional networks for retrospective or real-time analysis.

The focus of this work is placed upon the issue of *Graph Topology Inference* (GTI). GTI is formally defined as recovering, learning, the underlying graph from observed nodal data, called *graph signals*. The data over nodes forms a snapshot of the states of the nodes and, typically, also the system [4].

Traditionally, GTI has been studied in a *static* setting, where the *topology*, underlying graph connectivity, is assumed constant, fixed over time. Static GTI methods typically follow and record blocks of the graph signal over time and process the data to produce a single, fixed, time-invariant structure per block [5, 6, 7, 8]. By aggregating many signal observations into one block, the system of equations relating the unknown graph to the data becomes overdetermined, which makes precise estimation of the graph possible. While effective for its intended purpose, assuming a static network can fail in practice particularly for real-world systems which evolve over time. This motivates the field of *dynamic* GTI, where both graph signals and graph topology vary over time [4]. Graph signals can be single- or multi-dimensional, but are always numeric representations of node states. A financial network, for example, may have a two-dimensional graph signal with one variable reporting the stock market price and the other the total revenue of the company in the last quarter. This work shall focus on graph signals with one input per node.

Performing dynamic GTI is significantly more challenging than static GTI. Slow-changing graphs can still utilize collecting a block of data to perform dynamic GTI but fast-changing graphs are not suitable for this approach [9, 5].

To address this challenge, approaches consider leveraging prior system knowledge and compressed representations of information such that a smaller block of data is needed to produce an estimate of graph connectivity. These may include smoothness priors that constrain signals to vary gradually over the graph, sparsity penalties that restrict the number of active edges, or low-rank assumptions on latent factors driving the observed signals [5, 10, 11]. The main criticism, pointed out also by the authors of

these works, is that the dynamics must be slow-time varying. They remain fundamentally limited by the windowing approach. That is chiefly because these approaches are based on a block of information rather than a recursive implementation.

Alternative approaches are starting to be developed which process incoming graph signals as they arrive and keep a running notion of graph connectivity which is then updated to accommodate varying graph connectivity dynamics. Algorithms which operate using an incoming stream of data are called *online*. Online algorithms can also be *recursive*. These algorithms only use the previous time-step computations to produce the next, and typically reduce complexity by avoiding recomputing over past data.

Recent work by Natali et al. proposed a prediction–correction framework for online GTI based on time-varying convex optimization [12]. Due to the prediction-correction nature, faster varying dynamics are able to be inferred. Although proven accurate on synthetic and real dynamic networks, the algorithm repeatedly solves iterative optimization problems at every time step. The computational complexity is therefore significant and can be increasingly an issue for larger graphs.

In this thesis, a different prediction–correction scheme is developed by formulating a linear graph state–space model, in the spirit of Alippi et al. [13], and applying a Kalman filter for recursive estimation of graph connectivity. The adjacency matrix is treated as the hidden state, graph signals form the observations, and the Kalman recursion provides closed-form prediction and correction updates that depend only on the most recent graph signal. This avoids repeated optimization, scales more efficiently than existing prediction-correction based algorithms, and remains well-suited for tracking fast time-varying network connectivity.

Independently, Dabush et al. introduced an extended Kalman filter for sparse dynamic GTI [14]. Their focus is on nonlinear dynamics with slow noise-driven evolution. In contrast, this work develops and tests a linear Kalman filter specifically designed and tested to handle fast time-varying networks. Further, a joint estimation scheme is developed to specifically learn the state dynamics along with the graph connectivity.

1.1 Thesis Outline and Contributions

This thesis develops a Kalman filter-based framework for GTI considering static and dynamic graph learning using online, streaming data, which is referred to as Kalman Dynamic Online GTI (KalDO GTI). Chapter 3 introduces the formulation in the static case and considers performance under diffusion, input–output data, and structural equation models. Chapter 4 extends the proposed algorithm to dynamic graphs with known transition dynamics. The algorithm is tasked with tracking evolving adjacency structures reliably under various state dynamics scenarios. Chapter 5 then addresses the more realistic scenario where the state dynamics itself are unknown, proposing a joint estimation scheme that combines structural priors, adaptive optimization, and retroactive smoothing to learn graph connectivity and state dynamics throughout time. Chapter 6 benchmarks the proposed framework against streaming LASSO and prediction–correction, highlighting KalDO's robustness under high noise. Together, these chapters establish Kalman filtering as a practical, scalable, and interpretable basis for

online graph topology inference and support applying such methods to real-world dynamic networks.

The main contributions are:

- Chapter 3 introduces a Kalman filter framework for static graph topology identification. Using a reduced–state representation that enforces symmetry and zero diagonals, the algorithm is shown to accurately recover adjacency matrices under several synthetic graph signal models.
- Chapter 4 extends this framework to dynamic graphs with known state dynamics. The recursive algorithm successfully tracks evolving adjacency matrices in synthetic experiments representing both gradual and abrupt structural changes.
- Chapter 5 develops a joint estimation method that simultaneously learns the adjacency matrix and the unknown state dynamics. With structural priors, adaptive optimization, and retroactive smoothing, the algorithm achieves stable and accurate long-horizon recovery in challenging dynamic regimes.
- Chapter 6 benchmarks the proposed approach against streaming LASSO and prediction–correction methods. Results show KalDO is competitive across regimes and particularly robust under high measurement noise.

2 Graph Inference Background

Representing networks mathematically through graph theory has a long history with work as far back as the 1700s [15]. Only in recent years has research shifted from describing graph structures to analyzing signals defined on them, giving rise to the field of Graph Signal Processing (GSP). GSP extends classical signal processing tools to networks, enabling inference of underlying graph connectivity directly from graph signals, node-level data. This field forms the basis for the algorithm proposed in this thesis. In what follows, the core ideas in graph learning are introduced alongside the historical development of dynamic GTI methods. This includes prediction–correction algorithms, and the graph state–space model perspective that anchors the proposed approach.

The chapter proceeds with Section 2.1 first introducing the foundations of GTI through statistical, physically motivated, and GSP or data–model based perspectives. Section 2.2 reviews static GTI methods, based on structural equation, diffusion, and input-output models, or using convex optimization programs. The section further highlights their reliance on large sample sets and fixed topologies. Section 2.3 covers dynamic GTI, focusing on windowed batch methods such as TV–GLASSO (Time-Varying Graphical Lasso) and temporal smoothness, as well as online prediction–correction algorithms. Section 2.5 presents graph state–space formulations, including Alippi's model and Dabush's Kalman and extended Kalman filters, and Section 2.6 positions the proposed approach between these extremes.

2.1 Background: Graph Learning and Graph Signal Processing

Graph topology identification (GTI) is defined as recovering the structure of a network from graph signals. A graph is denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{S})$, where $\mathcal{V}$ is the set of N nodes, $\mathcal{E}$ the edge set, and $\mathbf{S}$ a graph shift operator (GSO). The GSO could be, for example, the adjacency matrix $\mathbf{A}$ or graph Laplacian $\mathbf{L}$. Graph signals are vectors $\mathbf{y}_t \in \mathbb{R}^N$ indexed by time t, and collected in a data matrix $\mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_T] \in \mathbb{R}^{N \times T}$. The inference task is to estimate $\mathbf{S}$ given $\mathbf{Y}$, typically under the model

$$\mathbf{Y} = \mathcal{F}(\mathbf{S}), \tag{2.1}$$

where $\mathcal{F}$ denotes the generative mechanism linking graph signals to graph topology [4, 5, 10]. Because many $\mathbf{S}$ estimates could explain the same $\mathbf{Y}$, GTI can be ill-posed and could require additional assumptions or priors. This depends on the generative mechanism, amount of available samples, and whether GTI is under static or dynamic conditions. Approaches to GTI can generally be described as being statistically motivated, physically motivated, inspired by graph signal processing (GSP)

techniques or motivated by a specific generative data model. Some methods may combine elements from the aforementioned categories as well. Here, a brief overview.

Several broad families of GTI methods have been developed:

- **Statistical models:** infer edges from conditional dependencies among node signals, often via precision matrices or graphical Lasso (GLASSO) [4, 5].
- **Physically motivated models:** assume signals arise from spreading processes such as diffusion or cascades, common in epidemiology or social contagion [5, 4].
- **Graph signal processing (GSP) priors:** impose smoothness or spectral structure on signals, leading to convex formulations for learning $\mathbf{L}$ or $\mathbf{A}$ [4, 16].
- **Explicit data models:** e.g., structural equation, diffusion, or input–output models, which specify a direct functional relation between $\mathbf{S}$ and $\mathbf{Y}$ [4, 17].

The next section details representative methods for the static GTI setting.

2.2 Static GTI

We now focus on algorithms that dominate the static GTI literature. These can be grouped into explicit generative data models and convex optimization formulations which may also combine the broad families of GTI methods.

The first class of methods specifies an explicit generative link between observed signals and the unknown topology. The *Structural Equation Model (SEM)* assumes each observation $\mathbf{y}_t \in \mathbb{R}^N$ is a linear combination of neighboring values,

$$\mathbf{y}_t = \mathbf{A}\mathbf{y}_t + \mathbf{w}_t, \tag{2.2}$$

where $\mathbf{A}$ is the adjacency matrix and $\mathbf{w}_t$ is zero mean noise [5, 4]. Given observations $\{\mathbf{y}_t\}_{t=1}^T$, the adjacency is estimated by solving a least squares or regularized regression problem, enforcing symmetry and a zero diagonal when undirected graphs are desired.

The *diffusion model* instead assumes signals are generated through a polynomial graph filter,

$$\mathbf{y}_t = \sum_{l=0}^{L-1} h_l \mathbf{S}^l \mathbf{x}_t, \tag{2.3}$$

where $\{h_l\}$ are filter coefficients, $\mathbf{S}$ is the graph shift operator, and $\mathbf{x}_t \in \mathbb{R}^N$ denotes an excitation signal [4]. Recovery of $\mathbf{S}$ is typically posed as a spectral identification problem [5, 4]. Notably, the sample covariance of $\mathbf{y}_t$, $\hat{\mathbf{\Sigma}}$, shares eigenvectors with $\mathbf{S}$ under this model so an eigendecomposition of $\hat{\mathbf{\Sigma}}$ provides the eigenvectors of the graph shift operator. To find the eigenvalues, an optimization problem is solved using priors such as sparsity, smoothness, or nonnegativity of the edge weights.

A simpler variant of the diffusion model is the *input–output (IO) model*,

$$\mathbf{y}_t = \mathbf{A}\,\mathbf{x}_t + \mathbf{w}_t, \tag{2.4}$$

where $\mathbf{x}_t$ are known inputs and $\mathbf{y}_t$ are measured outputs [17]. In this case the adjacency is obtained by multivariate regression that minimizes the squared error between $\mathbf{y}_t$ and $\mathbf{A}\mathbf{x}_t$ with optional sparsity regularization.

A second major class of methods does not assume a detailed generative model but instead exploits statistical or structural priors. The popular GLASSO estimates the precision matrix $\mathbf{\Theta} = \mathbf{\Sigma}^{-1}$ under Gaussian assumptions by solving the ℓ_1 regularized maximum likelihood problem,

$$\hat{\mathbf{\Theta}} = \arg\min_{\mathbf{\Theta} \succ 0} \Big(-\log\det \mathbf{\Theta} + \operatorname{tr}(\hat{\mathbf{\Sigma}}\mathbf{\Theta}) + \lambda \|\mathbf{\Theta}\|_1 \Big), \tag{2.5}$$

where $\hat{\mathbf{\Sigma}} = \frac{1}{T}\mathbf{Y}\mathbf{Y}^\top$ is the sample covariance [5]. Edges are then placed wherever $\hat{\Theta}_{ij} \neq 0$.

Smoothness based approaches build instead on the prior that graph signals vary smoothly across connected nodes. This is quantified by the Laplacian quadratic form and recovery of a valid Laplacian is posed as,

$$\hat{\mathbf{L}} = \arg\min_{\mathbf{L} \in \mathcal{L}} \operatorname{tr}(\mathbf{L}\hat{\mathbf{\Sigma}}), \tag{2.6}$$

where $\mathcal{L}$ is the set of combinatorial Laplacians [4, 16]. This convex program yields a sparse Laplacian whose structure encourages low frequency representations in the graph Fourier basis.

Although widely applied, these methods share common limitations. Both generative model based and convex optimization approaches require large sample sets for stable estimation and they assume the graph is fixed in time. Windowed adaptations are often used to approximate dynamics but these cannot reliably track fast or abrupt changes [5, 10]. These shortcomings motivate the development of dynamic GTI methods, addressed in the following section.

2.3 Dynamic GTI

In real-world networks, connections evolve over time due to changing environments, shocks, or adaptations. Static inference assumes a fixed connectivity structure and requires large batches of independent samples. In practice, signals arrive sequentially and the network itself is dynamic, so static GTI cannot accurately capture evolving topologies.

A natural extension of static methods is to estimate a time-indexed sequence of graphs from windowed data. Signals $\{\mathbf{y}_t\}_{t=1}^T$ are partitioned into overlapping windows of length W, and for each window a static estimator is applied,

$$\hat{\mathbf{S}}_t = \arg\min_{\mathbf{S} \in \mathcal{S}} f(\mathbf{Y}_t, \mathbf{S}) + \lambda g(\mathbf{S}), \tag{2.7}$$

where $f(\cdot)$ is a data fidelity term, $g(\cdot)$ a structural prior such as sparsity or smoothness, $\mathbf{Y}_t = [\mathbf{y}_t, \ldots, \mathbf{y}_{t+W}]$, and $\mathcal{S}$ the feasible set of graph operators [17]. The result is a sequence $\{\hat{\mathbf{S}}_t\}$ indexed by time windows.

An example is the *time-varying graphical LASSO* (TVGL) [9], which estimates a sequence of precision matrices by solving,

$$\{\hat{\mathbf{\Theta}}_t\}_{t=1}^T = \arg\min_{\{\mathbf{\Theta}_t \succ 0\}} \sum_{t=1}^T \Big(-\log\det \mathbf{\Theta}_t + \operatorname{tr}(\hat{\mathbf{\Sigma}}_t \mathbf{\Theta}_t) + \lambda \|\mathbf{\Theta}_t\|_1 \Big) + \beta \sum_{t=2}^T \psi(\mathbf{\Theta}_t - \mathbf{\Theta}_{t-1}), \tag{2.8}$$

where $\hat{\mathbf{\Sigma}}_t$ is an empirical covariance from the t-th window $\mathbf{Y}_t$ and $\psi(\cdot)$ encodes temporal regularization. The penalty term encourages consecutive estimates to remain similar, with λ controlling sparsity and β controlling temporal consistency.

Kalofolias [16] proposed a related method based on smooth graph signals, learning adjacency matrices $\mathbf{A}_t$ by solving,

$$\{\hat{\mathbf{A}}_t\}_{t=1}^T = \arg\min_{\{\mathbf{A}_t \in \mathcal{A}\}} \sum_{t=1}^T \operatorname{tr}(\mathbf{Y}_t^\top \mathbf{L}(\mathbf{A}_t) \mathbf{Y}_t) + \beta \sum_{t=2}^T \|\mathbf{A}_t - \mathbf{A}_{t-1}\|_F^2, \tag{2.9}$$

where $\mathbf{L}(\mathbf{A}_t)$ is the Laplacian associated with $\mathbf{A}_t$ and β enforces smooth evolution. Dong et al. linked such methods to GSP smoothness priors [4], while Natali et al. generalized them into a composite optimization framework for different data models including the Gaussian Graphical Model (GGM), SEM, and smoothness-based models [17].

Although effective when networks vary slowly, batch dynamic GTI faces two key limitations. First, temporal resolution is traded for statistical reliability. Use of larger windows reduces the estimation variance but fails to capture high temporal resolution, so the estimator converges to the average graph over each window. Second, each update requires solving a convex optimization problem such as (2.8), which scales poorly with the number of nodes and windows. As a result, these methods are best suited for retrospective analysis or systems with slowly evolving topologies, motivating the development of online dynamic GTI algorithms.

2.4 Dynamic GTI via Online Optimization

A key step toward online dynamic GTI is the use of *prediction–correction* methods. Natali et al. proposed this framework for time-varying generative data models [12]. The graph learning problem is cast as a composite convex program with a data fidelity term and an ℓ_1 regularizer promoting sparsity.

At each time step, the update proceeds in two stages. The *prediction* step uses a local Taylor expansion of the cost function around the previous solution s_t to anticipate how the optimum will move as time advances. This expansion includes the gradient (first derivative) that indicates the slope of the cost, the Hessian (second derivative) that describes its curvature, and the time derivative of the gradient. Together, these terms shift the solution in the direction the optimizer is expected to travel, producing an estimate of the next time step $\hat{s}_{t+1|t}$.

The *correction* step then adjusts this estimate using a proximal gradient update based on the actual cost at time $t+1$. A proximal gradient update means taking a gradient step using f and then applying the proximal operator of the regularizer g. For

ℓ_1 penalties this proximal operator performs soft thresholding, which enforces sparsity in the estimated graph.

This scheme can track both abrupt and gradual temporal changes, and convergence is guaranteed under strong convexity and Lipschitz smoothness assumptions [17]. Experiments show that prediction–correction methods closely follow the offline batch solution with low error. The main drawback is computational. Each step requires solving an iterative convex subproblem which limits scalability to larger graphs. In addition, the method depends on a specified data model such as the GGM, SEM, or smoothness-based models, restricting applicability when such assumptions are not valid.

Lastly, although prediction–correction achieves low error relative to the batch solution, this similarity also highlights a limitation. Since the method extrapolates from the previous optimum using a local Taylor expansion, it implicitly assumes smooth temporal evolution. When the underlying graph varies more abruptly than the model can capture, the updates lag behind and short-lived changes are smoothed out. Thus, while effective for slowly varying networks, this method could potentially be less suited for identifying fast dynamics though it remains to be explicitly tested.

2.5 Graph State Space Formulation and Predecessor Algorithms

In addition to methods like Natali et al. to overcome the computational bottlenecks of batch dynamic GTI, recent work has turned to recursive estimation techniques such as the Kalman filter. This requires casting GTI into a state–space model. Alippi et al. [13] formalized this idea by introducing the *graph state–space model (graph SSM)*, where the vectorized adjacency is treated as the hidden state,

$$\mathbf{a}_t = \mathbf{F}_{t-1}\mathbf{a}_{t-1} + \boldsymbol{\eta}_{t-1}, \qquad \mathbf{y}_t = \mathbf{H}_t\mathbf{a}_t + \boldsymbol{\nu}_t, \tag{2.10}$$

with $\mathbf{a}_t = \mathrm{vec}(\mathbf{A}_t)$, $\mathbf{F}_{t-1}$ the state transition, $\mathbf{H}_t$ the system matrix from the chosen observation model, and $\boldsymbol{\eta}_{t-1}, \boldsymbol{\nu}_t$ process and measurement noise. Their work established this formulation as a natural fit for recursive filtering, proposing a *Graph Kalman Filter* that applies extended Kalman filtering to nonlinear state and readout functions. These functions are not specified analytically but instead learned with machine learning models and then linearized at each step through Jacobians. This demonstrated the feasibility of recursive GTI but did not commit to a closed-form or parameterized transition operator $\mathbf{F}_{t-1}$.

Building on this formulation, Dabush et al. (2024) [18] considered the simple case of a random–walk evolution,

$$\mathbf{a}_t = \mathbf{a}_{t-1} + \boldsymbol{\eta}_{t-1}, \tag{2.11}$$

and applied a standard Kalman filter to track $\mathbf{a}_t$ from sequential graph signals. Here dynamics are entirely noise driven, which yields efficient tracking and stable performance under gradual drift but cannot capture structured temporal dependencies.

In follow-up work, Dabush et al. (2025) [14] extended the framework to allow nonlinear state dynamics and proposed a sparsity-aware extended Kalman filter. Their

model takes the form,

$$\mathbf{a}_t = f_t(\mathbf{a}_{t-1}) + \boldsymbol{\eta}_{t-1}, \tag{2.12}$$

where $\mathbf{a}_t = \mathrm{vec}(\mathbf{A}_t)$ is the vectorized adjacency at time t, and $\boldsymbol{\eta}_{t-1}$ is process noise. The function $f_t(\cdot)$ can in principle describe nonlinear temporal evolution of the graph state, but in practice is often chosen as the identity mapping, corresponding to noise-driven dynamics. They use a sparsity-enforcing thresholding operator in the EKF correction step. While the framework supports nonlinear state dynamics, the transition function f_t is not learned from data and no structured transition operator $\mathbf{F}_{t-1}$ is estimated because the state dynamics are assumed to be noise-driven.

Together, these works demonstrated that Kalman filtering is a viable tool for dynamic GTI, but also highlighted a key limitation that current solutions operating in the graph state-space model treat graph dynamics as random drift, rather than parameterizing and learning a transition operator $\mathbf{F}_{t-1}$. This motivates the present work, where the state transition is explicitly modeled and estimated online along with dynamic graph connectivity tracking.

2.5.1 Graph SSMs and Causal Graphs

More recent work tackles the graph SSM from a different perspective where the adjacency is the state transition to be learned. In one of these methods, GraphEM, the state transition is updated by alternating between Kalman smoothing (E-step) and a proximal optimization (M-step) [19]. Another, DGLASSO, extends this by alternating block updates of the transition and noise-precision matrices via majorization–minimization [20]. Both approaches require repeated smoothing passes over the full dataset and are therefore batch methods posing limitations both computationally and in streaming settings.

2.6 Proposed Approach

The methods discussed so far reveal a tradeoff between expressivity and scalability. Batch methods such as the time-varying graphical LASSO or temporal smoothness priors [9, 16] can recover structured dynamics but scale poorly and are limited to retrospective analysis. Online prediction–correction methods [17] achieve accurate tracking with convergence guarantees, but each step involves solving an iterative convex subproblem. State–space formulations with Kalman filtering [13, 18, 14] are lightweight and recursive, but typically assume random-walk or purely noise-driven graph dynamics. Methods considering an alternative graph SSM have been proposed but still require all data to be available prior to processing and utilize computationally intensive methods making them strongly unsuitable for online, dynamic GTI [19, 20].

In contrast, the proposed method builds on the state–space perspective where the adjacency is vectorized as the hidden state but uses a lightweight recursive Kalman-based framework. The state dynamics are learned but lessen only the burden of learning the state, graph topology, without significant prior state-space knowledge. By combining short-horizon buffers with the recursive Kalman filter, the algorithm updates both

Table 2.1: Comparable Dynamic GTI algorithms.

Method	Model	Learns	Temporal	Online?	Notes
TV-GLASSO (§2.3)	GGM	precision matrices Θ_t	windowed and smoothness	×	Batch convex optimization, trades temporal resolution for lower variance.
Natali Pred. Corr. (§2.4)	SEM/ GGM	parameters per window	prediction-correction	✓	Online algo., each step is a convex subproblem with proximal updates, computationally intensive
Alippi/Dabush (§2.5)	Graph SSM	state vector $\mathbf{a}_t$ as adj.	Kalman / EKF	✓	Lightweight KF recursion but assumes noise-driven state dynamics.
GraphEM (§2.5.1)	LG-SSM	sparse state dynamics as adj.	EM with KF + RTS	×	Batch EM: E-step runs KF/RTS smoothing, M-step applies proximal updates for sparsity. Learns a latent Granger graph. Requires full dataset, not suited for streaming.
DGLASSO (§2.5.1)	LG-SSM	sparse state dyn. as ajd. & noise precision $\mathbf{P}$	block MM (proximal)	×	Jointly estimates dynamic Granger graph and static graphs with sparsity penalties. Uses block majorization–minimization (MM) with proximal steps and KF inference. Batch only.
KalDO GTI	Graph SSM	sparse state dyn. $\mathbf{F}$ and state vector $\mathbf{a}_t$ as adj.	Kalman with short buffers	✓	Online, streaming, no full smoothing, lightweight per-step updates.

the adjacency estimate and the state transition online, offering a scalable and streaming-compatible approach to dynamic GTI. The anticipated outcome is an algorithm that is perhaps less expressive than existing models but significantly more computationally efficient, more scalable to larger graphs, and better suited for real-time GTI tasks where recursive, online updates are required.

2.7 Conclusion

This chapter has presented a progression of approaches to graph topology identification starting from static GTI methods, to batch dynamic extensions, and finally to online prediction-correction schemes. Existing online approaches either impose strong model assumptions or impose high computational costs. The state-space perspective introduced by Alippi and specialized by Dabush provides a natural entry point for recursive filtering, but remains restricted to noise-driven dynamics.

KalDO GTI will extend state-space formulations into a recursive Kalman filter suitable for static and dynamic GTI. By combining the scalability of recursive filtering with

learnable structured state dynamics, it aims to offer a middle ground between an expressive but costly method such as Natali's, and lightweight but limited to noise-driven approaches such as Dabush et al.

3 Static Graph Inference

A Kalman filter-based algorithm is proposed for *static graph topology inference* (GTI) processing incoming graph signals in a recursive manner. It is tested across several data models relating graph connectivity to graph signals. The algorithm approaches the problem of GTI via recursive estimation of a graph's adjacency matrix. It operates using *streaming data*, incoming data measurements, processed as they arrive without necessitating reprocessing the previous measurements. Such an approach with online estimation of graph connectivity operates by formulation of a *graph state-space model* (graph SSM) introduced in Chapter 2, in which the adjacency matrix is treated as the hidden state. A Kalman filter is then employed to estimate this hidden state from observed signals [13, 14].

This chapter first formally introduces the graph SSM formulation and the specific static GTI case. The Kalman filter-based algorithm and its intuition are then discussed after a presentation of the key system assumptions. Next, various data models are presented with derivations of their system parameters. Finally, we describe evaluation metrics used to assess performance and present experimental results.

3.1 Problem Setting and Graph State-Space Formulation

Section 2.5 provides a short overview of graph SSMs (Equation 2.10). In this chapter, we use the same notation and further describe the static GTI scenario.

GTI involves inferring a fixed topology from observed graph signals. We frame the problem within a linear state–space model where the adjacency is the hidden state [18]. Let $\mathbf{A}_t \in \mathbb{R}^{N\times N}$ denote the adjacency and $\mathbf{a}_t = \mathrm{vec}(\mathbf{A}_t) \in \mathbb{R}^{N^2}$ its vectorization at time t. Throughout, $\mathbf{A}_t$ is weighted where binary weights in some tests are merely a special case. The general graph SSM reads,

$$\begin{aligned} \mathbf{a}_t &= \mathbf{F}_{t-1}\,\mathbf{a}_{t-1} + \boldsymbol{\eta}_{t-1} && \text{(state transition)} \\ \mathbf{y}_t &= \mathbf{H}_t\,\mathbf{a}_t + \boldsymbol{\nu}_t && \text{(observation)} \end{aligned} \tag{3.1}$$

where $\mathbf{y}_t \in \mathbb{R}^N$ is the observed graph signal, $\mathbf{H}_t \in \mathbb{R}^{N\times N^2}$ the measurement matrix (model–dependent), $\boldsymbol{\eta}_{t-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{t-1})$ the process noise, and $\boldsymbol{\nu}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_t)$ the measurement noise. We define $\mathbf{Q}_{t-1} = \sigma^2_{\text{proc.}}\mathbf{I}$ and $\mathbf{R}_t = \sigma^2_{\text{meas.}}\mathbf{I}$.

The static GTI is a special case of this graph SSM. There the hidden state does not evolve, i.e., $\mathbf{F}_{t-1} = \mathbf{I}$ and $\boldsymbol{\eta}_t = \mathbf{0}$, so Equation 3.1 for static GTI reduces to,

$$\begin{aligned} \mathbf{a}_t &= \mathbf{a}_{t-1} && \text{(state transition)} \\ \mathbf{y}_t &= \mathbf{H}_t\,\mathbf{a}_t + \boldsymbol{\nu}_t && \text{(observation)} \end{aligned} \tag{3.2}$$

This reduced form captures the static case and allows testing of the Kalman filter under idealized conditions where ample samples are available and there are no intrinsic

state dynamics. In this static chapter we use time-invariant covariances:$\mathbf{Q}_{t-1} = \mathbf{Q}$ and $\mathbf{R}_t = \mathbf{R}$. In all static experiments, $\mathbf{Q} = \mathbf{0}$ and $\mathbf{R} = \sigma^2_{\text{meas.}}\mathbf{I}$. When introducing the Kalman filter derivation, the time-varying notation will be used. However, this is removed when discussing the static algorithm.

3.2 Static GTI Model Assumptions

The development of the proposed inference algorithm relies on several key assumptions:

- The underlying graph is static (i.e., the adjacency matrix $\mathbf{A}$ does not vary over time), undirected, and connected. There is no process noise for the static GTI test scenario.
- The matrix $\mathbf{A}$ is unknown and assumed sparse, especially for larger graphs ($N > 20$), with few entries nonzero.
- Observations $\{\mathbf{y}_t\}$ are complete across all nodes in the network (complete, not partial information) and available at each time step.

These assumptions simplify the problem structure and enable more accurate recursive estimation using a Kalman filter.

3.3 General Kalman Filter Algorithm

The Kalman filter is a recursive algorithm for estimating the hidden state of a dynamic system in the presence of noise and uncertainty. For systems that can be expressed in linear state–space form with Gaussian process and measurement noise, it is proven to be the *optimal linear estimator*. It is defined as the estimator that minimizes the mean–squared error of the state estimate given all past measurements [21, 22]. Its structure alternates between prediction and correction in a recursive manner using only the last step variables. In the prediction step, the current state estimate and its uncertainty are projected forward through the state transition model which details state dynamics. In the correction step, this prediction is updated with new data upon arrival. The key mechanism balancing these two sources of information is the Kalman gain matrix, which adapts at every step according to the current confidence in the prior versus the incoming measurements. This makes the filter both fast and self–adjusting, and a natural choice for online graph topology inference.

Part 1: Prediction. The current a priori state estimate $\mathbf{a}_t^-$ and its covariance $\mathbf{\Sigma}_t^-$ are predicted from the previous a posteriori values $\mathbf{a}_{t-1}^+$ and $\mathbf{\Sigma}_{t-1}^+$ using the state evolution matrix $\mathbf{F}$,

$$\mathbf{a}_t^- = \mathbf{F}_{t-1}\mathbf{a}_{t-1}^+, \tag{3.3}$$

$$\mathbf{\Sigma}_t^- = \mathbf{F}_{t-1}\mathbf{\Sigma}_{t-1}^+\mathbf{F}_{t-1}^\top + \mathbf{Q}_{t-1}. \tag{3.4}$$

The covariance $\mathbf{\Sigma}_t^-$ quantifies the predicted uncertainty of the hidden state, which grows by the addition of process noise $\mathbf{Q}_t$. The predicted measurement under the

current observation model $\mathbf{H}_t$ is

$$\mathbf{y}_t^- = \mathbf{H}_t \mathbf{a}_t^-. \tag{3.5}$$

Part 2: Correction. Once the actual measurement $\mathbf{y}_t$ is available, the prediction is corrected. The Kalman gain,

$$\mathbf{K}_t = \mathbf{\Sigma}_t^- \mathbf{H}_t^\top \left(\mathbf{H}_t \mathbf{\Sigma}_t^- \mathbf{H}_t^\top + \mathbf{R}_t \right)^{-1} \tag{3.6}$$

balances prior uncertainty against measurement noise. A large gain reflects either high prior uncertainty or low measurement noise, which causes the filter to rely more heavily on the measurement. A small gain reflects the opposite. The posterior state and covariance are then updated as,

$$\mathbf{a}_t^+ = \mathbf{a}_t^- + \mathbf{K}_t \left(\mathbf{y}_t - \mathbf{y}_t^- \right), \tag{3.7}$$

$$\mathbf{\Sigma}_t^+ = \left(\mathbf{I} - \mathbf{K}_t \mathbf{H}_t \right) \mathbf{\Sigma}_t^- \left(\mathbf{I} - \mathbf{K}_t \mathbf{H}_t \right)^\top + \mathbf{K}_t \mathbf{R}_t \mathbf{K}_t^\top. \tag{3.8}$$

The difference $\mathbf{y}_t - \mathbf{y}_t^-$ is called the *innovation*, capturing the mismatch between the observed and predicted measurement. This is the value that is then multiplied by the Kalman gain and ultimately determines the final update to the state. The posterior covariance $\mathbf{\Sigma}_t^+$ expresses the remaining uncertainty after this update, and the recursion ensures that state estimates and uncertainties evolve consistently as new data arrive [21, 23, 22, 13].

3.4 Generative Models and Observation Matrix Design

Although the proposed algorithm is designed for real-world applications with streaming data, it is essential to first assess the static GTI Kalman Filter algorithm performance in controlled synthetic environments. Synthetic experiments enable a systematic evaluation of the Kalman filter's behavior under known ground truth conditions, free from external uncertainties.

This evaluation involves generating graph signals from a predefined data model with known underlying topologies. Therefore, the observation matrices, $\mathbf{H}_t$, are explicitly constructed based on the underlying data model. Recall, the state transition matrix is held constant as the identity matrix and the process noise covariance to zero to reflect the static GTI setting. With these adjustments, the Kalman filter can properly be evaluated in the static tracking task.

Three generative data models are considered in the synthetic testing of the Kalman Filter-based static GTI algorithm - the SEM, Diffusion, and Input-Output (IO) models. These are chosen for their prevalence in recent literature and ability to be intuitively understood.

3.4.1 SEM Data Model

The SEM formulation is briefly introduced in Section 2.2. Here, we derive its observation matrix representation for use in the Kalman filter framework. Starting from

$$\mathbf{y}_t = \mathbf{A}\mathbf{y}_t + \mathbf{w}_t, \tag{3.9}$$

where $\mathbf{A}$ is the adjacency matrix to be estimated and $\mathbf{w}_t \sim \mathcal{N}(0, \sigma_w^2 \mathbf{I})$ is additive noise [10, 4, 17]. We rearrange and apply the vectorization identity to obtain the linearized observation model used in this chapter.

Rearranging gives,

$$(\mathbf{I} - \mathbf{A})\mathbf{y}_t = \mathbf{w}_t \quad \Rightarrow \quad \mathbf{y}_t = (\mathbf{I} - \mathbf{A})^{-1}\mathbf{w}_t. \tag{3.10}$$

This exact relation is nonlinear in $\mathbf{A}$ due to the matrix inversion, which prevents direct use in a linear Kalman filter. To obtain a linear form, we start again from the original relation,

$$\mathbf{y}_t = \mathbf{A}\mathbf{y}_t + \mathbf{w}_t. \tag{3.11}$$

Applying the identity $\text{vec}(\mathbf{CXD}) = (\mathbf{D}^\top \otimes \mathbf{C})\text{vec}(\mathbf{X})$ with $\mathbf{C} = \mathbf{I}_N$ to $\text{vec}(\mathbf{I}_N \mathbf{A}\mathbf{y}_t)$ gives,

$$\mathbf{y}_t = (\mathbf{y}_t^\top \otimes \mathbf{I}_N)\,\text{vec}(\mathbf{A}) + \mathbf{w}_t. \tag{3.12}$$

Under this formulation the observation matrix becomes,

$$\mathbf{H}_t = \mathbf{y}_t^\top \otimes \mathbf{I}_N, \tag{3.13}$$

mapping the unknown graph structure into the measurement space at each time step ($\mathbf{H}_t \in \mathbb{R}^{N \times N^2}$) in line with Equation 3.3.

3.4.2 Diffusion Model

Graph signals can also be generated through diffusion dynamics, where heat, information, or influence spreads across the network [1, 5, 4]. The Diffusion model is introduced in Section 2.2. Here, we derive a linear observation form suitable for Kalman filtering. Under the Diffusion model, the signal is modeled as,

$$\mathbf{y}_t = e^{-\tau \mathbf{L}}\mathbf{x}_t + \mathbf{w}_t \tag{3.14}$$

with $\tau > 0$ a diffusion parameter, $\mathbf{L}$ the graph Laplacian, a known excitation vector which here we consider to be, $\mathbf{x}_t \sim \mathcal{N}(\mathbf{0}, \sigma_x^2 \mathbf{I}_N)$ with small $\sigma_x = 0.1$, and a Gaussian noise term $\mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, \sigma_w^2 \mathbf{I}_N)$. The graph signal has one entry per node in the network $\mathbf{y}_t \in \mathbb{R}^N$. Since $e^{-\tau \mathbf{L}}$ is nonlinear in $\mathbf{L}$, the first-order approximation is used,

$$e^{-\tau \mathbf{L}} \approx \mathbf{I}_N - \tau \mathbf{L}. \tag{3.15}$$

Substituting the Laplacian definition $\mathbf{L} = \mathbf{D} - \mathbf{A}$ gives,

$$\mathbf{y}_t \approx \tau \mathbf{A}\mathbf{x}_t + \mathbf{w}_t, \tag{3.16}$$

where $\mathbf{w}_t$ absorbs the leftover degree terms and approximation error. Vectorizing yields,

$$\mathbf{y}_t \approx (\tau \mathbf{x}_t^\top \otimes \mathbf{I}_N)\text{vec}(\mathbf{A}) + \mathbf{w}_t. \tag{3.17}$$

With $\tau = 1$, the observation matrix becomes,

$$\mathbf{H}_t = \mathbf{x}_t^\top \otimes \mathbf{I}_N. \tag{3.18}$$

A drawback of this approach, however, is that we often do not know the generation model of the input exactly, which can lead to model misspecification. As with the SEM model, $\mathbf{H}_t \in \mathbb{R}^{N \times N^2}$.

3.4.3 Input–Output Model

An alternative setting assumes known input signals drive the network [24, 10]. The IO model is briefly summarized in Section 2.2. Here, we use it to derive the observation matrix for the Kalman update. With known inputs $\mathbf{x}_t$, the observations follow,

$$\mathbf{y}_t = \mathbf{A}\,\mathbf{x}_t + \mathbf{w}_t, \tag{3.19}$$

where $\mathbf{x}_t \in \mathbb{R}^N$ is a known excitation vector and $\mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, \sigma_w^2 \mathbf{I}_N)$ with $\mathbf{w}_t \in \mathbb{R}^N$ the measurement noise. For these experiments, we consider the known excitation vector to be white Gaussian noise. Vectorization gives,

$$\mathbf{y}_t = (\mathbf{x}_t^\top \otimes \mathbf{I}_N)\,\mathrm{vec}(\mathbf{A}) + \mathbf{w}_t, \tag{3.20}$$

so the observation matrix is,

$$\mathbf{H}_t = \mathbf{x}_t^\top \otimes \mathbf{I}_N. \tag{3.21}$$

Accurate recovery of the underlying graph under the IO model requires that the matrix $\sum_t \mathbf{x}_t \mathbf{x}_t^\top$ is full rank, ensuring identifiability of the solution. Similar to SEM and Diffusion models, $\mathbf{H}_t \in \mathbb{R}^{N \times N^2}$.

Note on Convergence. For any of the static data generation models considered (SEM, Diffusion, and IO), the linear Gaussian formulation implies that the Kalman filter estimate converges to the posterior mean. This indicates that in the static case, the recursive estimate asymptotically reaches the batch least squares solution, as shown in Appendix A.

These generative data models are used to synthetically generate data to test the Kalman Filter for Static GTI. Their formulas are also used to generate the observation matrices. Recall, $\mathbf{Q}$ and $\mathbf{R}$ remain fixed over time. In the static scenario specifically, $\mathbf{Q} = 0$. The full algorithm is provided in Algorithm 1.

3.5 Evaluation Metrics

A good estimation algorithm is one that faithfully reconstructs the parameter of interest. The match between estimated and ground truth adjacency can be defined using the Frobenius norm. For a matrix, this is defined as,

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2}, \tag{3.22}$$

and it quantifies the overall magnitude or "energy" of the matrix by summing the squared contributions of all its elements. Therefore, measuring the Frobenius norm between the estimated and true adjacency matrices, $||\mathbf{A} - \hat{\mathbf{A}}||_F$, measures how well the estimated graph matches the ground truth at each step. It captures the overall difference in energy. A Relative Error of 0 indicates a perfect match.

When the networks become larger, this difference scales larger accordingly. To enable a quick comparison in reconstruction capability across networks of different size,

Algorithm 1 Kalman Filter for Static Graph Topology Identification (GTI)

Input: Number of nodes N, Time horizon T, Observation noise variance σ^2, Data model: SEM, Diffu, or IO, Observed signals $\{\mathbf{y}_t\}_{t=1}^T$ (and $\{\mathbf{x}_t\}$ if IO)
Initialization: $\mathbf{a}_0^+ \leftarrow \mathbf{0}$, $\mathbf{\Sigma}_0^+ \leftarrow \alpha\mathbf{I}$ (Large α for high uncertainty), $\mathbf{Q} \leftarrow \mathbf{0}$, $\sigma \leftarrow 0.1$, $\mathbf{R} \leftarrow \sigma^2_{\text{meas.}}\mathbf{I}$
for $t = 1$ to T **do**
    if SEM **then**
        $\mathbf{H}_t \leftarrow \mathbf{y}_t^\top \otimes \mathbf{I}_N$
    else if Diffusion model **then**
        $\mathbf{H}_t \leftarrow \mathbf{x}_t^\top \otimes \mathbf{I}_N$
    else if IO model **then**
        $\mathbf{H}_t \leftarrow \mathbf{x}_t^\top \otimes \mathbf{I}_N$
    end if
    Prediction: $\mathbf{a}_t^- \leftarrow \mathbf{a}_{t-1}^+ \qquad \mathbf{\Sigma}_t^- \leftarrow \mathbf{\Sigma}_{t-1}^+$
    Correction:

$$\mathbf{y}_t^- \leftarrow \mathbf{H}_t\mathbf{a}_t^-, \qquad \mathbf{S}_t \leftarrow \mathbf{H}_t\mathbf{\Sigma}_t^-\mathbf{H}_t^\top + \mathbf{R}, \qquad \mathbf{K}_t \leftarrow \mathbf{\Sigma}_t^-\mathbf{H}_t^\top\mathbf{S}_t^{-1},$$

$$\mathbf{a}_t^+ \leftarrow \mathbf{a}_t^- + \mathbf{K}_t(\mathbf{y}_t - \mathbf{y}_t^-), \qquad \mathbf{\Sigma}_t^+ \leftarrow (\mathbf{I} - \mathbf{K}_t\mathbf{H}_t)\mathbf{\Sigma}_t^-(\mathbf{I} - \mathbf{K}_t\mathbf{H}_t)^\top + \mathbf{K}_t\mathbf{R}\mathbf{K}_t^\top$$

end for
Output: Estimated adjacency matrix $\hat{\mathbf{A}} = \text{unvec}(\mathbf{a}_T^+)$

the difference metric is normalized by the energy of the ground truth adjacency matrix, $||\mathbf{A}||_F$. This helps differentiate between absolute and relative estimation errors. The relative error metric is,

$$\text{RelError}(\mathbf{A}, \hat{\mathbf{A}}) = \left(\frac{\|\mathbf{A} - \hat{\mathbf{A}}\|_F^2}{\|\mathbf{A}\|_F^2} \right), \tag{3.23}$$

where lower values indicate more accurate reconstruction.

3.6 Results

Experimental evaluation of the Kalman filtering approach in the static GTI setting under three data models are provided. The focus is primarily on small networks (e.g., $N = 10$), as this setting allows clearer visualization of adjacency matrix reconstructions. It also directly highlights the influence of varying sample sizes T, which empirically show the largest effect on estimation performance. Varying the number of nodes N would have instead the most significant impact seen in its convergence speed, a consideration for later.

The adjacency matrix is generated from an Erdős–Rényi model $\mathbf{A} \sim \mathcal{G}(N, p)$ with $p = 0.1$ and resampled until the graph is connected. This yields a sparse but connected ground truth. The initial estimate is set to $\hat{\mathbf{A}}_0 = \mathbf{0}_{N\times N}$.

For each model, heatmaps of the true and estimated adjacency matrices are shown across t where the maximum time step is $10,000$, alongside the corresponding relative

error and estimated sparsity. Additionally, the relative error trajectory is more directly seen over time in the second plot provided per experiment.

At early iterations, spikes, increases in the relative error, may be present. With an uninformative prior ($\mathbf{\Sigma}_0^+$ set large), the Kalman filter initially assigns high uncertainty to all directions of the state. Early measurements may also be nearly linearly dependent causing large Kalman gains and unstable corrections. As more independent samples arrive, the measurement space fills its true size and the filter gain stabilizes.

3.6.1 Using the SEM Data Model

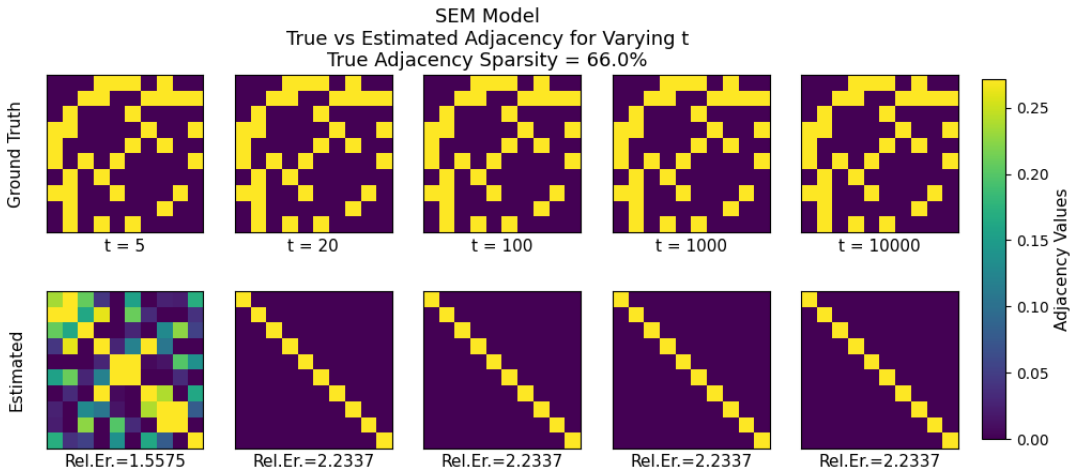


Figure 3.1: The SEM model in the static GTI setting is *unable* to reconstruct the desired adjacency matrix, as confirmed visually across timesteps. Rather, the estimated adjacency becomes the identity matrix matching the least-squares solution

As shown in Figure 3.1, reconstruction under the SEM model is quite poor. Even as t increases, the relative error remains around 2.2 and does not decrease as supported by Figure 3.2. This is because the estimated adjacency converges to the identity matrix. This outcome arises because static GTI with a Kalman filter in this formulation is equivalent to recursive least-squares estimation of $\mathbf{A}$, where each update incrementally minimizes the reconstruction error $\|\mathbf{y}_t - \mathbf{A}\,\mathbf{y}_t\|_2^2$. Given the SEM data model, the identity matrix $\mathbf{A} = \mathbf{I}$ becomes a trivial minimizer. It reproduces the data perfectly in the noiseless limit, since $\mathbf{y}_t \approx \mathbf{I}\,\mathbf{y}_t$. This non-uniqueness causes the estimator to converge to this trivial solution, even though it does not reflect the true underlying graph.

Even with $\mathrm{diag}(\mathbf{A}) = 0$ enforced, the Kalman filter does not yield the correct adjacency matrix. The collapse to the identity reflects a trivial minimizer of the least-squares cost. Even if this solution is excluded, the SEM remains unidentifiable, meaning that different adjacency matrix solutions can explain the same observed signals. To obtain a unique solution, additional structure such as external excitation, stronger priors, or dynamic formulations is required [25].

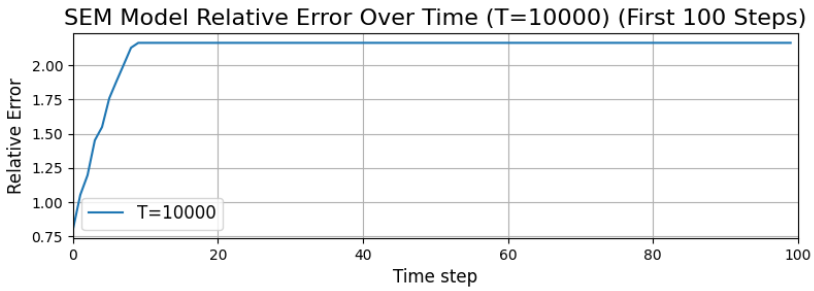


Figure 3.2: The Relative Error for the SEM Model in the Static GTI setting severely inflates and stabilizes at a high value reflecting poor reconstruction.

3.6.2 Using the Diffusion Data Model

Empirically, the Diffusion model achieves accurate and rapid reconstruction of the adjacency matrix. Already at $t = 20$, the relative error drops low, and by $t = 100$ it decreases further, see Figure 3.3. Visually, the reconstructed adjacency resembles the ground truth and performance continues to improve steadily with larger t.

As shown in Figure 3.4, the relative error trajectory confirms this trend: across all values of t, the error decreases smoothly toward a small value, reflecting convergence of the estimator over time. There are some small spikes in error described at the beginning of this section, after which the error stabilizes and steadily decreases.

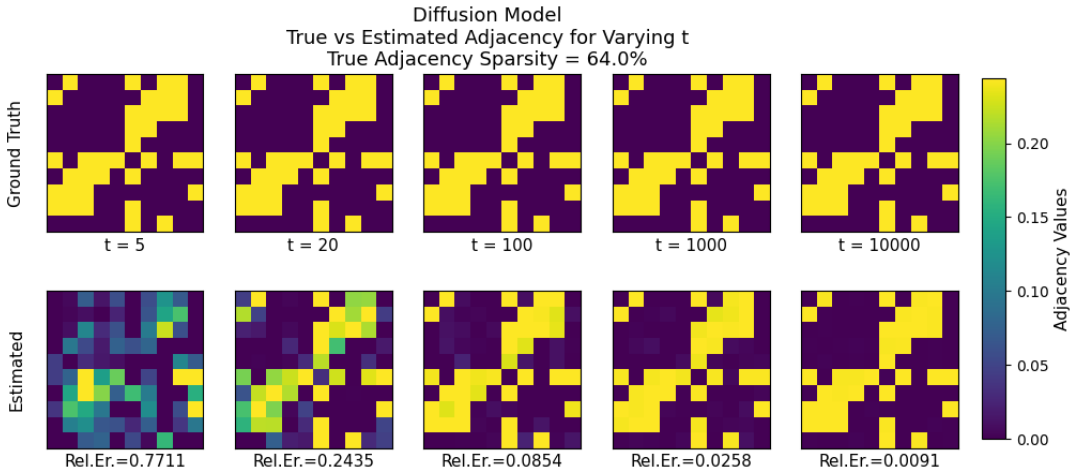


Figure 3.3: The Diffusion model in the Static GTI setting is able to accurately reconstruct the desired adjacency matrix as confirmed visually across timesteps.

3.6.3 Using the Input/Output Model

Recall that in the IO model, each input vector (which we assume white noise here, $\mathbf{x}_t \sim \mathcal{N}(0, \sigma_x^2)$) excites the network. With an input vector of white noise, the input

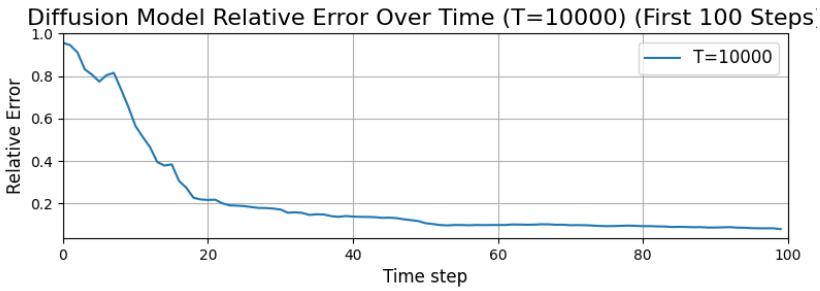


Figure 3.4: The Relative Error for the Diffusion Model in the Static GTI setting *readily decreases to a small value*, a desired outcome, confirming afore displayed visual results.

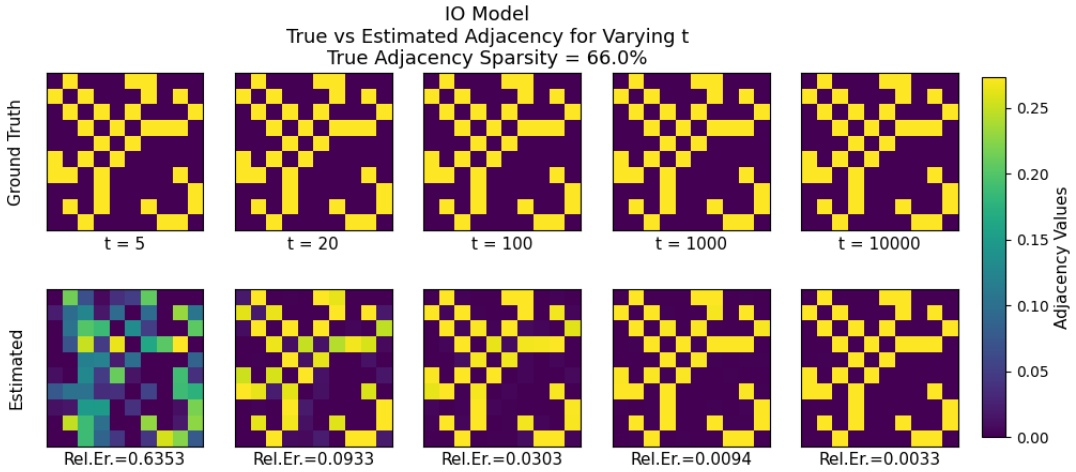


Figure 3.5: The IO model in the Static GTI setting is able to reconstruct the desired adjacency matrix. As the number of timesteps increases, the relative error decreases.

excites all modes of the system [26]. Therefore, as the input gets pushed into the adjacency matrix, the measurement $\mathbf{y}_t$ carries significant information about $\mathbf{A}$. This fosters consistent recovery of the adjacency matrix $\mathbf{A}$.

This is confirmed by an adjacency that is very quickly learned. At $t = 5$ the estimate already exhibits the correct sparsity pattern, and by $t = 20$ the relative error has dropped to 0.09, producing a visually accurate reconstruction. The relative error trajectory in Figure 3.6 shows the same sharp decrease seen in the Diffusion model, leveling off around 12–15 steps as the system approaches the true adjacency.

3.6.4 Summary Observations Across Models

Several general trends emerge from these static GTI experiments:

- The IO model consistently achieves the most accurate and stable graph recovery, due to fully independent excitation at each time step.
- The Diffusion model performs slightly worse, likely due to its first-order approxi-

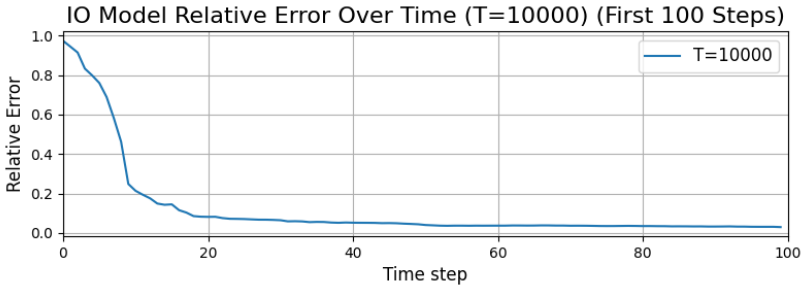


Figure 3.6: The Relative Error for the IO Model in the Static GTI setting *readily decreases to a small value*, a desired outcome, confirming afore displayed visual results.

mation required to be applied to the Kalman-based approach.

- The SEM model struggles significantly more due to the non-identifiability. Even with the reduced state space, it still cannot find the true solution. Findings are consistent with Han et al.[25].
- IO and Diffusion models show improved recovery as t increases.
- Lower noise levels (higher SNR) universally reduce estimation error, as the Kalman update places greater weight on more reliable measurements. The improvement is most pronounced in the IO and Diffusion models, while SEM remains fundamentally limited by non-identifiability rather than noise level.

These findings highlight how the fundamental excitation structure of each generative model governs its complexity and recoverability properties in static GTI settings.

3.7 Reduction of Hidden State: Zero-Diagonal and Symmetry

In many applications, the adjacency matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$ is known to have additional structure. For undirected graphs, $\mathbf{A}$ is symmetric and its diagonal is zero, i.e. $A_{ii} = 0$ and $A_{ij} = A_{ji}$. Enforcing these constraints reduces the number of free parameters from N^2 to $\frac{N(N-1)}{2}$. Thus, instead of working with the full vectorized adjacency $\mathbf{a}_t = \mathrm{vec}(\mathbf{A}_t) \in \mathbb{R}^{N^2}$, we define a reduced hidden state $\mathbf{u}_t \in \mathbb{R}^{d_{\mathrm{red}}}$. The relation is

$$\mathbf{u}_t = \mathbf{D}\,\mathbf{a}_t, \qquad \mathbf{a}_t = \mathbf{E}\,\mathbf{u}_t, \tag{3.24}$$

where $\mathbf{D}$ removes redundant entries and $\mathbf{E}$ reconstructs the full adjacency. Explicit constructions of $\mathbf{D}$ and $\mathbf{E}$, together with illustrative examples, are provided in Appendix B.

With this reduction, all prediction and update steps of the Kalman filter are carried out in the lower-dimensional space $\mathbb{R}^{d_{\mathrm{red}}}$. The observation matrix becomes $\mathbf{H}_t^{\mathrm{red}} = \mathbf{H}_t\mathbf{E}$, and the state transition is $\mathbf{F}_{\mathrm{red}} = \mathbf{DFE}$. The adjacency estimate is only expanded back to full form after filtering via $\hat{\mathbf{A}}_t = \mathrm{unvec}(\mathbf{E}\hat{\mathbf{u}}_t)$. This ensures that zero diagonals and symmetry are satisfied by construction, while reducing the number of parameters to be estimated.

The effect of working in reduced coordinates is evaluated in the experiments below, where we compare estimation accuracy and stability first considering only imposing the zero-diagonal on the adjacency and then imposing both that and the symmetry constraint together.

The estimation results show strong improvements in stability and convergence for IO and Diffusion models as both priors are implemented. The relative error reduces slightly for the zero diagonal and significantly for the symmetry prior. As shown in Figure 3.7, the estimated adjacency matrices recover the undirected structure with a lower relative error.

Under these reductions, the SEM estimate is non-identity. However, identifiability issues persist.

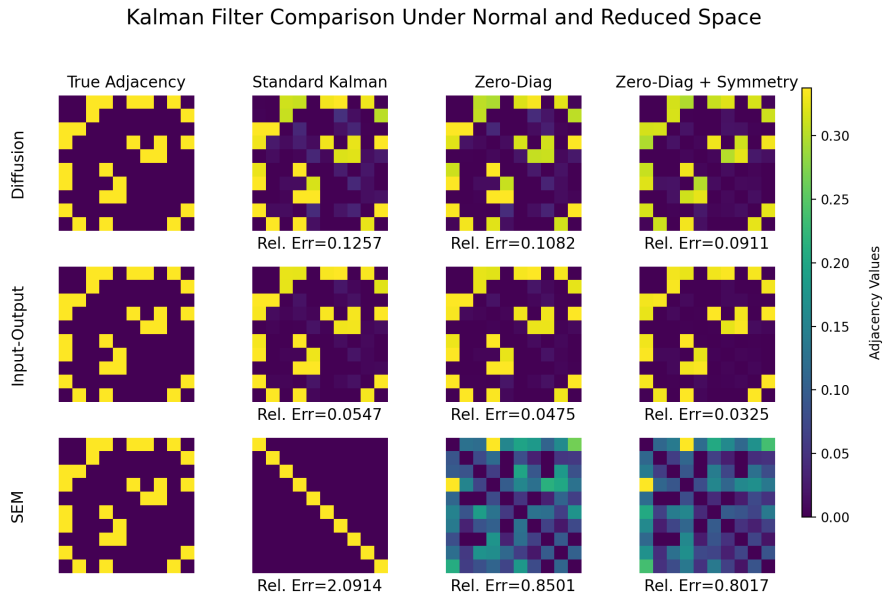


Figure 3.7: Estimated adjacency matrices, $N = 10$ and $T = 50$. Estimation accuracy improves when tracking in the reduced space for both types of state space reductions.

3.8 Conclusion

The IO model consistently demonstrates superior performance across all static graph topology identification (GTI) experiments. Unlike the Structural Equation Model (SEM), which fails to reliably recover the ground truth adjacency and exhibits high estimation variability, the IO model offers a more stable and interpretable formulation. While the Diffusion model also captures the correct dynamics, it can be viewed as a constrained variant of the IO model. Specifically, the transformation $\mathbf{Y} = (\mathbf{I} - \tau(\mathbf{D} - \mathbf{A}))$ encodes a diffusion operator implicitly governed by $\mathbf{A}$. Using the first-order approximation $e^{-\tau \mathbf{L}} \approx \mathbf{I} - \tau \mathbf{L}$, the Diffusion model reduces to the same input–output form $\mathbf{y}_t \approx \tau \mathbf{A} \mathbf{x}_t + \mathbf{w}_t$, which explains why the IO setting consistently achieves the strongest recovery.

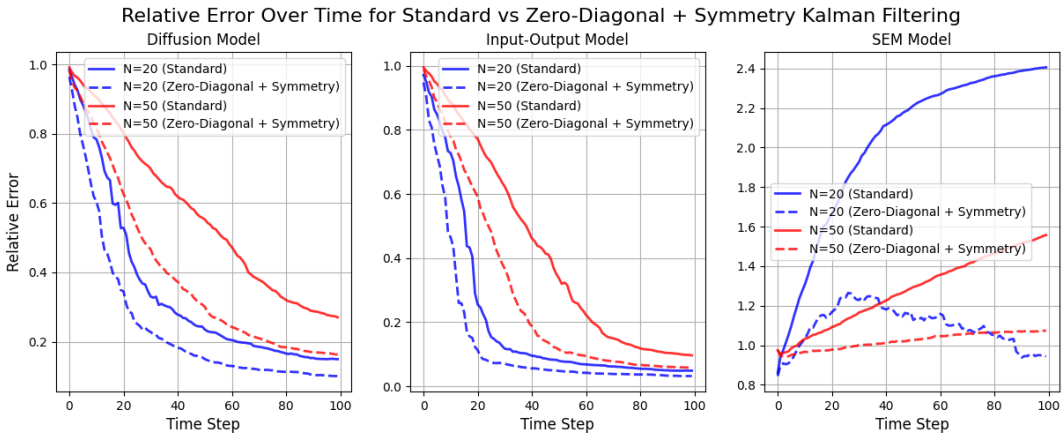


Figure 3.8: Demonstrates faster convergence and better relative error in the Diffusion and IO models. The SEM, not identifiable, still struggles.

To extend this framework to dynamic graphs, we generalize the hidden state to a time-varying adjacency vector. In this setting, a transition matrix $\mathbf{F}$ governs the evolution of the adjacency matrix over time. The graph under these conditions no longer remains fixed. This opens the door to modeling various types of graph dynamics, which could include identity drift, regime switching, and sparse local updates. These dynamic extensions are the focus of the following chapter.

Summary of Key Conclusions

- Static graph topology can be reliably tracked using a graph state space model and Kalman Filter algorithm.
- Compressing the parameter space (using assumptions of undirectedness and no self-loops) improves estimation.
- Findings suggest that the Kalman filter with a graph state–space model is a promising avenue, motivating the dynamic extensions in the next chapter.

4 Dynamic Graph Inference

The Kalman filter-based algorithm presented in Chapter 3 is extended here to the dynamic graph topology inference (GTI) setting, where the underlying graph is allowed to evolve over time. This transforms the problem from estimating a fixed topology to tracking a sequence of time-varying adjacency matrices using sequential graph signal observations.

To approach the dynamic GTI problem, the same graph state-space model now in dynamic form is cast, enabling online updates of the topology estimate as new data arrive. The Kalman Filter-based tracking algorithm, introduced formally in Chapter 3, is used on the data to find the time-varying adjacency matrix. The approach is tested under many state dynamic scenarios, represented by different state transition models, and its performance is evaluated in terms of tracking accuracy.

4.1 Dynamic Graph State-Space Model

We build directly on the graph state-space model (GSSM) defined in Chapter 3, where the adjacency is treated as the hidden state. In the static case, the state does not evolve, and the state dynamics reduce to the identity for the transition matrix. In the dynamic GTI setting considered here, the adjacency $\mathbf{A}_t \in \mathbb{R}^{N\times N}$ is allowed to vary over time and is tracked sequentially.

The adjacency evolves through time via a state transition matrix. This matrix, $\mathbf{F}$, is notably fixed across time. This assumption is common in dynamic state estimation problems where the underlying physical law of evolution is constant [22]. Anderson and Moore describe the example of target tracking with constant-velocity dynamics. There, $\mathbf{F}$ captures the fixed mapping of position and velocity between time steps. Uncertainty in maneuvering is handled through the process noise covariance $\mathbf{Q}$. In our case, $\mathbf{F}$ analogously encodes the stable part of edge evolution, with variability handled by $\mathbf{Q}$. This work strictly considers a fixed evolution matrix with a time-varying evolution scenario left for future work.

All estimation is carried out in reduced space. Let $\mathbf{u}_t \in \mathbb{R}^{N(N-1)/2}$ denote the reduced hidden state, i.e., the vector of unique off-diagonal entries of $\mathbf{A}_t$. The expansion matrix $\mathbf{E} \in \mathbb{R}^{N^2\times N(N-1)/2}$ maps $\mathbf{u}_t$ back to the full adjacency vector $\mathbf{a}_t = \mathbf{E}\mathbf{u}_t \in \mathbb{R}^{N^2}$, which enforces symmetry and a zero diagonal by construction. These details are given in Chapter 3 and supported by Appendix B.

The dynamic GTI problem is therefore cast as the GSSM,

$$\begin{aligned} \mathbf{u}_t &= \mathbf{F}\,\mathbf{u}_{t-1} + \boldsymbol{\eta}_{t-1} \qquad \text{(state transition)} \\ \mathbf{y}_t &= \mathbf{H}_t\mathbf{E}\,\mathbf{u}_t + \boldsymbol{\nu}_t \qquad \text{(observation)} \end{aligned} \tag{4.1}$$

where $\mathbf{u}_t \in \mathbb{R}^{\frac{N(N-1)}{2}}$ is the reduced hidden state, $\mathbf{F} \in \mathbb{R}^{\frac{N(N-1)}{2}\times\frac{N(N-1)}{2}}$ is the state tran-

sition matrix, $\boldsymbol{\eta}_{t-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$ is process noise with fixed covariance $\mathbf{Q} = \sigma^2_{\text{proc.}}\mathbf{I} \in \mathbb{R}^{\frac{N(N-1)}{2} \times \frac{N(N-1)}{2}}$, $\mathbf{H}_t \in \mathbb{R}^{N \times N^2}$ is the observation matrix, $\mathbf{E} \in \mathbb{R}^{N^2 \times \frac{N(N-1)}{2}}$ is the expansion matrix, and $\boldsymbol{\nu}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ is measurement noise with fixed covariance $\mathbf{R} = \sigma^2_{\text{meas.}}\mathbf{I} \in \mathbb{R}^{N \times N}$.

For notational clarity, we write the state dynamics and process noise in the compact form of Equation 4.1 without reduced subscripts or superscripts. In practice, both $\mathbf{F}$ and $\boldsymbol{\eta}_{t-1}$ (and hence $\mathbf{Q}$) operate in the reduced dimension $N(N-1)/2$. This compressed representation lowers the dimensionality of the state, enforces structural constraints automatically, and keeps the mathematics concise.

4.1.1 Model Assumptions

The following assumptions are made in the dynamic GTI setting:

- The graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$, where $\mathbf{S} = \mathbf{A}$ is connected. The adjacency is undirected $\mathbf{A} = \mathbf{A}^\top$, weighted, and satisfies $a_{ii} = 0 \ \forall i$ (no self-loops).
- The graph at each time step is *sparse*, i.e., most entries of $\mathbf{A}_t$ are zero.
- The *graph signal* $\{\mathbf{y}_t\}_{t=1}^T$ is *fully observed*.
- The evolution of $\mathbf{A}_t$ can be modeled as a *linear* process with additive *Gaussian noise*.

4.2 Dynamic Graph Topology Tracking with the Proposed Kalman Filter Algorithm

This algorithm performs online tracking of a time-varying graph topology (the adjacency matrix here) using a Kalman filter, assuming the state transition matrix $\mathbf{F}$ is known. All operations are carried out in the reduced hidden state $\mathbf{u}_t$ introduced in Section 3.7, with $\mathbf{F}$ denoting the reduced transition matrix for notational clarity. The observation matrix $\mathbf{H}_t$ is generated directly from the chosen data model (Input-Output) in the reduced dimension. The full adjacency vector is only reconstructed after estimation through the expansion matrix $\mathbf{E}$. The algorithm begins from a high-uncertainty prior, represented by an all-zero initial state with large covariance, and proceeds with the standard Kalman prediction–correction recursion.

4.2.1 Expected Tracking Capabilities

As introduced in Section 3.3, the Kalman filter alternates between prediction and correction. In a linear state-space system with Gaussian process and measurement noise, this recursion achieves the minimum mean-squared error (MMSE) providing the optimal estimator for linear systems with Gaussian noise [22].

In graph topology inference, however, these assumptions hold only approximately. The hidden state, the vectorized adjacency, is typically sparse or even binary in simple cases, which is not Gaussian. In this case, the Kalman filter still produces the linear

Algorithm 2 Kalman Filter for Online Dynamic Graph Topology Tracking

Input: Observations $\{\mathbf{y}_t\}_{t=1}^T$, Expansion matrix $\mathbf{E}$, State transition matrix $\mathbf{F}$
Initialize: $\hat{\mathbf{u}}_0^+ \leftarrow \mathbf{0}$, $\mathbf{\Sigma}_0^+ \leftarrow \alpha\mathbf{I}$, $\mathbf{Q} \leftarrow \sigma^2_{\text{proc.}}\mathbf{I}$, $\mathbf{R} \leftarrow \sigma^2_{\text{meas.}}\mathbf{I}$ ▷ No state prior
for $t = 1$ to T **do**
  Calculate $\mathbf{H}_t \leftarrow \mathbf{x}_t^\top \otimes \mathbf{I}_N$
  Prediction:
    $\hat{\mathbf{u}}_t^- \leftarrow \mathbf{F}\hat{\mathbf{u}}_{t-1}^+$
    $\mathbf{\Sigma}_t^- \leftarrow \mathbf{F}\mathbf{\Sigma}_{t-1}^+\mathbf{F}^\top + \mathbf{Q}$
  Correction:
    $\mathbf{K}_t \leftarrow \mathbf{\Sigma}_t^- \left(\mathbf{E}^\top\mathbf{H}_t^\top\right)\left(\mathbf{H}_t\mathbf{E}\mathbf{\Sigma}_t^-\mathbf{E}^\top\mathbf{H}_t^\top + \mathbf{R}\right)^{-1}$
    $\hat{\mathbf{u}}_t^+ \leftarrow \hat{\mathbf{u}}_t^- + \mathbf{K}_t\left(\mathbf{y}_t - \mathbf{H}_t\mathbf{E}\hat{\mathbf{u}}_t^-\right)$
    $\mathbf{\Sigma}_t^+ \leftarrow \left(\mathbf{I} - \mathbf{K}_t\mathbf{H}_t\mathbf{E}\right)\mathbf{\Sigma}_t^-\left(\mathbf{I} - \mathbf{K}_t\mathbf{H}_t\mathbf{E}\right)^\top + \mathbf{K}_t\mathbf{R}\mathbf{K}_t^\top$
  Reconstruct: $\hat{\mathbf{a}}_t \leftarrow \mathbf{E}\hat{\mathbf{u}}_t^+$
end for
Output: $\{\hat{\mathbf{a}}_t\}_{t=1}^T$

minimum-variance estimate but it is no longer globally MMSE. It can be viewed as an approximation to the MMSE estimator for the graph SSM [22].

Within this approximate setting, tracking performance is primarily influenced by three factors:

- **Asymptotic Stability.** In the time-varying case, a simple strong way to guarantee an asymptotically stable Kalman filter is to verify observability of $(\mathbf{F}, \mathbf{H}_t)$ [22]. Observability means that the hidden state can be uniquely determined from a finite sequence of outputs [26]. For observability, form the stacked matrix,

$$\mathcal{O}_{t,L} = \begin{bmatrix} \mathbf{H}_t \\ \mathbf{H}_{t+1}\mathbf{F} \\ \cdots \\ \mathbf{H}_{t+L-1}\mathbf{F}^{L-1} \end{bmatrix} \tag{4.2}$$

  and check full rank for every t over a fixed window L. With a satisfied observability condition, the Kalman Filter has asymptotic stability. The additional controllability requirement for asymptotic stability is skipped as there are no control inputs to the GSSM [27]. When these checks hold, the covariance stays bounded and the estimation error decays. Observability is explored for the given test scenarios in Section 4.4.

- **Process noise strength.** The matrix $\mathbf{Q}$ specifies the covariance of the process noise and thus encodes how much uncertainty is injected into the state at every prediction step. It appears directly in the covariance recursion which feeds into the Kalman gain. A larger $\mathbf{Q}$ therefore increases the gain, making the filter more responsive to new data but also raising the variance of the estimates. Conversely, a smaller $\mathbf{Q}$ enforces smoother trajectories but reduces the ability to adapt quickly to sudden changes.

- **Accurate modeling.** In the experiments in this chapter, the true $\mathbf{F}$ is used to evaluate tracking performance. $\mathbf{H}_t$ is generated based on the true IO model and $\mathbf{Q}$ and $\mathbf{R}$ are also set to their ground-truth values unless explicitly stated otherwise.

4.2.2 Managing Noise Accumulation

Errors can accumulate across time steps in a dynamic network. The Kalman filter models this explicitly through the prediction and correction steps in Algorithm 2. In prediction, past uncertainty is propagated and process noise is added via the current state covariance matrix,

$$\mathbf{\Sigma}_t^- = \mathbf{F}\mathbf{\Sigma}_{t-1}^+\mathbf{F}^\top + \mathbf{Q}. \tag{4.3}$$

In correction, the new measurement adjusts the state estimate and shrinks uncertainty via the state covariance,

$$\hat{\mathbf{u}}_t^+ = \hat{\mathbf{u}}_t^- + \mathbf{K}_t\big(\mathbf{y}_t - \mathbf{H}_t\mathbf{E}\hat{\mathbf{u}}_t^-\big), \qquad \mathbf{\Sigma}_t^+ = (\mathbf{I} - \mathbf{K}_t\mathbf{H}_t\mathbf{E})\,\mathbf{\Sigma}_t^-\,(\mathbf{I} - \mathbf{K}_t\mathbf{H}_t\mathbf{E})^\top + \mathbf{K}_t\mathbf{R}\mathbf{K}_t^\top. \tag{4.4}$$

The gain balances the prediction and the incoming measurement. Larger $\mathbf{Q}$ increases $\mathbf{\Sigma}_t^-$ and raises $\mathbf{K}_t$ in directions seen by $\mathbf{H}_t\mathbf{E}$. Larger $\mathbf{R}$ enlarges the matrix $(\mathbf{H}_t\mathbf{E}\,\mathbf{\Sigma}_t^-\,\mathbf{E}^\top\mathbf{H}_t^\top + \mathbf{R})$ inside the inverse, which reduces $\mathbf{K}_t$, so the filter relies more on the prediction. In the experiments, $\mathbf{Q}$ and $\mathbf{R}$ equal the ground-truth values from data generation in order to solely focus on the strengths and limitations of the tracking algorithm.

4.3 Experiments

Synthetic experiments are conducted that evaluate the ability of the dynamic GTI Kalman-based algorithm to track time-varying adjacency matrices. First, the experimental setup is described, followed by the various formulations for the state dynamics. Then, the energy constraining process for the state dynamics, the transition matrix in particular, is detailed, followed by an observability analysis. Results are reported in the following section. We use the same Frobenius relative error as in Chapter 3.

4.3.1 Experimental Setup

Experiments begin with generating a sparse base adjacency matrix using an Erdős–Rényi model with connection probability 0.1. If the resulting graph is disconnected, the procedure is repeated until connectivity is achieved. This ensures a sparse but fully connected starting point, consistent with the setup in Chapter 3.

The adjacency then evolves over time according to the GSSM in Equation 4.1, with process noise variance $\sigma^2_{\text{proc.}}$ tuned to the aggressiveness of the dynamics. For instance, dense state transitions require larger process noise to allow sufficient flexibility. The transition matrix $\mathbf{F}$ is varied across test scenarios, chosen either as the identity or of the form $\alpha\mathbf{I} + \epsilon\mathbf{C}$, where $\alpha \in [0.9, 1]$ controls decay and $\mathbf{C}$ is a sparse or dense coupling matrix encoding latent edge co-evolution. The specific scenarios are detailed in Subsections 4.3.2-4.3.4.

Graph signals are generated through the IO model: inputs $\mathbf{x}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ define the observation matrix $\mathbf{H}_t$, and outputs are given in Equation 4.1.

Together, these ingredients define a controlled synthetic environment in which different formulations of $\mathbf{F}$ govern how the reduced adjacency $\mathbf{u}_t$ evolves. This provides a set of testbeds to probe the tracking behavior of the Kalman filter under varying dynamic regimes.

4.3.2 Random Walk

First, we consider the simplest, baseline case - Random Walk. Unlike the static GTI setting where $\mathbf{Q} = \mathbf{0}$, the Random Walk includes process noise, so the adjacency drifts slowly over time rather than remaining fixed. In this scenario, the transition matrix is the identity matrix,

$$\mathbf{F} = \mathbf{I}. \tag{4.5}$$

There is therefore no latent co-evolution or coupling between entries in $\mathbf{u}_t$. Each element simply carries over its previous value and any change in the hidden state originates entirely from the process noise. This represents slow, unstructured drift. This formulation is notably similar to the static GTI case which may cause confusion. The distinction between Random Walk and Static GTI scenarios lies in that the former includes process noise contributing to topology drift and the latter not. This is also the experimental setting considered by Dabush et al. [18].

4.3.3 Block Coupling

The second formulation for the state transition matrix considered is Block Coupling. This model introduces structured interactions between two edges in the network. The transition matrix takes the form,

$$\mathbf{F} = \mathbf{I} + \epsilon\mathbf{C}, \tag{4.6}$$

where $\epsilon = 0.2$ and $\mathbf{C}$ is a (sparse) matrix with sparse off-diagonal entries, $\{0, 1\}$ indicating a latent coupling between two edges in the network. Below, we consider three variations based on the sparsity of entries in $\mathbf{C}$:

- **Small Block Coupling:** The off-diagonal entries of $\mathbf{C}$ are sparse, leading to weak interactions between blocks. Sparsity levels in $\mathbf{C}$ are designed to be approximately 97.5%.
- **Medium Block Coupling:** There is more coupling between individual edges in the hidden state, with more non-zero off-diagonal entries in $\mathbf{C}$. Sparsity levels in $\mathbf{C}$ are designed to be approximately 95%.
- **Large Block Coupling:** The matrix $\mathbf{C}$ has even more non-zero off-diagonal entries, with a sparsity level of approximately 92.5%, creating stronger co-evolutionary effects between entries in $\mathbf{u}_t$.

4.3.4 Decay Coupling

In the last test scenario for the state transition model, two effects are combined: Decay and Coupling. The transition matrix is,

$$\mathbf{F} = \alpha\mathbf{I} + \epsilon\mathbf{C}, \tag{4.7}$$

where $\alpha = 0.9$, $\epsilon = 0.2$, and $\mathbf{C}$ is a dense Gaussian matrix where entries are sampled from $\mathcal{N}(0, 0.1^2\mathbf{I})$. The decay term α reduces the magnitude of each element over time, while $\mathbf{C}$ enables global interaction between all edge weights. This leads to a more aggressive and entangled evolution than the block-based variant, as all edges co-evolve.

4.3.5 Generating a Synthetic F with Constrained Energy

Without proper design of the transition matrix, the system may become unstable, with hidden states diverging over time. To avoid this, normalization is applied to $\mathbf{F}$.

First, the per-row energy is constrained,

$$E_i \;=\; \|\mathbf{F}_{i,:}\|_2^2 + \sigma^2_{\text{proc.}} \;\leq\; 1, \tag{4.8}$$

where $\mathbf{F}_{i,:}$ is the ith row of $\mathbf{F}$ and $\sigma^2_{\text{proc.}}$ is the process noise variance. Rows that violate the bound are rescaled so that,

$$\|\mathbf{F}_{i,:}\|_2^2 \;=\; 1 - \sigma^2_{\text{proc}}. \tag{4.9}$$

This step improves conditioning and ensures the system does not diverge.

After this normalization process, the $\mathbf{F}$ matrices under the various scenarios may look like,

$$\mathbf{F}_{\text{RW}} = \begin{pmatrix} 0.99 & 0 & 0 & \dots & 0 \\ 0 & 0.99 & 0 & \dots & 0 \\ 0 & 0 & 0.99 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0.99 \end{pmatrix} \quad \mathbf{F}_{\text{BC, Small}} = \begin{pmatrix} 0.95 & 0 & 0 & \dots & 0.29 & 0 \\ 0 & 0.92 & 0 & \dots & 0.27 & 0.27 \\ 0 & 0 & 0.99 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 0.99 \end{pmatrix} \quad \mathbf{F}_{\text{DC}} = \begin{pmatrix} 0.60 & 0.01 & 0.01 & \dots & 0.03 & 0.02 \\ 0.01 & 0.60 & 0.1 & \dots & 0.03 & 0.01 \\ 0.01 & 0.03 & 0.60 & \dots & 0.01 & 0.02 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0.01 & 0.01 & 0.01 & \dots & 0.02 & 0.60 \end{pmatrix}$$

Figure 4.1: Examples of state transition matrices used in experiments: random walk (RW), block coupling (BC), and decay coupling (DC).

These matrices were generated for a network with 10 nodes and represent what the state dynamics look like after normalization, used to constrain total system energy and prevent divergence.

4.4 Results and Discussion

This section presents the performance of the reduced Kalman filter for dynamic graph topology tracking on synthetic data. All experiments run for $1,000$ time steps. The dynamic scenarios include Random Walk, Block Coupling with small, medium, and large co-evolution, and Decay Coupling.

For each scenario, the *evolution of the adjacency matrix is visualized with stacked plots*. The top row shows the ground truth and the bottom row shows the estimate.

Visual inspection provides an initial view of tracking quality over time. Next, the *numerical values at the final time step* are examined. A close-up of the true and estimated adjacency matrices enables direct value-wise comparison and highlights the effect of error accumulation. Finally, the *adjacency is vectorized* at the last time step, and both the ground truth and the estimate are plotted. This *per-edge view* tests accuracy at the level of individual connections and complements the earlier matrix plots. Taken together, these three views assess global and local performance under the different dynamic regimes and at the point where accumulated error is most visible.

4.4.1 Observability Diagnostics

Before analyzing tracking accuracy, we first test whether the dynamic graph SSM is observable under each state dynamics scenario.

For a finite horizon L, the observability matrix is defined in Subsection 4.2.1. Full column rank of $\mathcal{O}_{t,L}$ implies that the reduced state $\mathbf{u}_t$ can be uniquely determined from L consecutive outputs [22, 26]. The parameter L is therefore interpreted as the number of steps over which the system must be observed for all state directions to be "seen" or "observable." In our case, this means that over any block of L time steps, the outputs contain enough independent information to reconstruct all $d_{\text{red}} = 45$ parameters of the reduced adjacency state. In theory, choosing $L = d_{\text{red}} = \frac{N(N-1)}{2}$ guarantees observability, although in practice much smaller horizons already reveal the same behavior.

Determining a suitable buffer size is a function of the number of nodes. With N=10 nodes, the reduced dimension is $d_{\text{red}} = 45$. Each time step contributes at most N rows to $\mathcal{O}_{t,L}$, so a natural lower bound is $L \geq \left\lceil \frac{d_{\text{red}}}{N} \right\rceil = \left\lceil \frac{45}{10} \right\rceil = 5$.

Table 4.1: Observability results at $L = 5, 10$ for $N = 10$. All scenarios achieve full rank at $L = 10$, though conditioning varies by transition model.

	$L = 5$		$L = 10$	
Scenario	Rank	Cond. Num.	Rank	Cond. Num.
Random Walk	40/45	2.07×10^{16}	45/45	6.10
Block Coupling (Small)	45/45	4.94×10^{2}	45/45	6.79
Decay Coupling	45/45	2.09×10^{2}	45/45	4.21×10^{1}

These tests in Table 4.1 show that the minimal horizon $L = 5$ is insufficient as Random Walk is not observable as the observability matrix is not full rank. Therefore, doubling to $L = 10$, we observe that rank is always full across scenarios. As lower condition numbers of the observability matrix indicate better conditioning, there is also a notable improvement for all scenarios when doubling to $L = 10$. As expected, Random Walk exhibits the best conditioning under this scenario, while Block and Decay coupling degrade conditioning despite formal observability. These differences foreshadow the tracking behavior discussed in Subsection 4.3.2-4.3.4. $L = 10$ throughout the remainder of the experiments. L is used here for a local observability/conditioning and not used by the Kalman filter itself.

4.4.2 Random Walk

Figure 4.2 indicates the estimation is faithful to the ground truth and improves gradually over time evinced by a readily decreasing relative error. Tracking in this scenario is easier than in the other scenarios as the evolution of the system is slow time-varying and purely noise-driven. The transition matrix $\mathbf{F} = \mathbf{I}$ lacks any structured coupling between link values, and the process noise has a small variance ($\sigma^2_{\text{proc}} = 0.01$). Similar behavior has been reported in the literature by [14] under extended Kalman filtering with different data models but a similar $\mathbf{F} = \mathbf{I}$ scenario. In this work, the Random Walk transition matrix is rescaled as $\mathbf{F} = 0.99\mathbf{I}$ rather than exactly $\mathbf{I}$, so that the spectral radius remains just below one and long-term drift is bounded.

Figure 4.2 further shows the actual values of the adjacency matrix at $t = 0$ and $t = 1000$. At the final time step, the estimates in the yellow-highlighted region match the ground truth values very closely. The relative error drops to 0.16, indicating excellent quantitative tracking. Some entries take on small negative values, which stem from the synthetic data generation process as their edge weights are drawn from a zero-mean distribution and evolve linearly with noise, rather than being explicitly constrained to remain positive. We note also that the normalization process successfully constrains the system across all time steps.

Finally, the vector, per entry, comparison offers a detailed look at the per-edge estimation performance at the last time step. The estimated and true adjacency vectors are nearly indistinguishable, with minimal deviation across all dimensions. This confirms that the Kalman filter achieves close and accurate tracking of the full topology vector under the Random Walk scenario.

4.4.3 Block Coupling

The Block Coupling experiments vary the strength of latent co–evolution in $\mathbf{F}$ from small to large. Figure 4.3 shows stacked evolutions of the true and estimated adjacencies, and Figure 4.4 reports per–edge comparisons at the final time.

In the *small coupling (Figures 4.3a, 4.4a)* scenario, the filter tracks the co–evolution structure well and recovers sparsity patterns, with minor over/underestimation of amplitudes. The *medium coupling (Figures 4.3b, 4.4b)* scenario begins to struggle with a mild blurring effect appearing. Large values are damped and small values are inflated, so sharp contrasts are partially lost. Finally, in *large coupling (Figures 4.3c, 4.4c)*, the damping is pronounced. The estimates gravitate toward the average adjacency cell value, and extreme edges are poorly reconstructed.

4.4.3.1 Why does the increase in latent co-evolutions decrease tracking ability?

When more off–diagonal terms are present in $\mathbf{F}$, each hidden state entry becomes a weighted combination of more contributors. For example, node i may combine inputs from nodes j, k, and itself at the previous step. This averaging pushes extremes toward mean values. There might still be contrast but the contrast will be within a smaller range. This can be seen in Figure 4.3 where the maximum adjacency value more than halves as coupling increases from a) to c).

From an estimation perspective, stronger coupling worsens the conditioning of the observability matrix (see Table 4.2). In more poorly conditioned systems, different directions of the state affect the outputs in nearly the same way. This means the measurements provide less independent information to distinguish among them.

In the prediction step,

$$\boldsymbol{\Sigma}_t^- = \mathbf{F}\,\boldsymbol{\Sigma}_{t-1}^+\,\mathbf{F}^\top + \mathbf{Q}, \tag{4.10}$$

stronger coupling propagates more cross–dependence between entries of the state covariance, inflating the matrix. This increases the Kalman gain in response, so the filter makes larger corrections in general. This is seen in Table 4.2 with a larger correction for increased coupling despite the smaller state values as previously mentioned. When conditioning is poor, these larger corrections are less well informed by the data meaning large incorrect corrections are occuring. Instead of restoring sharp contrasts, the updates tend to spread noise across the state, leading to blurred and damped adjacency estimates.

Increasing coupling worsens conditioning, inflates corrections, and reduces the ability of the Kalman filter to recover sharp high and low adjacency values. This explains the rise in steady-state error with stronger latent co-evolutions.

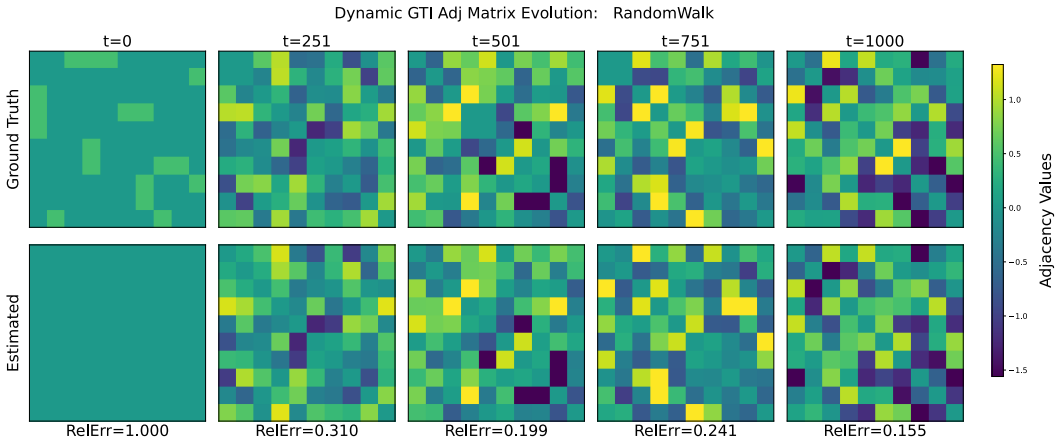
Table 4.2: Effects of increasing latent co-evolution in $\mathbf{F}$ (averaged over the last 50% of time steps, N=10, T=1000).

Coupling	Rel. Err.	cond($\mathcal{O}$)	$\|\hat{\mathbf{u}}_t^+ - \hat{\mathbf{u}}_t^-\|_2$
Low	0.48	3.6×10^1	1.27
Medium	0.57	3.3×10^2	2.61
High	0.82	8.2×10^2	3.04

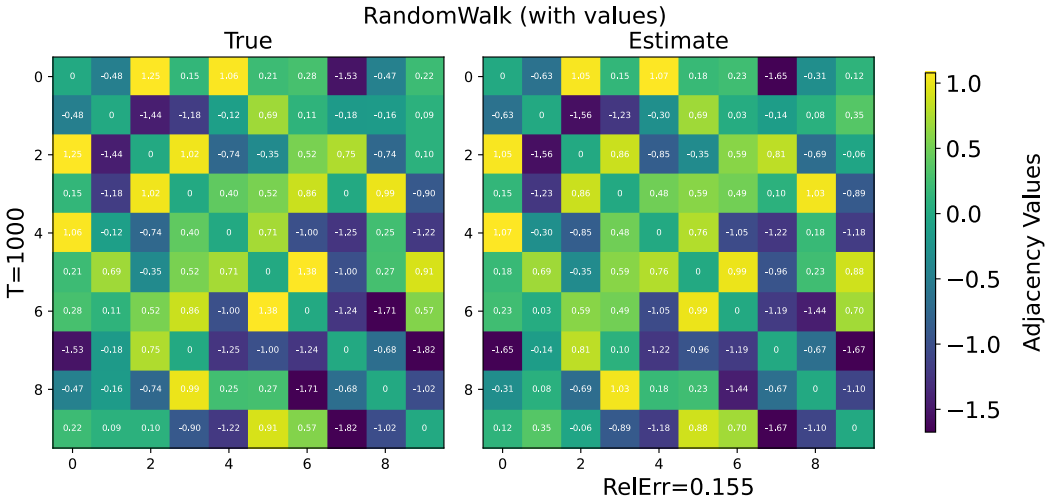
4.4.4 Decay Coupling

Under Decay Coupling, the structure is partially recovered by the algorithm, with some similarities in block patterns. However, the latent co-evolution effects are strong presenting a strong dampening effect of all signal values. Larger and smaller adjacency matrix values struggle to be identified. This is evident in Figure 4.6.

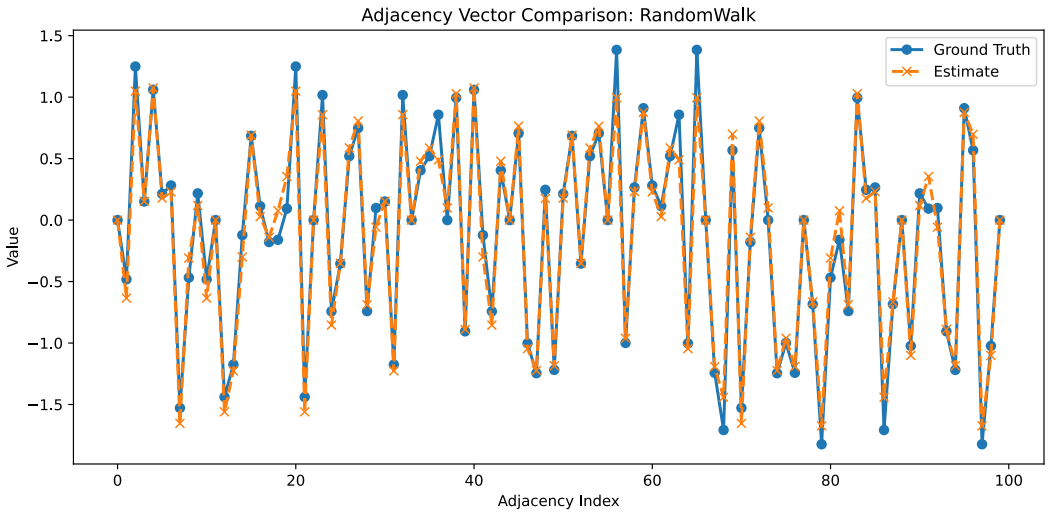
The combined decay and dense coupling in $\mathbf{F}$ acts as a strong smoothing prior as the decay term reduces the magnitude of all edges over time, while the dense coupling spreads values across the state vector, moving all edges toward a common mean. This dynamic flattens the predicted state even before the measurement update. Because $\mathbf{Q}$ is small, the Kalman gain is low, so the filter only partially adjusts toward the measurements, preserving much of this flattened prediction. As a result, sharp high or low edges in the true adjacency are systematically underestimated or overestimated, producing reconstructions dominated by medium-valued entries.



(a) Stacked adjacency matrices over time.

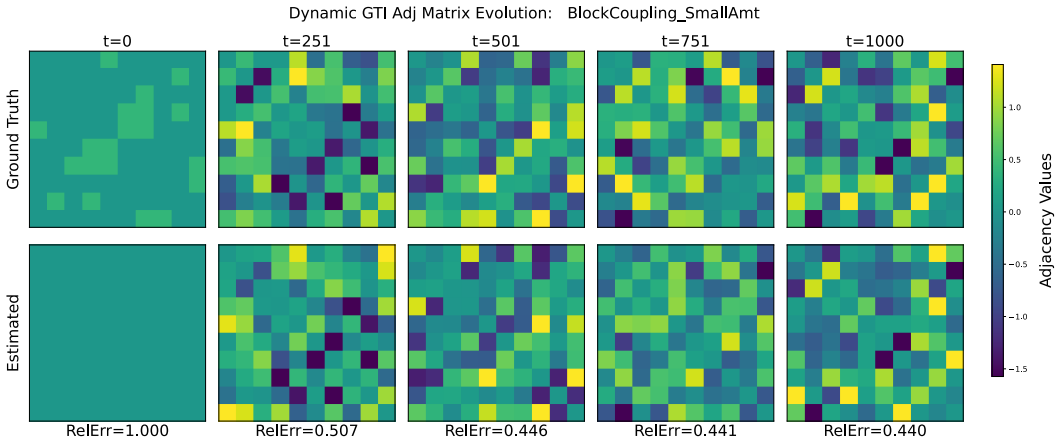


(b) Close-up of last ground truth and estimated adjacency values.

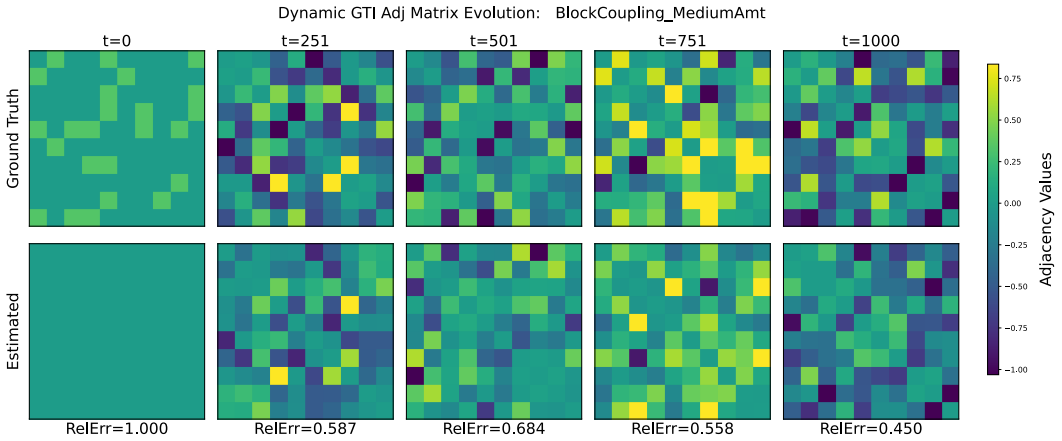


(c) Comparison of true and estimated adjacency vectors at the final time step.

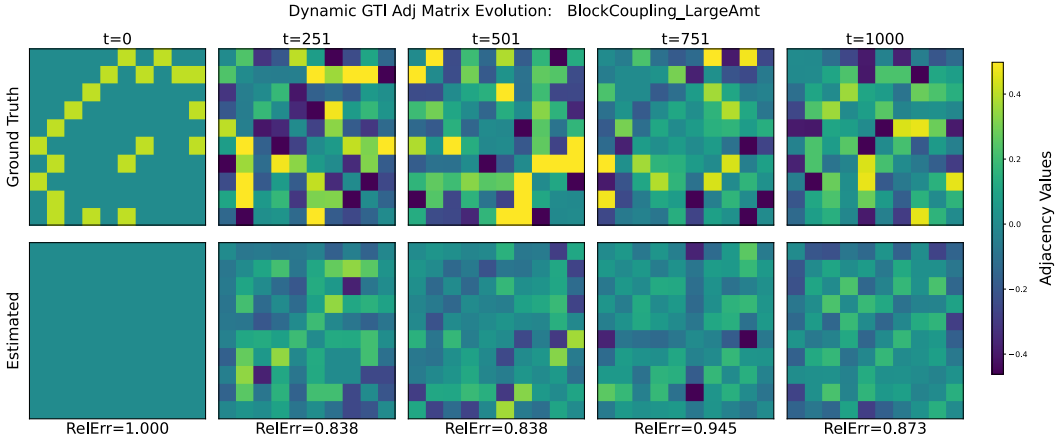
Figure 4.2: Random Walk scenario. (a) Evolution of adjacency matrices over time, (b) close-up at last time step comparing ground truth and estimated, and (c) adjacency vector comparison at the final step.



(a) Small latent co-evolution.

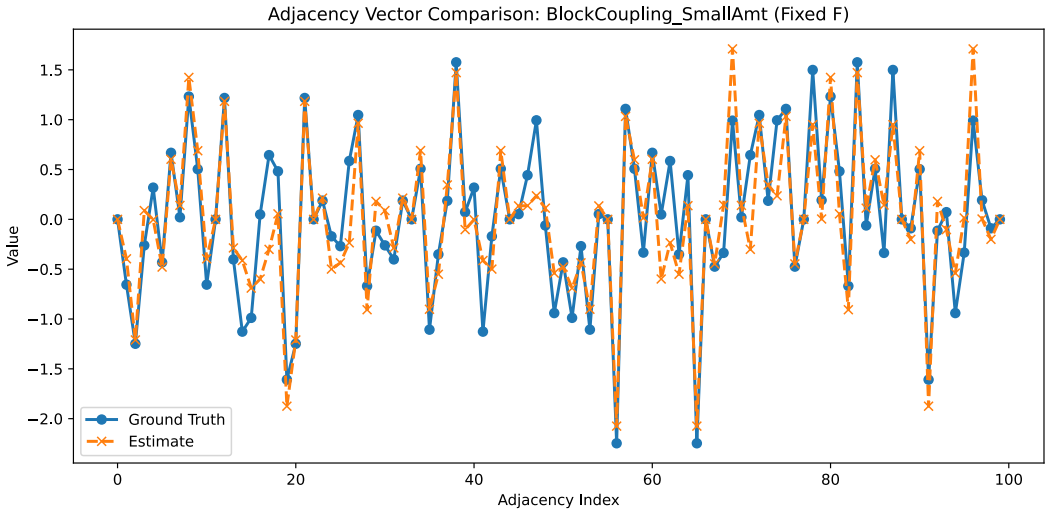


(b) Medium latent co-evolution.

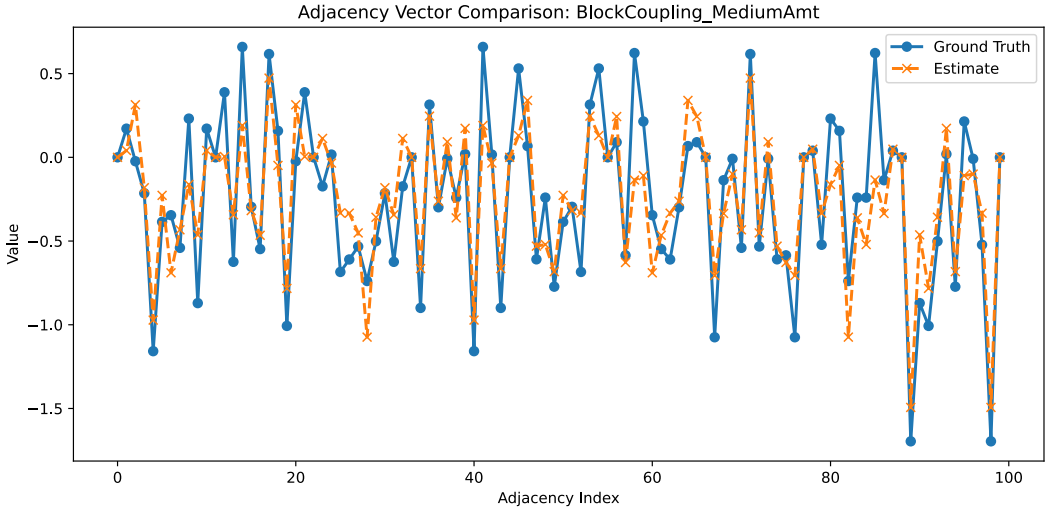


(c) Large latent co-evolution.

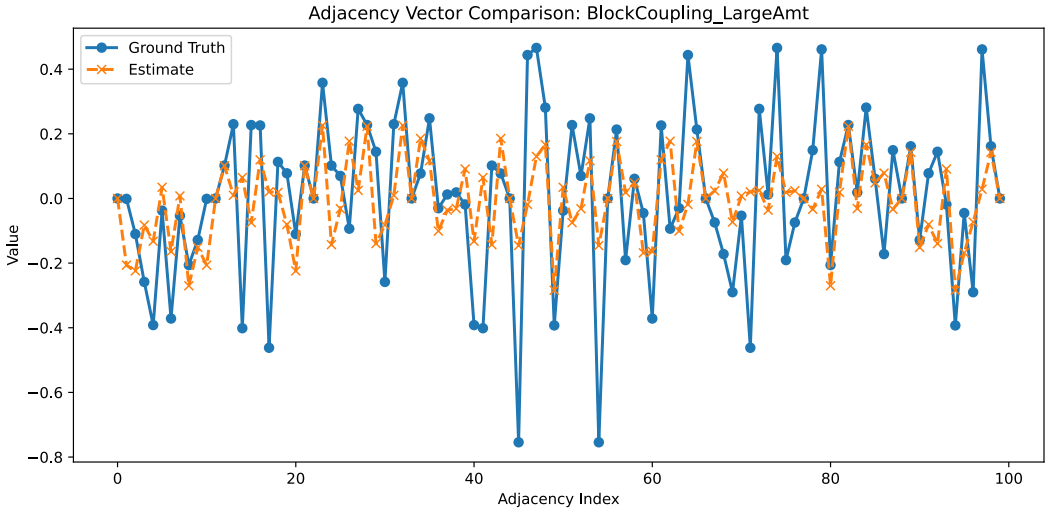
Figure 4.3: Block Coupling scenarios with increasing latent co-evolution. Tracking performance decreases as latent co-evolution increases.



(a) Small latent co-evolution. Accurate reconstruction with minor overestimation effects.



(b) Medium latent co-evolution. Increased blurriness with both over- and under-estimations.



(c) Large latent co-evolution. Most edges underestimated or mismatched, with values pushed toward the average.

Figure 4.4: Adjacency vector comparison for Block Coupling scenarios with increasing latent co-evolution: (a) small, (b) medium, (c) large. A strong dampening effect incurs as co-evolution increases.

4.4.5 Performance Comparison

Figure 4.5 presents the evolution of the relative error over time for the three synthetic scenarios under known transition matrix $\mathbf{F}$. In all cases, the Kalman filter is able to dynamically track the underlying graph structure, but with varying degrees of speed and accuracy. The Random Walk and Block Coupling Small Amount converge to a low relative error. The other three schemes with increasingly more latent co-evolution of the hidden state have lesser tracking capabilities reflected in a higher relative error even after multiple time steps.

4.5 Conclusion

This chapter extends the static GTI method to an online, dynamic setting using a Kalman filter in a reduced state that enforces symmetry and a zero diagonal by construction. We evaluated three transition models—Random Walk (RW), Block Coupling (BC, small/medium/large), and Decay Coupling (DC)—with row–energy and spectral–radius normalization of $\mathbf{F}$ to avoid divergence.

- **Random Walk.** With diagonal $\mathbf{F}$ and small process noise, the filter tracks structure and magnitudes well.

- **Block Coupling.** Increasing off–diagonal coupling reduces contrast in the predictions and degrades reconstruction of high/low edges. Diagnostics show the observability matrix remains full rank but becomes more ill–conditioned as coupling grows (condition number increases). Larger updates without added innovation information explain the rise in steady–state error and the observed damping/blurring.

- **Decay Coupling.** The decay term contracts all entries while dense coupling spreads values, producing strong smoothing before the update. With modest $\mathbf{Q}$, the gain is limited and sharp differences are not fully restored, leading to under/over–estimation and medium–valued reconstructions.

- **Stability and identifiability.** For all scenarios, $(\mathbf{F}, \mathbf{H}_t)$ is observable (full rank over the chosen horizon). Poorer conditioning with stronger coupling reduces practical recoverability, matching the error trends above.

Overall, dynamic tracking is effective when evolution is weakly coupled. Performance deteriorates as latent co–evolutions increase because information along weakly excited directions is limited while prediction smoothing grows. Two direct avenues follow: (i) adjust $\mathbf{Q}$ (or its structure) to raise responsiveness in coupled directions, and (ii) encode sparsity/contrast during prediction or update (e.g., structured priors or masks). Joint estimation of the state transition is addressed in the next chapter.

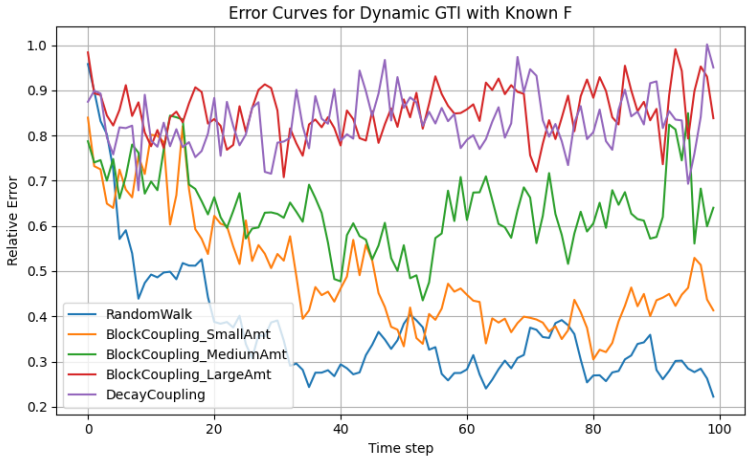
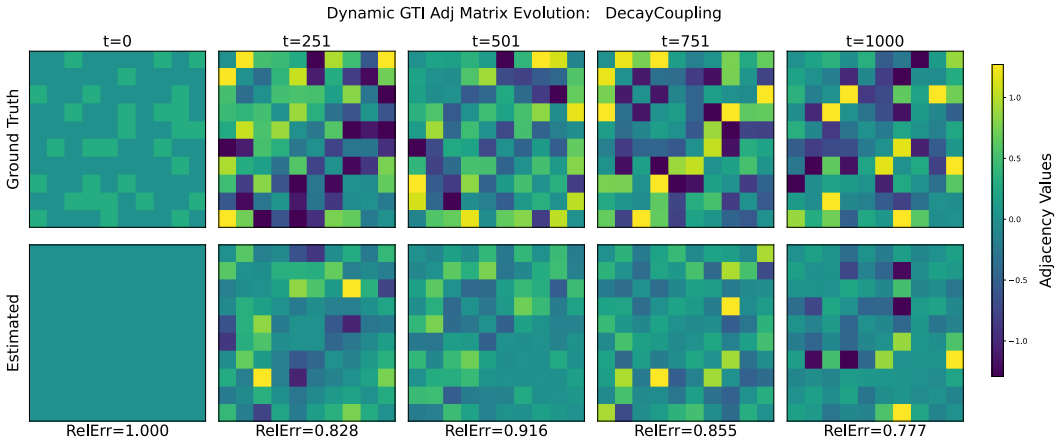


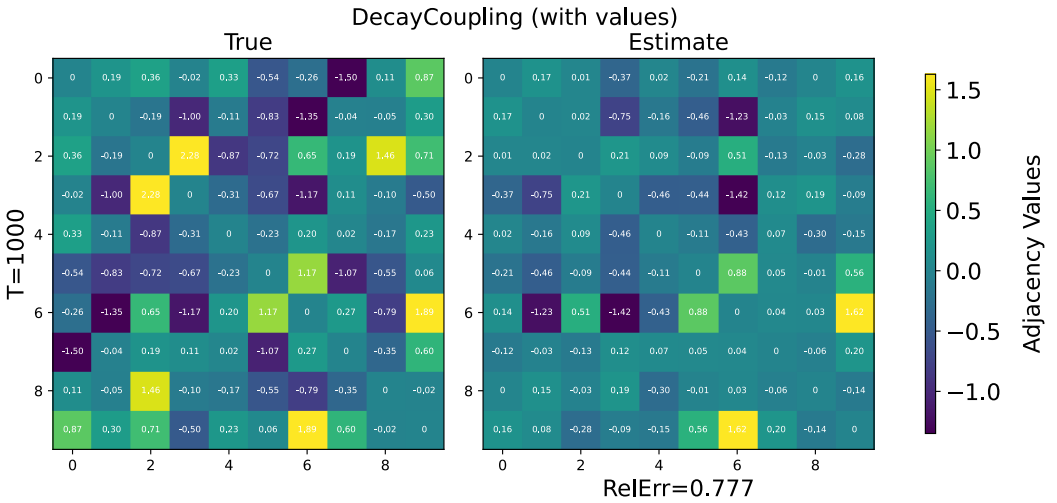
Figure 4.5: Random Walk and Block Coupling Small Amount converge to a small relative error. The Block Coupling and Decay Coupling scenarios struggle due to more co-evolution.

Summary of Key Conclusions

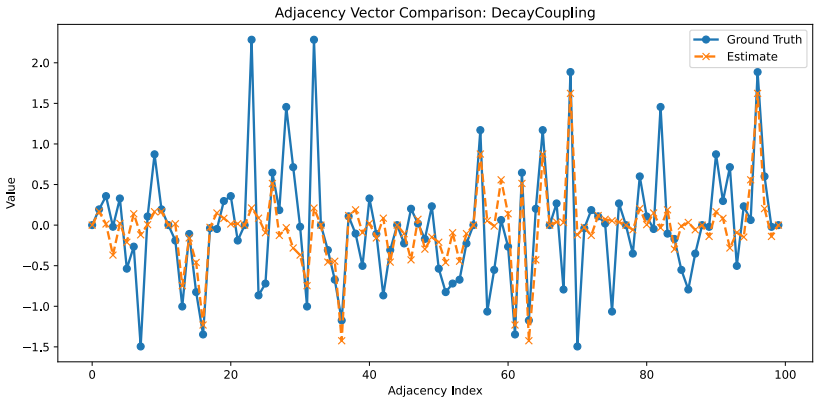
- Reliable adjacency tracking is possible by modeling the graph in a reduced GSSM and applying a Kalman filter under known state dynamics.
- Performance is strongest when edges evolve independently or only weakly co-evolve preserving sharp contrasts with low Relative Error.
- Increasing the co-evolution raises $\text{cond}(\mathcal{O}_{t,L})$ despite the matrix being full rank which has the practical effect of causing damping/blurring in the graph adjacency estimate.



(a) Stacked snapshots of true and estimated adjacency matrices. The algorithm captures average values but struggles with sharp high and low entries.



(b) Close-up of true versus estimated entries. Magnitudes are roughly reconstructed, though with both over- and under-estimations.



(c) Adjacency vector comparison. Underestimation is consistent across indices, but overall structure is partially learned.

Figure 4.6: Decay Coupling scenario. (a) Evolution of adjacency matrices over time, (b) close-up comparison at $T = 100$, and (c) adjacency vector comparison at the final step.

Joint Estimation of the State Dynamics and Dynamic Adjacency

5

Earlier chapters considered settings where the state transition matrix $\mathbf{F}$ was assumed known and fixed, focusing solely on tracking the dynamic adjacency matrix. This enabled evaluation of whether the Kalman filter can track a dynamic adjacency when the state dynamics are specified. In most real applications, however, the state dynamics are unknown. Estimating it is required for practical use but introduces a circular dependency: the Kalman filter needs the state dynamics for prediction and update steps, while learning the state dynamics requires predicted states from the filter. This dependency risks continuous error propagation between state and transition estimates. Regularization through structural priors is therefore essential to shrink the solution space and stabilize estimation.

This chapter develops an online method to jointly estimate the hidden state $\hat{\mathbf{u}}_t$ and $\hat{\mathbf{F}}_t$. The primary objective is to test feasibility and accuracy of joint estimation as the circular dependency between estimated states and estimated dynamics has the potential to be detrimental to tracking capabilities. Performance is compared under two main sparsity regimes: (i) when a sparsity mask for the transition matrix specifying the state dynamics is known, and (ii) when no sparsity mask is provided. A third regime, where a sparsity mask is learned from estimated states, is described separately in Section 5.3. Throughout, the ground truth state dynamics $\mathbf{F}$ are fixed but unknown, while the estimate is updated online.

5.1 Problem Formulation

Building on the reduced graph state-space model (GSSM) introduced in Chapter 4, the adjacency $\mathbf{A}_t$ is represented by the reduced hidden state $\mathbf{u}_t \in \mathbb{R}^{d_{\text{red}}}$ with $d_{\text{red}} = \frac{N(N-1)}{2}$. The expansion matrix $\mathbf{E}$ maps $\mathbf{u}_t$ back to the full adjacency vector $\mathbf{a}_t = \mathbf{E}\mathbf{u}_t$, enforcing symmetry and zero diagonal. The reduced GSSM is,

$$\begin{aligned} \mathbf{u}_t &= \mathbf{F}\,\mathbf{u}_{t-1} + \boldsymbol{\eta}_{t-1}, & \boldsymbol{\eta}_{t-1} &\sim \mathcal{N}(\mathbf{0}, \mathbf{Q}), \\ \mathbf{y}_t &= \mathbf{H}_t\mathbf{E}\,\mathbf{u}_t + \boldsymbol{\nu}_t, & \boldsymbol{\nu}_t &\sim \mathcal{N}(\mathbf{0}, \mathbf{R}), \end{aligned} \tag{5.1}$$

where $\mathbf{F} \in \mathbb{R}^{d_{\text{red}} \times d_{\text{red}}}$ is the *unknown* state transition matrix to be estimated jointly with the reduced state $\mathbf{u}_t$.

Unlike in Chapter 4, where $\mathbf{F}$ was assumed fixed and known, here the filter must infer $\mathbf{F}$ online. This introduces a circular dependency: predicting $\mathbf{u}_t$ requires a reliable $\hat{\mathbf{F}}$, while estimating $\mathbf{F}$ requires an accurate $\hat{\mathbf{u}}_t$. Inaccuracies in either propagate directly into the other, making the scheme vulnerable to drift and instability.

Suppose a buffer of T consecutive reduced states is available, $\mathbf{U}_{\text{next}} \approx \mathbf{F}\,\mathbf{U}_{\text{prev}}$ where,

$$\mathbf{U}_{\text{prev}} = \begin{bmatrix} \mathbf{u}_t & \mathbf{u}_{t+1} & \cdots & \mathbf{u}_{t+T-1} \end{bmatrix}, \quad \mathbf{U}_{\text{next}} = \begin{bmatrix} \mathbf{u}_{t+1} & \mathbf{u}_{t+2} & \cdots & \mathbf{u}_{t+T} \end{bmatrix}. \tag{5.2}$$

Here, each $\mathbf{u}_\tau \in \mathbb{R}^{d_{\text{red}}}$ is a reduced state vector, so $\mathbf{U}_{\text{prev}}, \mathbf{U}_{\text{next}} \in \mathbb{R}^{d_{\text{red}} \times T}$. Recovering $\mathbf{F}$ uniquely requires $\text{rank}(\mathbf{U}_{\text{prev}}) = d_{\text{red}}$, which implies $T \geq d_{\text{red}}$. Since $\mathbf{F}$ has d_{red}^2 entries, in practice $T \gg d_{\text{red}}$ is needed. If dynamics are weakly excited (e.g., Random Walk with small process noise), consecutive states are nearly dependent, and $\mathbf{U}_{\text{prev}}$ becomes ill-conditioned. In such cases, many different $\mathbf{F}$ fit the data equally well, and the estimation problem becomes ill-posed.

In practice, only a short window of noisy estimates $\hat{\mathbf{u}}_t$ is available. Identifiability is then further restricted because (i) samples can be correlated and (ii) measurement noise corrupts the states.

To make the joint problem feasible, structural regularization is essential. Sparsity masks, adaptive penalties, or learned priors shrink the solution space so that the effective sample requirement scales with the number of free parameters rather than d_{red}^2. These priors not only stabilize estimation but also improve interpretability by encouraging realistic dynamics (e.g., sparse or block-structured $\mathbf{F}$).

5.2 Joint Estimation Algorithm

The joint estimation task combines Kalman filtering for state tracking with structure-aware learning of $\hat{\mathbf{F}}_t$. The Kalman filter updates $\hat{\mathbf{u}}_t$ online, and $\hat{\mathbf{F}}_t$ is refined from a short, rolling window via a least-squares objective with structural priors and an Adam-based adaptive learning rate. A retroactive smoothing pass re-estimates the recent states under the latest $\hat{\mathbf{F}}_t$ to limit bias from early, inaccurate transitions. Because the problem is underdetermined in general, it is advantageous to reduce the solution space enabling the best estimates from the least amount of data. A sparsity mask accomplishes this by only allowing a small set of parameters in $\hat{\mathbf{F}}_t$ to be nonzero. To compare performance, we consider a known sparsity mask and unknown sparsity mask regime. Because having a known sparsity mask is not reliably available in real-world settings, we also propose and explore a proposed method to learn the sparsity mask from the estimated states where the sparsity mask was unknown.

5.2.1 Kalman Filter to Perform Online Adjacency Tracking

To estimate the state, the same Kalman prediction and update steps are executed as in Algorithm 2. The Kalman filter algorithm has produced accurate tracking of the adjacency matrix under various known state dynamic schemes as demonstrated in Chapter 4. Thus, this approach is continued for adjacency estimation in the joint algorithm.

5.2.2 Least-Squares Based Windowed Optimization Scheme for Online $\hat{\mathbf{F}}_t$ Learning

The transition matrix $\mathbf{F}$ is estimated through a windowed optimization scheme that matches the sequential nature of the graph SSM, where data arrive step by step. Batch methods such as least-squares or subspace identification require access to a full or very large set of time-stamped samples to produce a single estimate and are less suited for

fast-varying systems. Expectation–Maximization (EM) could in principle estimate both states and parameters jointly, but it requires repeated passes over the entire dataset and is computationally expensive for large graphs with large dimensionalities [28]. In contrast, online optimization updates $\hat{\mathbf{F}}_t$ incrementally using only a short buffer of past states. This reduces memory usage, allows the algorithm to adapt quickly as dynamics change, and keeps per-step complexity low.

The optimization is built on a small, windowed least-squares objective. The estimate of $\mathbf{F}$ is obtained by minimizing a least-squares loss with structural priors

$$\begin{aligned}\mathcal{L}_t(\mathbf{F}) = &\underbrace{\sum_{\tau=t-k_{\text{alt}}+1}^{t} \left\|\mathbf{y}_\tau - \mathbf{H}_\tau \mathbf{E}\mathbf{F}\hat{\mathbf{u}}^+_{\tau-1}\right\|_2^2}_{\text{data fidelity}} + \underbrace{\lambda \sum_i (F_{ii}-1)^2}_{\text{identity prior}} \\ &+ \underbrace{\mu\|\mathbf{F}\|_1}_{\text{sparse } \mathbf{F}} + \underbrace{\gamma \sum_{i\neq j}(1-M_{ij})F_{ij}^2}_{\text{mask penalty}}.\end{aligned} \tag{5.3}$$

where hyperparameters are updated during optimization, being learned periodically via Adam as described in Subsection 5.2.2.4.

During startup, when the buffer is not yet full ($L < k_{\text{alt}}$), the transition matrix is held fixed at its initialization $\mathbf{F} = \mathbf{I}$. In this phase, the algorithm only updates the hidden states $\hat{\mathbf{u}}_t$ via the Kalman filter while storing observations in the buffer. No updates of $\mathbf{F}$ are performed. Once the buffer length reaches $k_{\text{alt}} = 200$, the standard windowed update of $\mathbf{F}$ with retroactive smoothing is activated.

Choosing an appropriate buffer size is crucial. Too short and poor estimates will result. Too long and the online algorithm is suddenly not-so-online, taking copious amounts of computation time. Therefore, a heuristic is developed for buffer size.

Algorithm 3 Heuristic buffer size selection for online $\mathbf{F}$ learning

Require: Number of nodes N, sparsity percentage s (default $s = 90\%$)

1: Compute reduced dimension $d_{\text{red}} = \frac{N(N-1)}{2}$
2: Estimate number of free parameters $p = \left(1 - \frac{s}{100}\right) d_{\text{red}}^2 - d_{\text{red}}$
3: Set buffer size $L = \lceil p \rceil$ (round to nearest 10 if desired)
4: **return** L

Applying the heuristic for a network with $N = 10$ nodes yields $d_{\text{red}} = 45$. Assuming $s = 90\%$ sparsity, the effective number of free parameters is $p = (1-0.9)\cdot 45^2 - 45 \approx 158$. The rule therefore suggests a buffer size of roughly $L \approx 160$ (rounded to the nearest ten), which aligns with empirical practice. When the buffer is not yet full, the optimization holds $\mathbf{F}$ close to the identity, gradually moving toward the estimated matrix as the buffer fills.

5.2.2.1 Identity Prior on F

The diagonal entries of $\mathbf{F}$ are encouraged to remain close to one, reflecting a self-history effect where each edge evolves primarily from its own past value. This prior captures the

strong temporal persistence observed in many real-world networks, where link weights evolve gradually rather than abruptly. Similar persistence assumptions are used in dynamic graph learning and Kalman-based tracking of network states, where the state dynamics are close to the identity ($\mathbf{F} = \mathbf{I}$) [18]. From a state-space perspective, this corresponds to a random walk prior, which is a standard choice when little is known about the underlying dynamics.

A further justification comes from continuous-time models. Suppose edge weights evolve according to dynamics of the form $\dot{u}(t) = -\alpha u(t) + \cdots$, where $\dot{u}(t)$ denotes the time derivative of $u(t)$ and α is a decay rate. Intuitively, this describes a gradual decay of $u(t)$ toward zero at rate α. When this system is discretized with a sampling interval Δt, the update becomes $\mathbf{u}_t \approx (1 - \alpha\Delta t)\,\mathbf{u}_{t-1} + \cdots$. For small values of Δt, the diagonal entries of the transition matrix $\mathbf{F}$ naturally remain close to one. The identity prior therefore reflects the assumption that link values change gradually over time.

The prior is implemented as,

$$\lambda \sum_i (F_{ii} - 1)^2, \tag{5.4}$$

which penalizes deviations from unity. It helps prevent collapse of edge magnitudes toward a global mean and stabilizes online estimation. In consensus-type networks (e.g., heat diffusion or averaging dynamics), diagonals near one may be less appropriate than broader coupling, since such processes inherently diffuse values toward collective behavior. In those cases, diffusion or smoothness-based priors are more appropriate than self-evolution priors.

5.2.2.2 Sparsity Prior on F

It is unlikely that the evolution of an edge depends on *every* other edge in the network. More plausibly, an edge evolves based on its own past value and a limited number of other edges. Consider a network of airports where edges represent passenger flows. The flow on the Amsterdam–Frankfurt route will be influenced mainly by its own history and by a few neighboring flows like Amsterdam–Munich or Frankfurt–Zurich. It would be unrealistic to assume that the link of interest, Amsterdam–Frankfurt is directly driven by an entirely different link in the network, like New York–Tokyo. This motivates promoting sparsity via,

$$\mu \|\mathbf{F}\|_1, \tag{5.5}$$

which drives small coefficients toward zero so that only dominant interactions remain. A sparse state transition matrix is motivated by [22] where evidence supports sparsity because interactions between edges are typically local. Ljung et al. also motivate sparse state transition matrices highlighting that it is a necessary condition for identifiability in high-dimensional state-space models [26].

Without sparsity in the state dynamics, estimation risks collapsing toward nearly uniform dynamics, where large and small coefficients in $\mathbf{u}_t$ migrate toward a common average. Such homogenization reduces the heterogeneity of the state. Recall, here the state is a notion of graph connectivity. A homogenous state is mainly representative of consensus networks for which this proposed algorithm is not appropriate.

5.2.2.3 Sparsity Mask Prior on F

If a sparsity mask is given or learned, the mask is included to reduce the solution space. A binary mask $\mathbf{M}$ is used to penalize off-diagonal entries that should be zero,

$$\gamma \sum_{i \neq j} (1 - M_{ij}) F_{ij}^2. \tag{5.6}$$

This limits small, extraneous connections and prevents influence from spreading across the entire system. The mask is also explicitly enforced after the update step (see Algorithm 5).

5.2.2.4 Adam Optimization and Learning-Rate Boosting/Shrinking

Gradient-based updates of $\mathbf{F}$ can be unstable in ill-conditioned setups. Adaptive methods address this by scaling updates according to past gradients. Adam does this by keeping exponential moving averages of the gradients and their squares, which produces parameter-specific step sizes [29, 30, 28].

In our case, all variables are matrices of the same dimension as the transition matrix, so $\mathbf{F}_t, \mathbf{G}_t, \mathbf{M}_t, \mathbf{V}_t, \hat{\mathbf{M}}_t, \hat{\mathbf{V}}_t \in \mathbb{R}^{d_{\text{red}} \times d_{\text{red}}}$ with $d_{\text{red}} = \frac{N(N-1)}{2}$. The matrix $\mathbf{G}_t$ holds the partial derivatives of the loss $\mathcal{L}_t$ with respect to each entry of $\mathbf{F}$. The matrix $\mathbf{M}_t$ is the exponential moving average of $\mathbf{G}_t$ with decay β_1, and $\mathbf{V}_t$ is the exponential moving average of the elementwise squared gradient with decay β_2,

$$\mathbf{M}_t = \beta_1 \mathbf{M}_{t-1} + (1 - \beta_1)\mathbf{G}_t, \tag{5.7}$$

$$\mathbf{V}_t = \beta_2 \mathbf{V}_{t-1} + (1 - \beta_2)\big(\mathbf{G}_t \odot \mathbf{G}_t\big). \tag{5.8}$$

Their normalized forms $\hat{\mathbf{M}}_t$ and $\hat{\mathbf{V}}_t$ are found by dividing by $(1 - \beta_1^t)$ and $(1 - \beta_2^t)$, respectively. These factors decay toward one as t grows, so the correction is typically only active in the early steps. The square root and division in the update step are applied elementwise. Each entry of the transition matrix is updated independently, following,

$$(\mathbf{F}_t)_{ij} = (\mathbf{F}_{t-1})_{ij} - \eta \, \frac{(\hat{\mathbf{M}}_t)_{ij}}{\sqrt{(\hat{\mathbf{V}}_t)_{ij}} + \epsilon}. \tag{5.9}$$

This ensures that each parameter has its own effective learning rate. Entries with large and noisy gradients take smaller steps, while entries with stable gradients take larger steps. The transition matrix is updated at every time step using Adam, and is further refined every k_{alt} steps during the retroactive smoothing procedure described in subsection 5.2.3. This is seen by a secondary call to Adam in line 11 of Algorithm 5.

5.2.3 Retroactive Adjacency Estimate Smoothing Using Last $\hat{\mathbf{F}}_t$ in Window

The estimates of the hidden states and the transition matrix are updated in alternation. Early on, states are generated using transition matrices that are still inaccurate. If these biased states are used directly in the loss, updates of $\mathbf{F}$ may be misled and

Algorithm 4 Adam update for $\mathbf{F}$ (all variables in $\mathbb{R}^{d_{\text{red}} \times d_{\text{red}}}$)

Initialize $\mathbf{M}_0 = \mathbf{0}$, $\mathbf{V}_0 = \mathbf{0}$, learning rate η, β_1, β_2, ϵ
for $t = 1, 2, \ldots$ **do**
  Compute gradient $\mathbf{G}_t = \nabla_{\mathbf{F}} \mathcal{L}_t$
  for each entry (i, j) **do**
    $(\mathbf{M}_t)_{ij} \leftarrow \beta_1 (\mathbf{M}_{t-1})_{ij} + (1 - \beta_1)(\mathbf{G}_t)_{ij}$
    $(\mathbf{V}_t)_{ij} \leftarrow \beta_2 (\mathbf{V}_{t-1})_{ij} + (1 - \beta_2)\big((\mathbf{G}_t)_{ij}\big)^2$
    $(\hat{\mathbf{M}}_t)_{ij} \leftarrow (\mathbf{M}_t)_{ij} / (1 - \beta_1^t)$
    $(\hat{\mathbf{V}}_t)_{ij} \leftarrow (\mathbf{V}_t)_{ij} / (1 - \beta_2^t)$
    $(\mathbf{F}_t)_{ij} \leftarrow (\mathbf{F}_{t-1})_{ij} - \eta \dfrac{(\hat{\mathbf{M}}_t)_{ij}}{\sqrt{(\hat{\mathbf{V}}_t)_{ij}} + \epsilon}$
  end for
end for

errors can accumulate. To counter this, the algorithm periodically performs retroactive smoothing: it re-estimates past hidden states in the buffer using the most recent $\hat{\mathbf{F}}$ and replaces earlier estimates with the refined ones, reducing error propagation and stabilizing convergence. Once the past hidden states have been re-smoothed, the gradient of the windowed loss w.r.t. the state transition matrix *and the hyperparameters*, unlike the previous Adam step, are computed and Adam is applied again. Hyperparameters are adjusted here because the window of re-smoothed states provides a more stable gradient signal than a single noisy sample, making this block the good mechanism for tuning them.

5.2.4 Known vs. Unknown Sparsity Masks

A sparsity mask specifies which entries of $\mathbf{F}$ may be nonzero. Sometimes, the mask is known (e.g., locality or regulatory constraints). Spatial networks often restrict interactions to nearby nodes and their edges, and financial networks may be constrained by institutional or regulatory ties. For example, an airport analyzing passenger flows may state that their outgoing flows are strictly a function of their incoming flows and all other flows in the network are to be ignored. Knowing the sparsity provides a benchmark scenario for the joint estimation algorithm, showing how well the algorithm performs when the sparsity pattern is correct and reduces the number of free variables to be learned.

More often, the exact pattern is unknown and must be inferred. Partial knowledge (forbidden links) can be encoded as fixed zeros while the rest remain discoverable. The algorithm supports both modes and strives to reduce the number of active parameters so that a finite buffer contains enough information to identify $\mathbf{F}$. If the mask is unknown then the mask is set to all ones to effectively reduce that term in the loss function to zero [31, 9].

Algorithm 5 Online Joint Estimation of $\hat{\mathbf{F}}_t$ and $\hat{\mathbf{u}}_t$ (Known or Unknown Mask)

Inputs:
  Initial state $\hat{\mathbf{u}}_0^+$, covariance $\mathbf{\Sigma}_0^+$
  process noise $\mathbf{Q}$, measurement noise $\mathbf{R}$, expansion matrix $\mathbf{E}$
  Adam parameters $(\eta, \beta_1, \beta_2, \epsilon)$, buffer size k_{alt}
  hyperparameters λ, γ, μ
  known mask $\mathbf{M}$ if available, otherwise set mask penalty to zero
for $t = 1, 2, \ldots$ **do**
  Receive $\mathbf{y}_t$, $\mathbf{x}_t$. Set $\mathbf{H}_t = \mathbf{I}_N \otimes \mathbf{x}_t^\top$.
  Kalman prediction: ▷ Estimating Adjacency

$$\hat{\mathbf{u}}_t^- = \mathbf{F}_{t-1}\hat{\mathbf{u}}_{t-1}^+, \quad \mathbf{\Sigma}_t^- = \mathbf{F}_{t-1}\mathbf{\Sigma}_{t-1}^+\mathbf{F}_{t-1}^\top + \mathbf{Q}.$$

  Kalman update: ▷ Estimating Adjacency

$$\hat{\mathbf{u}}_t^+ = \hat{\mathbf{u}}_t^- + \mathbf{K}_t(\mathbf{y}_t - \mathbf{H}_t\mathbf{E}\hat{\mathbf{u}}_t^-), \quad \mathbf{\Sigma}_t^+ = (\mathbf{I} - \mathbf{K}_t\mathbf{H}_t\mathbf{E})\mathbf{\Sigma}_t^-.$$

  Define loss (estimate transition):

$$\begin{aligned}\mathcal{L}_t(\mathbf{F}) = &\sum_{\tau=t-k_{\text{alt}}+1}^{t} \left\|\mathbf{y}_\tau - \mathbf{H}_\tau\mathbf{E}\,\mathbf{F}\,\hat{\mathbf{u}}_{\tau-1}^+\right\|_2^2 \\ &+ \lambda\sum_{i=1}^{d_{\text{red}}}\left(F_{ii}-1\right)^2 + \mu\left\|\mathbf{F}\right\|_1 + \gamma\sum_{i\neq j}\left(1-M_{ij}\right)F_{ij}^2.\end{aligned} \tag{5.10}$$

  Update $\mathbf{F}_t \leftarrow \text{Adam}(\nabla_{\mathbf{F}}\mathcal{L}_t, \mathbf{F}_{t-1})$. ▷ Adam only on $\mathbf{F}$
  if mask $\mathbf{M}$ known **then** ▷ For State Transition Estimation
    Enforce mask: $\mathbf{F}_t \leftarrow \mathbf{F}_t \odot \mathbf{M}$.
  end if
  if $t \bmod k_{\text{alt}} = 0$ **then** ▷ Retroactively smoothing states
    Rerun KF over k_{alt} steps with fixed $\mathbf{F}_t$ to get smoothed $\hat{\mathbf{u}}_{t-k+1:t}^+$
    Recompute cumulative loss: $\sum_{\tau=t-k+1}^{t}\mathcal{L}_\tau(\mathbf{F})$
    for each $\theta \in \{\mathbf{F}_t, \lambda^{(t)}, \gamma^{(t)}, \mu^{(t)}\}$ **do** ▷ Adam on $\mathbf{F}$ and hyperparams.
      Update $\theta \leftarrow \text{Adam}(\nabla_\theta \sum \mathcal{L}_\tau^{(F)}, \theta)$
    end for
  end if
end for

5.3 Learned Sparsity Mask

In many applications, the exact sparsity structure of the transition matrix is not available. While a mask may be derived from domain knowledge, such information is often incomplete. It is therefore advantageous to infer a mask directly from data. Having a sparsity mask involved in the learning of $\mathbf{F}$ is advantageous because most link values would likely be influenced by its own previous value and a small amount of other links. In other words, there is a small finite set of links which influence the current link lead-

ing to a sparse $\mathbf{F}$. Forcing all entries of $\mathbf{F}$ to be potentially active could more often lead to an underdetermined system with poor identifiability. A learned mask provides a compromise in that it reduces the effective number of parameters while still allowing the system to reveal which connections are active.

In streaming settings, we adopt a windowed (buffered) online pipeline. The approach follows sparse regression with iterative thresholding (sequentially thresholded regression), introduced by Brunton et al. [32] and proven on the task of sparse identification of nonlinear dynamic systems. For a given time t we (i) run the joint estimator without a mask (unknown-mask for $\mathbf{F}$) to obtain state estimates in a finite buffer, (ii) learn a binary mask on this sliding window via ridge and thresholding (Algorithm 6), and (iii) re-run joint estimation using the learned mask and periodically refresh it as the window advances. Thus, mask discovery is performed with a *windowed batch approach*. Of note, is that the original mask is created using state estimates when the sparsity pattern of $\mathbf{F}$ is unknown. These are noisy state estimates. Therefore, we emphasize recall in learning the mask such that the learned mask may slightly over-estimate the true support. It is logical to assume small "extra" entries in the sparsity mask not in the true support can be regularized toward zero yielding only a moderate impact on relative error.

The learned mask step requires enough samples in the buffer to support a stable regression. During startup, when the buffer length is below a minimum threshold B_{mask}, it is preferable to use an identity mask $\mathbf{M} = \mathbf{I}$. This means that each edge is assumed to evolve primarily through its own history. Although this choice is not correct, it is the most neutral assumption when insufficient data are available to form a more reliable estimate. Once the buffer reaches B_{mask}, the mask is updated by running the learned mask discovery procedure on the available data. Empirical experiments show that this startup rule avoids the instability of using an underdetermined and noisy learned mask, and while the identity mask may incur some early error, it is generally less severe and quickly corrected once the buffer fills.

5.3.1 Learned Sparsity via Ridge Regression and Thresholding

We set the buffer size to be the same size used in the estimation of the $\mathbf{F}$ matrix, a natural choice given it is of the same dimension. The threshold can be tuned to provide a desired amount of sparsity in the state dynamics. The refresh cadence is the same as for the estimated state dynamics.

The learned mask approach is useful when the sparsity pattern is unknown. It operates as a windowed batch step within an online pipeline, unknown mask estimation then windowed mask learning then masked reestimation.

5.4 Observability and Identifiability Analysis for F

The joint recovery of $\mathbf{F}$ and the reduced hidden state $\mathbf{u}_t$ is only possible when a finite buffer of data contains enough independent information. Recall the reduced measurement model, $\mathbf{y}_t \approx \mathbf{H}_t\mathbf{E}\,\mathbf{F}\,\hat{\mathbf{u}}_{t-1}^{+}$ where $\mathbf{y}_t \in \mathbb{R}^N$ which can be vectorized into a linear form as, $\mathbf{y}_t = \left((\hat{\mathbf{u}}_{t-1}^{+})^\top \otimes \mathbf{H}_t\mathbf{E}\right)\text{vec}(\mathbf{F})$. We can define the shorthand matrix,

Algorithm 6 Learned Sparsity Mask Discovery windowed ridge and thresholding

Inputs
  window of reduced states $\{\hat{\mathbf{u}}_\tau^+\}_{\tau=t-L+1}^{t}$, ridge parameter λ, threshold τ_{thr}, maximum iterations `max_iters` $= 10$
Build stacked matrices design and response over the window
  $\mathbf{B} \leftarrow \left[\hat{\mathbf{u}}_{t-L+1}^+, \ldots, \hat{\mathbf{u}}_{t-1}^+\right]^\top \in \mathbb{R}^{(L-1)\times d_{\text{red}}}$
  $\mathbf{A} \leftarrow \left[\hat{\mathbf{u}}_{t-L+2}^+, \ldots, \hat{\mathbf{u}}_{t}^+\right]^\top \in \mathbb{R}^{(L-1)\times d_{\text{red}}}$
Initialize mask $\mathbf{M} \leftarrow \mathbf{0} \in \{0,1\}^{d_{\text{red}}\times d_{\text{red}}}$
for $i = 1, \ldots, d_{\text{red}}$ **do**
  Ridge fit initial, $\hat{\boldsymbol{f}}_i^{(0)} \leftarrow (\mathbf{B}^\top\mathbf{B} + \lambda\mathbf{I})^{-1}\mathbf{B}^\top\mathbf{A}_{:,i}$
  $k \leftarrow 0$, $S^{(-1)} \leftarrow \emptyset$
  while $S^{(k)} \neq S^{(k-1)}$ **and** $k <$ `max_iters` **do**
    Support by thresholding, $S^{(k)} \leftarrow \{j \mid |(\hat{\boldsymbol{f}}_i^{(k)})_j| > \tau_{\text{thr}}\}$
    Refit on support, $\hat{\boldsymbol{f}}_{i,S^{(k)}}^{(k+1)} \leftarrow \left(\mathbf{B}_{:,S^{(k)}}^\top\mathbf{B}_{:,S^{(k)}} + \lambda\mathbf{I}\right)^{-1}\mathbf{B}_{:,S^{(k)}}^\top\mathbf{A}_{:,i}$
    Assemble full vector, initialize $\hat{\boldsymbol{f}}_i^{(k+1)} \leftarrow \mathbf{0}$ then set $(\hat{\boldsymbol{f}}_i^{(k+1)})_{S^{(k)}} \leftarrow \hat{\boldsymbol{f}}_{i,S^{(k)}}^{(k+1)}$
    $k \leftarrow k + 1$
  end while
  Update mask column i, $M_{ji} \leftarrow \mathbb{I}\big(|(\hat{\boldsymbol{f}}_i^{(k)})_j| > \tau_{\text{thr}}\big)$ for $j = 1, \ldots, d_{\text{red}}$
end for
Output learned mask $\mathbf{M} \in \{0,1\}^{d_{\text{red}}\times d_{\text{red}}}$

$\boldsymbol{\Phi}_t = (\hat{\mathbf{u}}_{t-1}^+)^\top \otimes \mathbf{H}_t\mathbf{E}$ where $\boldsymbol{\Phi}_t \in \mathbb{R}^{N\times d^2}$ links the unknown transition parameters to the measurement at time t.

Stacking L consecutive measurements yields,

$$\mathbf{Y} = \begin{bmatrix} \mathbf{y}_{t-L+1} \\ \vdots \\ \mathbf{y}_t \end{bmatrix} \in \mathbb{R}^{NL}, \qquad \boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{\Phi}_{t-L+1} \\ \vdots \\ \boldsymbol{\Phi}_t \end{bmatrix} \in \mathbb{R}^{NL\times d^2}, \tag{5.11}$$

so the windowed regression problem becomes $\mathbf{Y} \approx \boldsymbol{\Phi}\,\text{vec}(\mathbf{F})$.

Identifiability of $\mathbf{F}$ requires that $\boldsymbol{\Phi}$ has sufficient rank relative to the number of free parameters. Let $\text{vec}(\mathbf{F})_{\text{active}}$ denote the entries that remain after enforcing structural constraints (such as zero diagonal or masks). A necessary condition is $\text{rank}(\boldsymbol{\Phi}) \geq \dim\big(\text{vec}(\mathbf{F})_{\text{active}}\big)$. If this holds, the least-squares solution is unique for the active set. Otherwise, multiple $\mathbf{F}$ can explain the same data and the problem is underdetermined.

Priors reduce this indeterminacy by shrinking the solution space. With priors, the effective information matrix becomes,

$$\mathcal{I}_{\text{window}} = \boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \boldsymbol{\Lambda}, \tag{5.12}$$

where $\boldsymbol{\Lambda}$ is a diagonal matrix that collects the penalty weights for different entries of $\mathbf{F}$ depending on the priors imposed (identity, mask, Frobenius). The explicit form of $\boldsymbol{\Lambda}$ is illustrated in Appendix C. Since $\boldsymbol{\Lambda} \succeq 0$, the smallest eigenvalue of $\mathcal{I}_{\text{window}}$ is increased, improving conditioning and stabilizing the estimation. Intuitively, the priors remove directions that are not sufficiently excited by $\boldsymbol{\Phi}$, thereby improving identifiability.

In addition to parameter identifiability, the reduced hidden state must also be observable. For a candidate $\hat{\mathbf{F}}$, the finite-horizon observability matrix is,

$$\mathcal{O}_{t-L+1:t} = \begin{bmatrix} \mathbf{H}_{t-L+1}\mathbf{E} \\ \mathbf{H}_{t-L+2}\mathbf{E}\hat{\mathbf{F}} \\ \vdots \\ \mathbf{H}_t\mathbf{E}\hat{\mathbf{F}}^{L-1} \end{bmatrix} \in \mathbb{R}^{NL \times d_{\text{red}}}. \tag{5.13}$$

The condition $\text{rank}(\mathcal{O}_{t-L+1:t}) = d_{\text{red}}$ ensures that every component of the reduced hidden state influences the output within the buffer. If this rank condition fails, certain directions of $\mathbf{u}_t$ remain hidden, and recovery of $\mathbf{F}$ is not possible.

In practice, both conditions are checked together. The rank of $\mathbf{\Phi}$ relative to the active dimension indicates whether the transition parameters are identifiable, while the rank of $\mathcal{O}_{t-L+1:t}$ indicates whether the reduced state is observable. Conditioning of either matrix reflects sensitivity to noise: small eigenvalues correspond to weakly excited directions, while larger smallest eigenvalues of $\mathcal{I}_{\text{window}}$ or $\mathcal{O}_{t-L+1:t}$ indicate stronger effective identifiability.

5.5 Experiments

The joint estimation algorithm is tested on synthetic data under the Random Walk and Block Coupling regimes similar to Chapter 4. Random Walk corresponds to $\mathbf{F} = \mathbf{I}$, while Block Coupling is modeled with $\mathbf{F} = \mathbf{I}+\mathbf{C}$, where $\mathbf{C}$ is sparse and off-diagonal with entries in $[0.1, 0.3]$. A similar row-normalization approach for bounding the adjacency values as used in Chapter 4 is utilized. Throughout the experiments in this chapter, the noise covariances are set to their ground-truth values used for data generation, i.e., $\mathbf{Q} = \sigma^2_{\text{proc.}}\mathbf{I}$ and $\mathbf{R} = \sigma^2_{\text{meas.}}\mathbf{I}$.

Performance is assessed visually and numerically. Visual results show trajectories of $\hat{\mathbf{F}}_t$ and $\hat{\mathbf{A}}_t$ alongside ground truth. Numerical performance uses the relative error as defined in Chapter 3 and used also in Chapter 4.

For learned masks, recovery accuracy is measured with precision, recall, and F1 score [33]. These metrics treat mask discovery as a binary classification task, since entries in $\mathbf{M}$ are either active or inactive [28],

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad \text{F1} = \frac{2\,\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{5.14}$$

where TP are true positives, FP are false positives, and FN are false negatives.

5.6 Results

We evaluate joint estimation under challenging evolution scenarios, over various time horizons, and sparsity regimes. In contrast to Chapter 4, both $\mathbf{A}$ and $\mathbf{F}$ are unknown and estimated by Algorithm 5. Results are shown through the learned $\hat{\mathbf{A}}_t$ and $\hat{\mathbf{F}}_t$. A close-up view of the state transition matrix at the final time step is also included

to illustrate the algorithm's learning capability. Visual results for the Random Walk joint estimation have also been tested but are not included as they directly support the conclusions from the Block Coupling scenario and are mathematically an easier regime to learn.

The section is organized around the following questions:

1. Is the system observable and identifiable in the dynamic setting?
2. Is a joint estimation algorithm beneficial compared to applying the algorithm from Chapter 4 to estimate $\mathbf{A}$ with fixed $\mathbf{F} = \mathbf{I}$?
3. Is an involved algorithm for estimating $\mathbf{F}$ worth the complexity trade-off compared to a simple ridge regression approach?
4. How does the joint estimation algorithm perform in estimating adjacency matrices that evolve dynamically under different scenarios? What role do the learned hyperparameters and structural priors play? How does performance vary across different time horizons?
5. How does knowledge of the sparsity mask for $\mathbf{F}$ affects the joint tracking performance?
6. In practical settings, the sparsity mask of $\mathbf{F}$ is typically unknown. Can the sparsity mask be learned within the joint algorithm?
7. Is learning the sparsity mask useful in practice?
8. How does the joint tracking algorithm perform over long time horizons?

By answering these questions, the section evaluates the strengths and limitations of the joint estimation algorithm to identify conditions under which it can be applied to non-synthetic domains.

5.6.1 Identifiability, Observability, and Effect of Priors

Identifiability and observability are vital to check before proceeding with joint estimation. Diagnostics are reported for $N = 10$, $L = 200$. Full rank of $\mathbf{\Phi}$ is needed for identifiability. Full rank of $\mathcal{O}_L$ is needed for observability across the buffer L. From Chapter 4, it is known that condition numbers worsen for increased amount of off diagonal couplings in $\mathbf{F}$.

Table 5.1 summarizes the diagnostics. Both Random Walk and Block Coupling satisfy identifiability and observability, though Block Coupling shows poorer conditioning of the observability matrix, consistent with its stronger latent couplings. The condition number of $\mathbf{\Phi}$ for Random Walk is notably slightly worse than for Block Coupling (8.95×10^3 vs. 1.45×10^3). This may be because the states under Random Walk only vary by noise contributions. The rows in $\mathbf{\Phi}$ would therefore be more correlated. Priors raise the smallest eigenvalue of the information matrix from nearly zero to about 6×10^{-6}, improving conditioning by six orders of magnitude and ensuring stable estimation. With these results, it is safe to proceed with joint estimation.

Table 5.1: Identifiability and observability diagnostics for $N = 10$, $d_{\text{red}} = 45$, buffer $L = 200$. Both scenarios are identifiable and observable, with priors improving conditioning by several orders of magnitude.

Metric	Random Walk	Block Coupling
$\mathbf{F}$ sparsity	45/2025 (2.2%)	146/2025 (7.2%)
condition num.($\mathbf{\Phi}$)	8.95×10^{3}	1.45×10^{3}
Identifiable	Yes	Yes
$\mathcal{O}$ rank / shape	45/2000×45	45/2000×45
condition num.($\mathcal{O}$)	1.41	61.4
Observable	Yes	Yes
$\lambda_{\min}(\mathbf{\Phi}^{\top}\mathbf{\Phi})$	-9.15×10^{-15}	-2.25×10^{-16}
$\lambda_{\min}(\mathbf{\Phi}^{\top}\mathbf{\Phi} + \mathbf{\Lambda})$	6.23×10^{-6}	6.04×10^{-6}
Improvement factor	6.23×10^{6}	6.04×10^{6}

5.6.2 Using Identity as State Dynamics

When knowledge of the state dynamics are unknown, the claim could be made to simply use $\mathbf{F} = \mathbf{I}$. This is not an unreasonable estimation as it has been previously mentioned that the current edge value is likely to be highly influenced by its previous value. When the true dynamics are Block Coupling, this prior is improper resulting in a high relative error persistent across time, see Figure 5.1. Visually, the adjacency is also not properly being reconstructed.

5.6.3 Solving for State Dynamics via Ridge Regression

The first experiment considers joint estimation with $\mathbf{F}$ updated using simple ridge regression. The goal is to establish whether such a baseline can capture the evolution of the state dynamics. A buffer of up to 400 samples is used to give the algorithm favorable conditions. The sparsity mask of $\mathbf{F}$ is assumed to be known, making the setting further favorable for ridge regression.

The ridge regression problem includes a data fidelity term and a norm to encourage the values in the solution to be small. It can be analytically calculated as,

$$\hat{\mathbf{F}} = \left(\mathbf{X}^{\top}\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^{\top}\mathbf{Y} \tag{5.15}$$

Once sufficient samples are available, the ridge solution converges to a fixed matrix that remains almost unchanged over time, Figure 5.1. This "frozen" behavior is expected as ridge regression solves a static optimization problem with strong priors. Ridge regression is inadequate for joint estimation, even with a known sparsity mask and large buffers. More advanced methods, developed in later sections, are required to incorporate structural priors, enforce temporal consistency, and apply retroactive smoothing to track system dynamics recorded in $\mathbf{F}$.

When the sparsity pattern of $\mathbf{F}$ is known a priori, the joint algorithm only estimates the active entries, while the remainder are fixed by the mask. This represents a favorable setting. All structural priors, adaptive learning rates, and retroactive smoothing are enabled. Results for the Block Coupling scenario with $T = 1{,}000$ samples are shown in

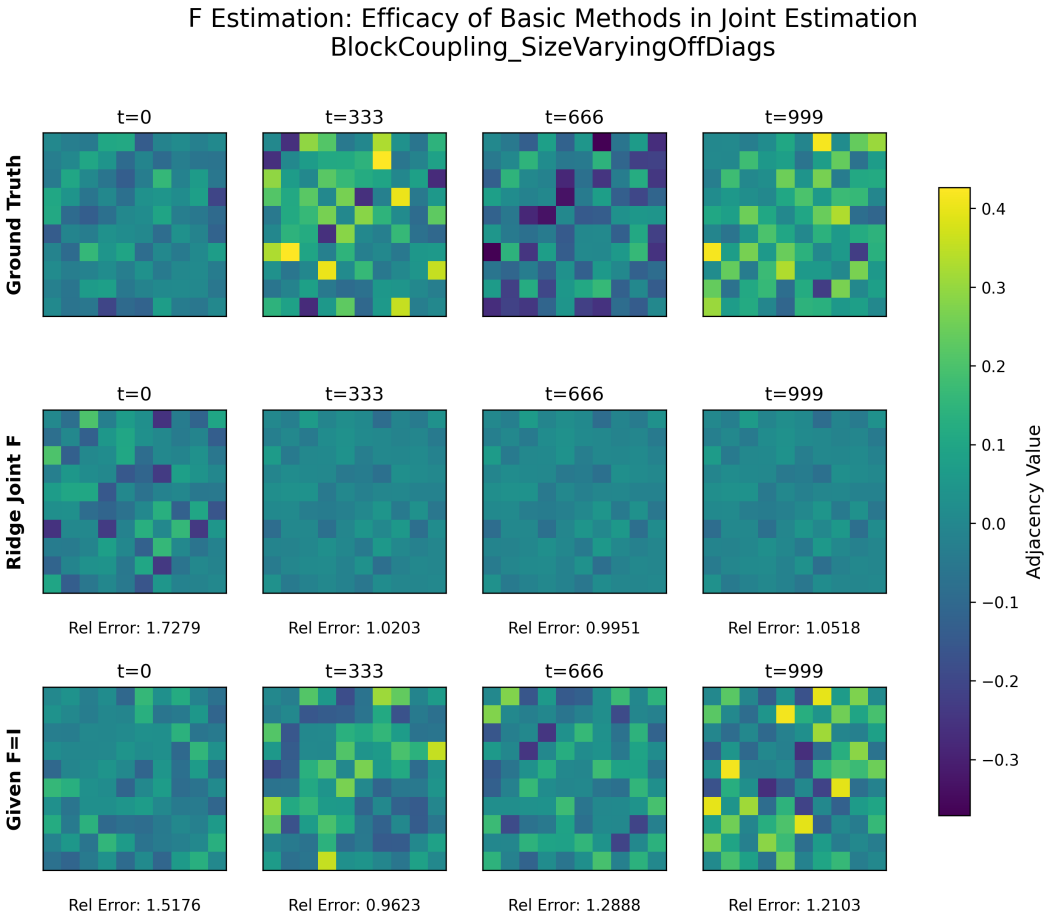


Figure 5.1: Block Coupling scenario: estimation of $\mathbf{F}$ using ridge regression. Ridge produces nearly static solutions, highlighting the need for advanced joint estimation approaches. Ridge estimation is poor but still provides marginal benefits over using $\mathbf{F} = \mathbf{I}$

Figure 5.2. Adjacency tracking performance is close to the baseline using the ground-truth $\mathbf{F}$, and relative errors converge to the same level as the baseline where the state dynamics are known. The estimate $\hat{\mathbf{F}}_t$ approaches the true dynamics within a few buffer lengths, with a final relative error below 0.11.

5.6.4 Known State Dynamics Mask: Joint Estimation Results

At $T = 10{,}000$ (Figure 5.3), accuracy is maintained indicating that error is appropriately managed over time and that the solution does not drift. The learned $\hat{\mathbf{F}}$ continues to match the ground truth, including weak off-diagonal entries, with relative error below 0.10. Errors in adjacency estimates are dominated by the Kalman filter step rather than by learning $\mathbf{F}$. This confirms that the joint algorithm can reliably track both adjacency and state dynamics when the sparsity mask is provided.

The main errors in the estimated adjacencies are therefore due to the estimation of $\mathbf{A}$ via the Kalman Filter. These can also primarily be attributed to the fast varying

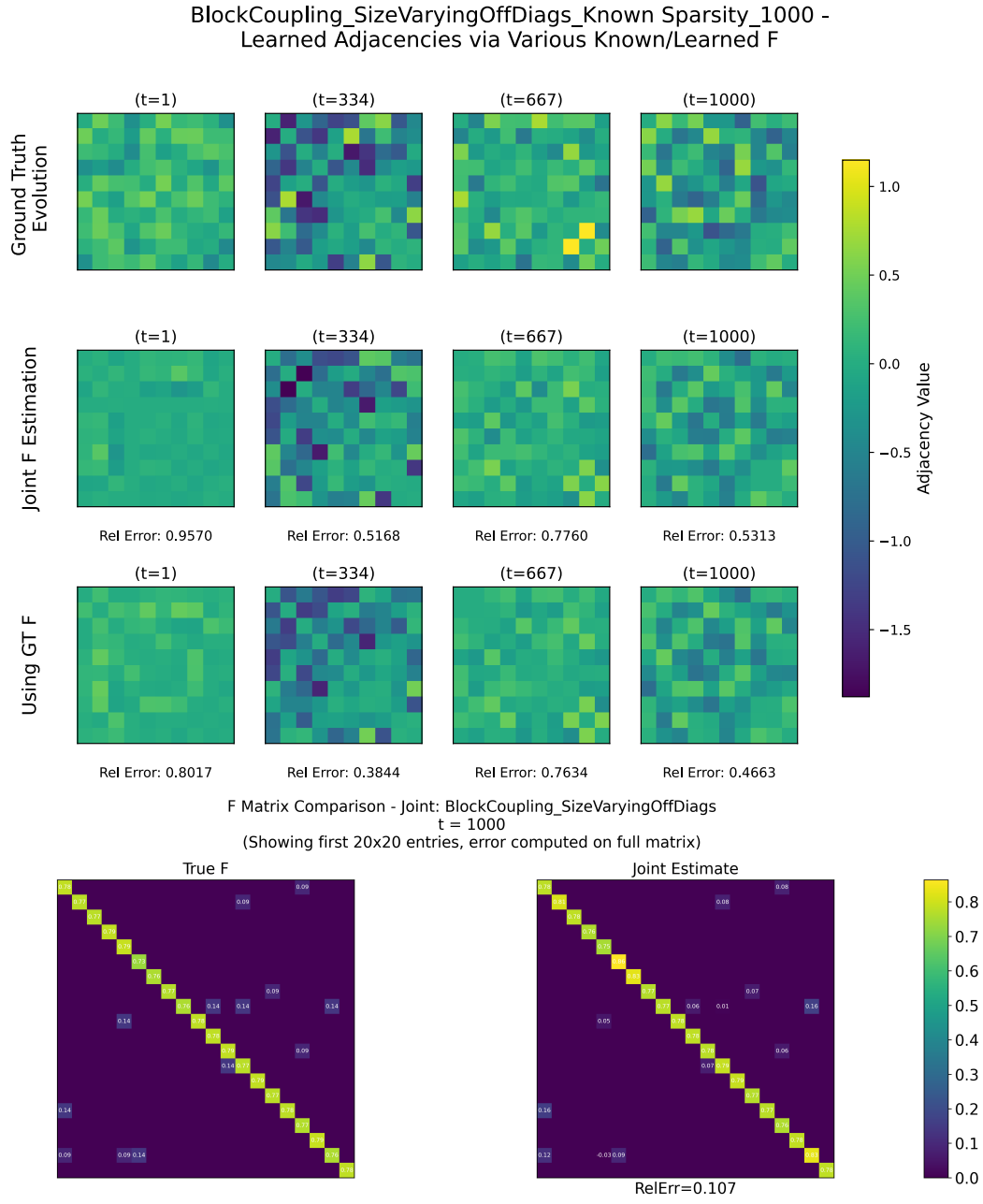


Figure 5.2: Block Coupling with known sparsity mask. (a) Joint estimation tracks performance close to the oracle baseline. Using GT F refers to Ground Truth $\mathbf{F}$ a.k.a. the Known $\mathbf{F}$ scenario. (b) The final estimated $\hat{\mathbf{F}}$ matches the ground-truth $\mathbf{F}$ with low relative error.

dynamics of the adjacency as Chapter 3 demonstrated that for a fixed adjacency the tracking algorithm exhibits very low error.

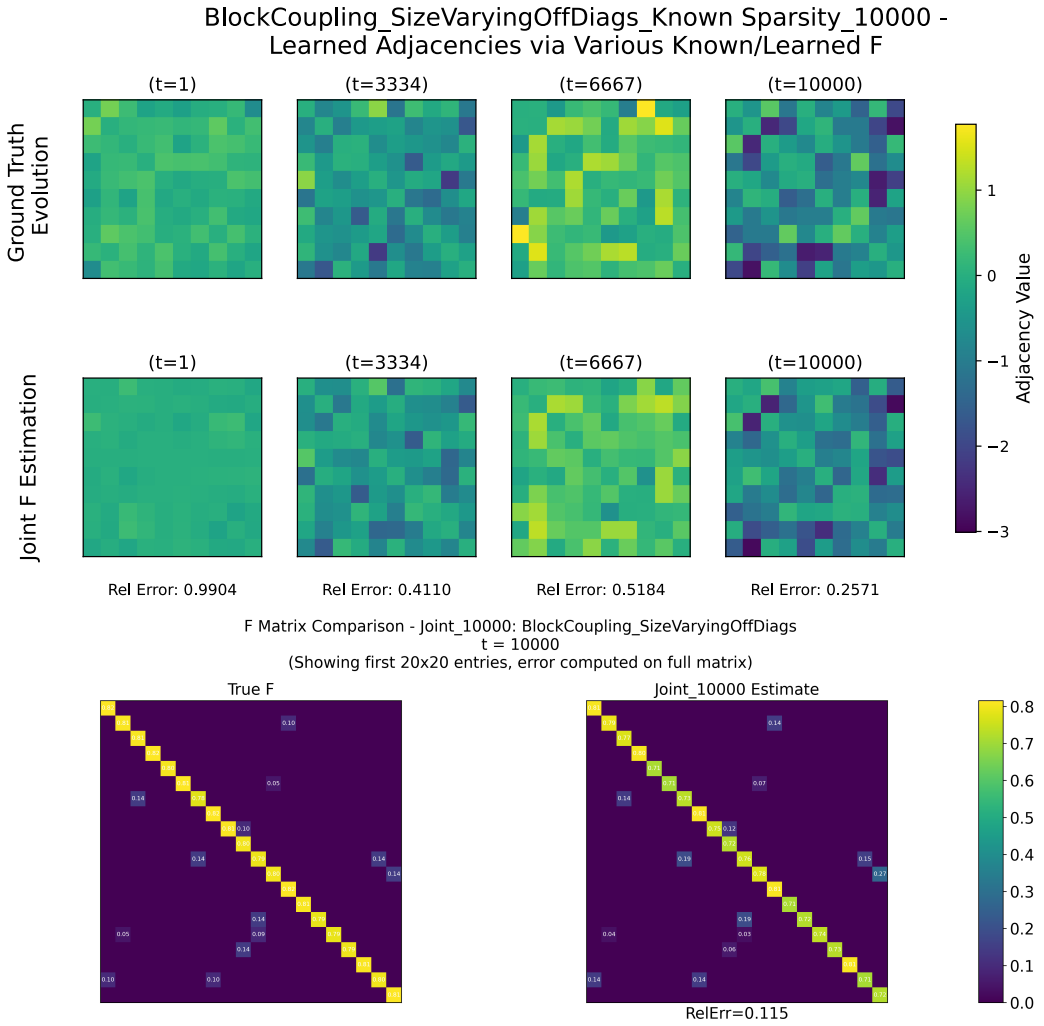


Figure 5.3: Block Coupling with known sparsity mask at $T = 10{,}000$. (a) Joint estimation tracks $\mathbf{F}$ effectively over the long horizon. (b) The final $\hat{\mathbf{F}}$ closely matches the true $\mathbf{F}^\star$, including weak off-diagonal entries.

5.6.5 Unknown State Dynamics Mask: Joint Estimation Results

Without knowledge of the sparsity pattern, all entries of $\mathbf{F}$ are estimated. This setting is more challenging but representative of real-world scenarios. The algorithm again uses priors, adaptive learning, and retroactive smoothing.

For Block Coupling with $T = 1{,}000$ (Figure 5.4), the adjacency errors are higher than in the known-sparsity case, but the evolving structure is still captured. The stacked trajectory of $\hat{\mathbf{F}}_t$ shows that the diagonal is well preserved, while noise contaminates the off-diagonals increasingly per iteration obscuring their true value. At $T = 10{,}000$ (Figure 5.5), estimates remain stable and track the diagonal structure, though off-diagonals remain very noisy. Despite weaker performance, the algorithm avoids divergence and achieves better adjacency accuracy over time coming close to the Relative Error results

when the mask is known.

When the horizon is extended to $T = 10,000$, the results in Figure 5.5 that the algorithm continues to track the dynamics without divergence despite this more challenging learning setting. This indicates that the algorithm can overcome error accumulation, a significant concern in dynamic GTI. The adjacency errors remain slightly higher than in the known sparsity case, but the estimates are stable and the diagonal structure of $\mathbf{F}$ remains accurate.

5.6.6 Learned State Dynamics Mask: Joint Estimation Results

It has been assumed that either full knowledge of $\mathbf{F}$ support has been provided or that there is no knowledge at all of the support. The learned sparsity experiments explore an intermediate regime. They seek to answer the question, can a useful mask be discovered directly from the data, and if so, does it improve joint estimation?

The learned sparsity and joint estimation procedure to track time-varying adjacencies is multi-pass. First, the joint algorithm runs without any mask, producing preliminary state estimates. A mask is then inferred from these states using a regression–thresholding procedure. Finally, the joint algorithm is re-run with the learned mask enforced. This setup adds computational cost but provides a data-driven constraint. All streaming experiments follow the online pipeline in Section 5.3.

5.6.7 Learning Mask from Ground-truth States vs. Learned States

To test the mask discovery step, a test is conducted using the ground-truth hidden states. Figure 5.6a shows that the recovered support captures the true support of $\mathbf{F}$. The diagonal term is accurately recovered, and many of the active off-diagonal entries are correctly identified. The relative error is low. Paired with the plot provided in Figure 5.6a, this indicates that the error is primarily due to the mismatch in the matrix-entry value. This result confirms that the regression–thresholding procedure can recover meaningful sparsity when clean states are available.

Realistically, the states will not be ground truth but a noisy filtered version from the Kalman Filter-based estimation method. The question remains whether the sparse support of $\mathbf{F}$ can be appropriately recovered from these noisy states. The noisy states are and must be obtained from the joint estimation setting where the true sparsity pattern is unknown. As Figure 5.6b demonstrates, the discovered mask can partially reconstruct the true support. The diagonal elements are preserved, and several dominant off-diagonal connections are identified. However, false positives do appear, and other entries remain unrecognized in the learned support. The relative error for the learned $\mathbf{F}$ rises to approximately 0.24 compared to 0.102 when the true states were used.

5.6.8 Performance Comparison With and Without Learned Mask

Figure 5.7a compare adjacency tracking under three regimes for learning state dynamics: known sparsity, learned sparsity, and unknown sparsity. The learned-mask configuration consistently improves performance relative to the unknown case. Errors decrease more quickly, and the adjacency evolution follows the ground truth more closely. The

improvement is not as strong as in the known-sparsity baseline, but the gap in relative error narrows with longer horizons.

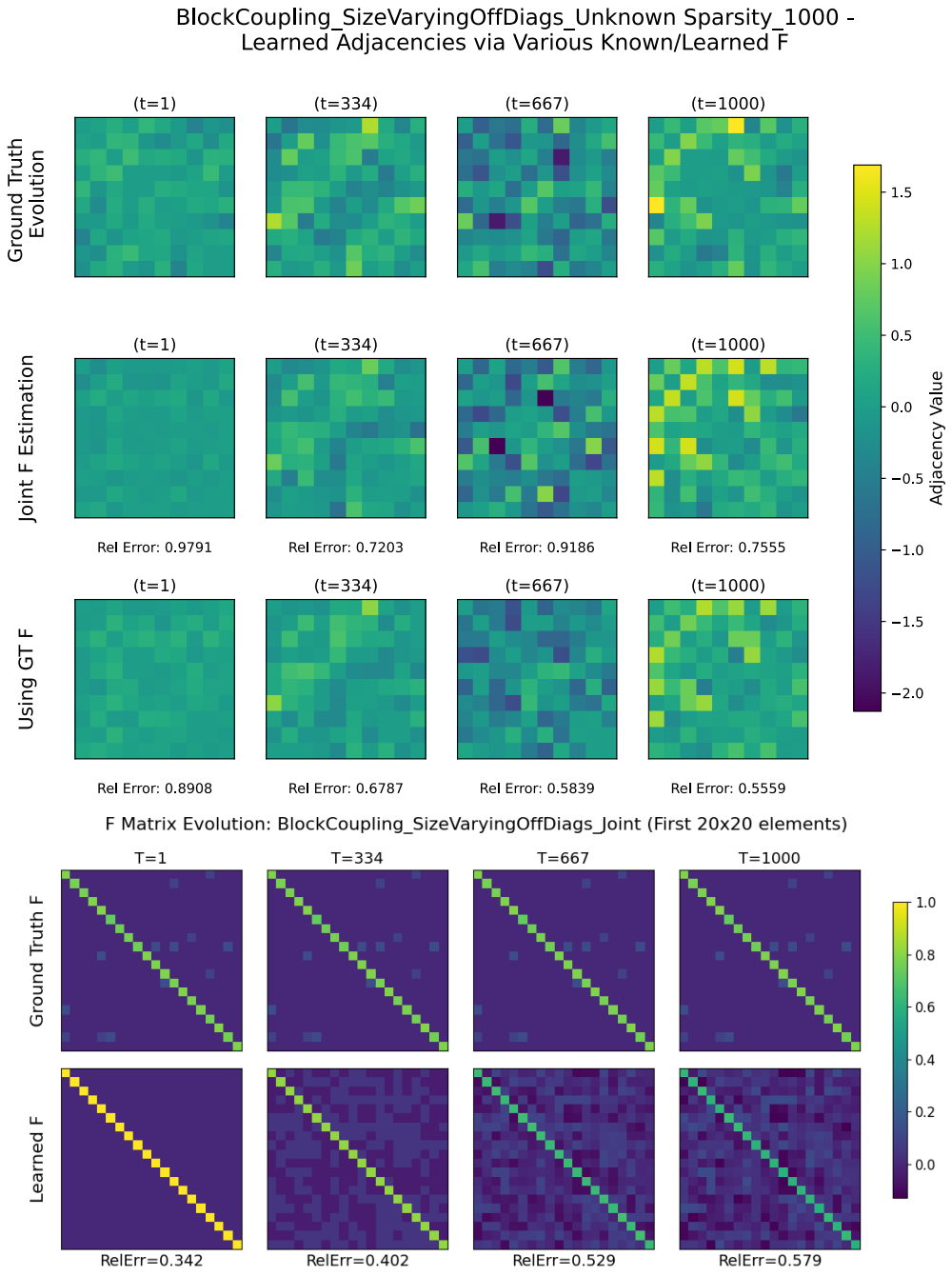


Figure 5.4: Block Coupling with unknown sparsity mask. (a) Adjacency tracking compared with ground-truth and fixed $\mathbf{F} = \mathbf{I}$. (b) Evolution of the estimated transition matrices shows preserved diagonal structure with increasingly noisy off-diagonals.

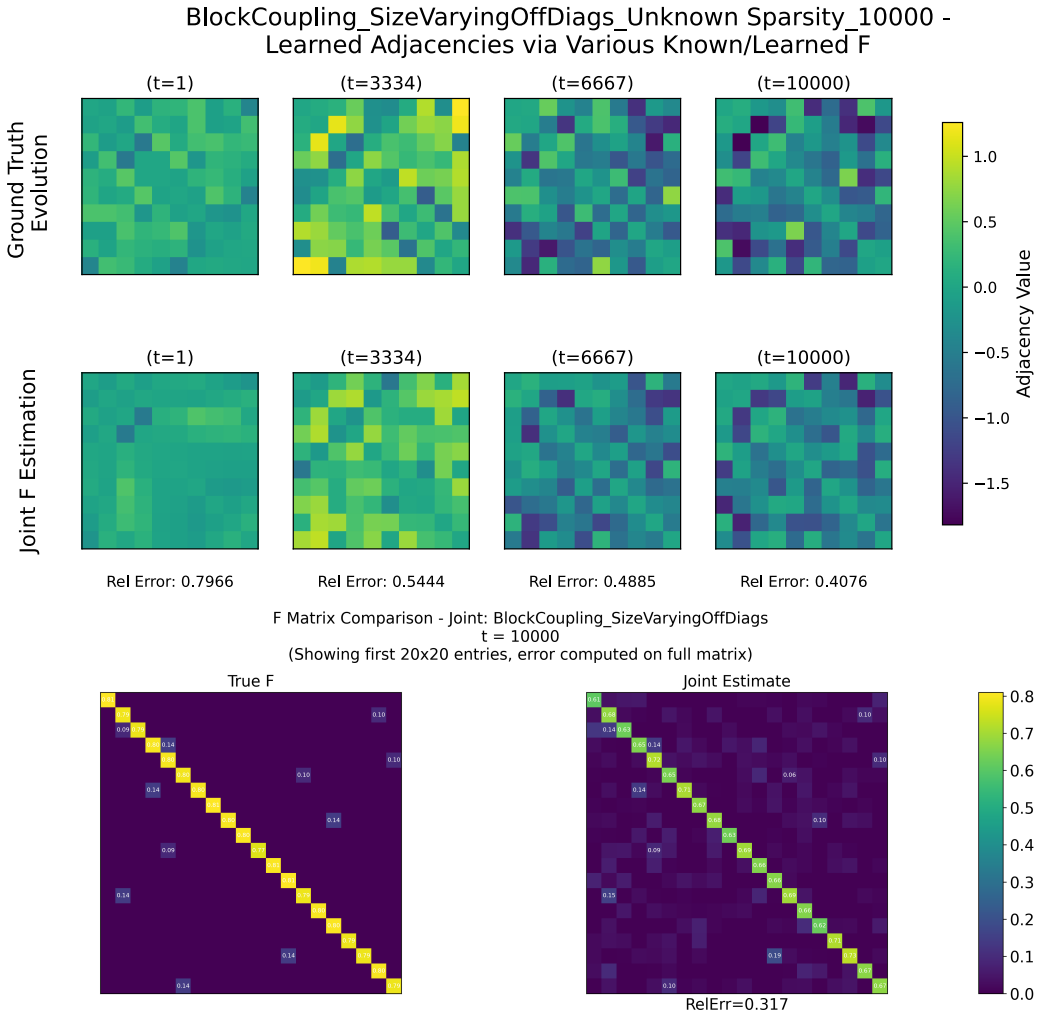


Figure 5.5: Block Coupling with unknown sparsity at $T = 10{,}000$. (a) Joint estimation tracks adjacency stably but less cleanly than with a known mask. (b) The final $\hat{\mathbf{F}}$ matches the diagonal well, but off-diagonals remain noisy.

The main drawback of the learned state algorithm is computational due to its multi-pass requirement. Therefore, when computation is not a limiting factor, the learned-mask setup is recommended. When efficiency is more important, the unknown-mask run remains advantageous, though less accurate. The same conclusions hold for long time horizons. With $T = 10,000$, Block Coupling maintains stable tracking under the learned mask. Figure 5.7b shows that the learned-mask configuration continues to outperform the unknown case while approaching the performance of the known-mask baseline.

Learning a sparsity mask provides clear benefits. With ground-truth states, the recovery is almost exact, and even with noisy estimated states, the mask improves performance. The approach narrows the gap between unknown and known sparsity settings, though it does not eliminate it. It also highlights the familiar chicken-and-egg

issue that good states are needed to identify a useful mask, while good masks help in recovering accurate states.

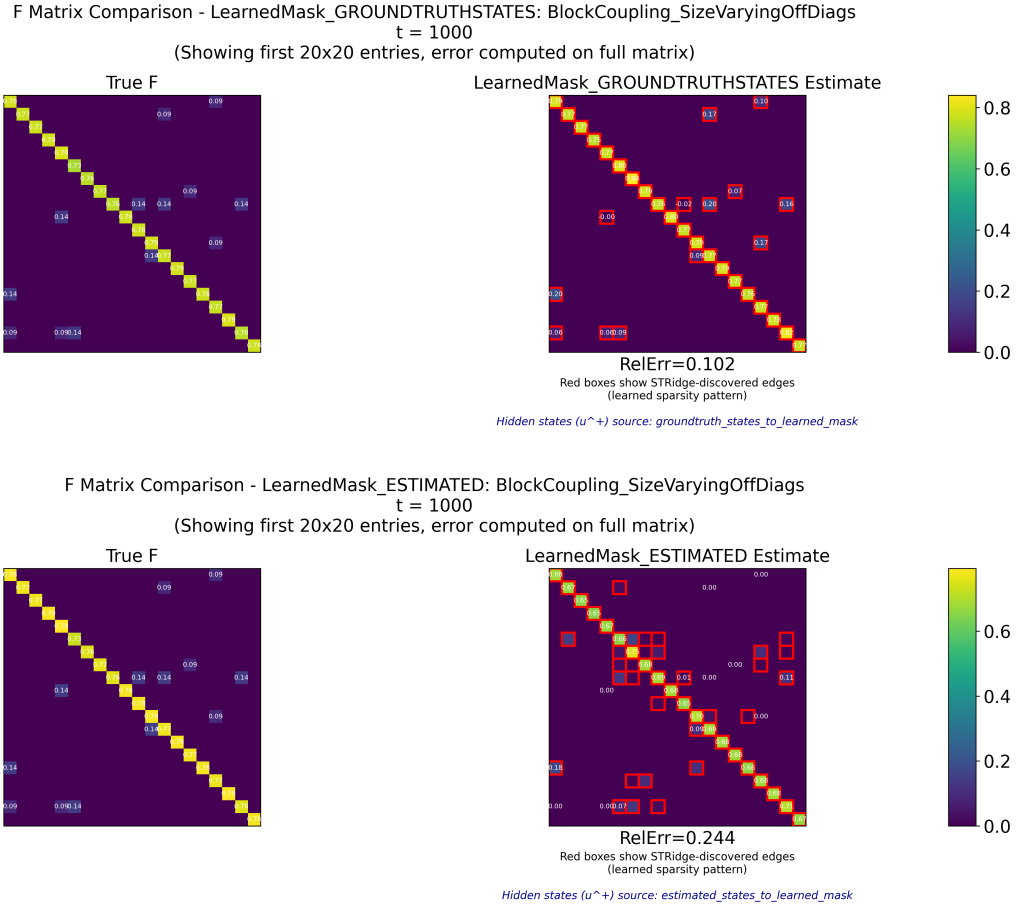


Figure 5.6: Block Coupling: Learned mask recovery. (a) Using ground-truth states yields accurate diagonal and several off-diagonal entries. (b) Using estimated states introduces noise and false positives but retains key connections.

5.7 Conclusion

In this chapter, the online joint estimation algorithm of a dynamic adjacency and its state transition under three sparsity regimes, known, unknown, and learned, is extensively evaluated. The streaming setup follows Algorithm 5 and updates $\mathbf{A}$, $\mathbf{F}$, and the transition–learning hyperparameters during operation. The sparsity mask can also be learned through a multi-pass pipeline, trading short-horizon accuracy for improved estimation performance.

Through testing, it has been demonstrated that simple ridge updates for $\mathbf{F}$ are not adequate. Therefore, an involved state transition estimator with priors, adaptive step sizes, and retroactive smoothing is proposed, implemented, and tested.

With a known mask, joint estimation reaches performance close to the ground truth $\mathbf{F}$ baseline. This holds over long horizons (e.g., $T = 10,000$), where the learned transition remains stable and adjacency errors stay near the ground truth curve. This scenario is tested for both Block Coupling and Random Walk, indicating that performance generalizes across dynamics. The priors effectively reduce the parameter space, lower variance, and suppress extra couplings and guide the algorithm to the correct result.

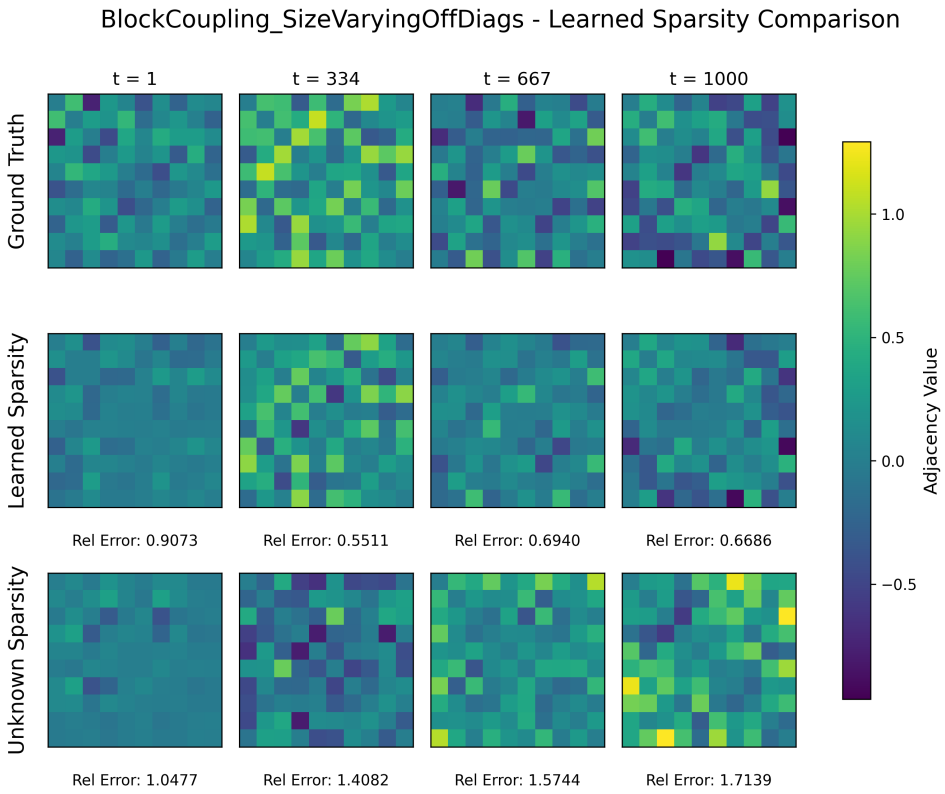
When the mask is unknown, early tracking is worse but improves steadily as more data accumulate. Estimating $\mathbf{F}$ jointly—even with no prior sparsity—still outperforms using a fixed, unknown $\mathbf{F} = \mathbf{I}$ in terms of adjacency accuracy. The relative error gap narrows over time toward the known-sparsity case without noise drift, which is promising for real-world use.

Learning a mask from data offers a useful compromise. It improves early performance (notably in the first 1000 steps) and remains slightly better than the unknown-sparsity scenario at long horizons, at the cost of a multi-pass procedure that increases computational complexity.

Based on these results, dynamic GTI can be performed using a graph state-space model and the proposed joint estimation approach to estimate time-varying adjacencies. Use a reliable sparsity mask for the state transition if available. Otherwise, learn a sparse support when early accuracy matters and computation is available. If the computation is limited and a longer horizon is acceptable, unknown-mask joint estimation remains advantageous. Overall, because the adjacency, the state transition, and the transition-learning hyperparameters are all learned online, the results are strong and approach ground-truth evolution. The algorithm is designed around information realistically available in practice, aiding transition from synthetic to real-world experimentation.

Summary of Key Conclusions

- The proposed online scheme jointly estimating the state transition $\mathbf{F}$ and the adjacency (via $\mathbf{u}_t$) is feasible and stable across the tested co-evolution scenarios.
- When a correct sparsity mask for $\mathbf{F}$ is available, convergence is faster and estimation errors are lower for both $\hat{\mathbf{F}}$ and $\hat{\mathbf{A}}_t$.
- In the absence of sparsity information, joint estimation remains effective. Given sufficient horizon, $\hat{\mathbf{F}}$ and $\hat{\mathbf{A}}_t$ approach the performance of the known-mask baseline.
- A windowed mask learned from noisy state estimates reduces the effective parameter dimension, improves early-horizon accuracy, and maintains a consistent advantage over the unknown-mask regime.
- Performance is strongest for non-homogeneous networks where edges evolve independently or only weakly co-evolve. Highly homogenizing dynamics (consensus-like networks) fall outside the intended scope.
- Structural priors and regularization (mask, diagonal, ridge) improve the conditioning of the window information matrix strengthening effective identifiability.



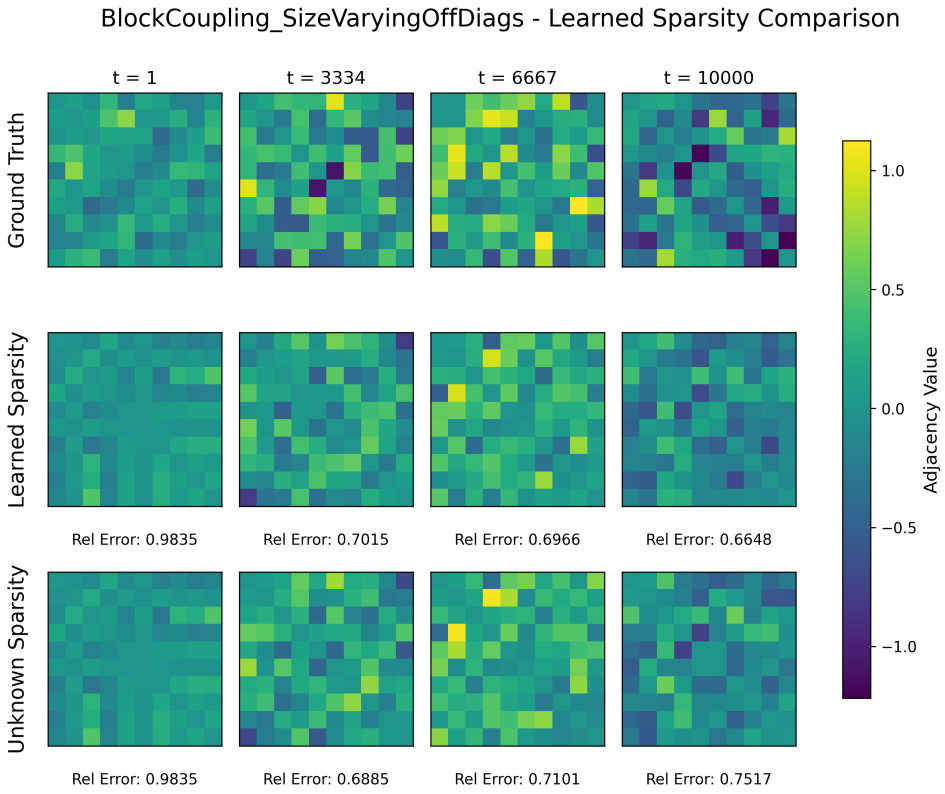


Figure 5.7: Block Coupling: Learned sparsity mask performance. (a) At moderate horizon, learned sparsity improves recovery relative to the unknown mask. (b) At long horizon $T = 10{,}000$, the advantage persists.

6 Performance Comparison

Placing performance in context is an integral step in algorithm development. Here, we consider two other algorithms which perform dynamic GTI using streaming data which can be used for comparison against the proposed Kalman based algorithm. These are the LASSO with a temporal consistency term [34] and the Prediction-Correction Method developed by Natali et al. [31]. These algorithms are chosen as they operate in an online manner providing real-time adjacency estimates for dynamic networks. First, the algorithms are briefly outlined, specifically how they will be applied within an experimental setting. We then compare the performance based on relative error and visual inspection of the reconstructed adjacencies.

6.1 LASSO with Temporal Consistency

A natural baseline for performance comparison within the dynamic GTI setting is sparse and temporally consistent regression. In the static case, the adjacency recovery reduces to the LASSO problem with a data fidelity term and an ℓ_1 penalty to promote sparsity. A similar ridge regression (with an ℓ_2 term) was implemented as a first line approach to estimate the state dynamics in chapter 5 where joint estimation was introduced. The data fidelity and sparsity term follow directly from the original Tibshirani work [34].

In the static scenario, we would stack $\{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1}^{T}$ into matrices $\mathbf{X} \in \mathbb{R}^{N\times T}$ and $\mathbf{Y} \in \mathbb{R}^{N\times T}$. The problem formulation would then be,

$$\hat{\mathbf{A}} = \arg\min_{\mathbf{A}} \ \tfrac{1}{2}\|\mathbf{Y} - \mathbf{A}\mathbf{X}\|_F^2 + \lambda\|\mathbf{A}\|_1, \tag{6.1}$$

where $\|\cdot\|_1$ denotes the elementwise ℓ_1 norm and $\lambda > 0$ is a sparsity parameter. Each row of A can equivalently be recovered by solving a separate LASSO regression of the outputs of that node on the inputs. This is the direct multivariate extension of the base case LASSO.

In the dynamic setting, particularly relevant as this will develop into a comparator method for the proposed Kalman-based dynamic GTI algorithm, the adjacency varies over time according to $\mathbf{A}_t$. To extend Equation 6.1, a temporal consistency term is added to discourage large deviations between sequential adjacency estimates. At each time t, the estimate solves,

$$\hat{\mathbf{A}}_t = \arg\min_{\mathbf{A}\in\mathcal{S}} \left\{ \tfrac{1}{2}\|\mathbf{y}_t - \mathbf{A}\mathbf{x}_t\|_2^2 + \lambda\|\mathbf{A}\|_1 + \beta\|\mathbf{A} - \hat{\mathbf{A}}_{t-1}\|_F^2 \right\}, \tag{6.2}$$

where $\beta \geq 0$ tunes the strength of temporal smoothness and $\mathcal{S}$ enforces structural constraints such as symmetry and zero diagonal. A larger β will yield a smoother but slower response, while small β allows faster adaptation but noisier trajectories. This trade-off

parallels the role of process noise versus measurement noise in the Kalman filter, but here it is encoded by optimization penalties rather than probabilistic modeling.

Equation 6.2 can be solved incrementally with a few proximal gradient iterations per step. Each update uses the current sample $(\mathbf{x}_t, \mathbf{y}_t)$ and the prior estimate $\hat{\mathbf{A}}_{t-1}$ as a warm start, ensuring bounded per-step cost. Projection onto $\mathcal{S}$ after each update maintains the assumed graph structure.

Equation 6.2 provides a direct, consistent baseline compatible with the IO data generation model. Unlike covariance-based methods such as the Time-Varying Graphical Lasso [9], which assume Gaussian graphical models and estimate evolving precision matrices, the formulation here is explicitly matched to the IO model $\mathbf{y}_t = \mathbf{A}_t\mathbf{x}_t + \mathbf{w}_t$ and recovers adjacency through sparse regression with temporal regularization.

6.2 Prediction Correction Algorithm

The prediction-correction framework updates the solution each time by first predicting where the optimizer will move, then correcting with a few proximal/gradient steps on the new cost [31]. Here the cost is the IO least-squares error per time step. Limited iterations serve as an intrinsic temporal regularizer and keep runtime fixed per sample.

Algorithm 7 Prediction-Correction for IO Data Generation Model

Inputs: Stream $\{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1}^{T}$, prediction steps P, correction steps C, step sizes α, β.
Initialize: $\mathbf{A}_0 \leftarrow 0$.
for $t = 0, \ldots, T-1$ **do**
  Prediction (time t+1 using t):
  $\mathbf{S}^{(0)} \leftarrow \mathbf{A}_t$.
  for $p = 0, \ldots, P-1$ **do**
    $\mathbf{S}^{(p+1)} \leftarrow \Pi_{\mathcal{S}}\Big(\mathbf{S}^{(p)} - \alpha \nabla_{\mathbf{S}} \frac{1}{2}\|\mathbf{y}_t - \mathbf{S}\mathbf{x}_t\|_2^2\Big)$.
  end for
  $\hat{\mathbf{A}}_{t+1|t} \leftarrow \mathbf{S}^{(P)}$.
  Correction (at t+1):
  $\mathbf{S}^{(0)} \leftarrow \hat{\mathbf{A}}_{t+1|t}$.
  for $c = 0, \ldots, C-1$ **do**
    $\mathbf{S}^{(c+1)} \leftarrow \Pi_{\mathcal{S}}\Big(\mathbf{S}^{(c)} - \beta \nabla_S \frac{1}{2}\|\mathbf{y}_{t+1} - \mathbf{S}\mathbf{x}_{t+1}\|_2^2\Big)$.
  end for
  $\mathbf{A}_{t+1} \leftarrow \mathbf{S}^{(C)}$.
end for
Output: $\{\mathbf{A}_t\}_{t=1}^{T}$.

6.3 Experiments

The experiments follow the protocol established in Chapters 4 and 5. We test over a horizon of $T = 10{,}000$ time steps on dynamically varying adjacency matrices driven by a linear state transition, as in the thesis' graph state-space model. The observation model is IO, $\mathbf{y}_t = \mathbf{A}_t\mathbf{x}_t + \mathbf{w}_t$ with $\mathbf{x}_t$ known and $\mathbf{w}_t$ Gaussian.

Table 6.1: Comparing streaming algorithms given various state dynamic scenarios. Mean and standard deviation of relative error provided. KalDO denotes the proposed Kalman-based online GTI algorithm. Top performer bolded and colored green.

Experiment	Algorithm	Mean	Std Dev.	Algo. Type	Avg F_{off}	F_{off} Spars.	$\sigma^2_{\text{proc.}}$	$\sigma^2_{\text{meas.}}$
RandomWalk	Known **F** (True)	0.19	0.07	Baseline	–	–	0.10	0.01
	Streaming LASSO	0.28	0.08	Competitor	–	–	0.10	0.01
	Prediction-Correction	**0.25**	**0.07**	Competitor	–	–	0.10	0.01
	Unkno. Spars. KalDO	0.41	0.12	KalDO GTI	–	–	0.10	0.01
BC_SVOffDiag	Known **F** (True)	0.67	0.10	Baseline	0.004	3.6%	0.10	0.01
	Streaming LASSO	0.87	0.13	Competitor	0.004	3.6%	0.10	0.01
	Prediction-Correction	0.87	0.14	Competitor	0.004	3.6%	0.10	0.01
	Unknown Spars. KalDO	**0.81**	**0.13**	KalDO GTI	0.004	3.6%	0.10	0.01
BC_ExtraMeasN.	Known **F** (True)	0.97	0.09	Baseline	0.003	3.0%	0.10	10.00
	Streaming LASSO	3.48	0.68	Competitor	0.003	3.0%	0.10	10.00
	Prediction-Correction	4.63	0.92	Competitor	0.003	3.0%	0.10	10.00
	Unkno. Spars. KalDO	**1.53**	**0.43**	KalDO GTI	0.003	3.0%	0.10	10.00
BC_ExtraProcN.	Known **F** (True)	0.61	0.10	Baseline	0.004	3.3%	10.00	0.01
	Streaming LASSO	0.83	0.13	Competitor	0.004	3.3%	10.00	0.01
	Prediction-Correction	0.82	0.13	Competitor	0.004	3.3%	10.00	0.01
	Unkno. Spars. KalDO	**0.73**	**0.12**	KalDO GTI	0.004	3.3%	10.00	0.01
BC_aggr	Known **F** (True)	0.45	0.21	Baseline	0.009	4.8%	0.10	0.01
	Streaming LASSO	0.61	0.28	Competitor	0.009	4.8%	0.10	0.01
	Prediction-Correction	0.61	0.27	Competitor	0.009	4.8%	0.10	0.01
	Unkno. Spars. KalDO	**0.69**	**0.23**	KalDO GTI	0.009	4.8%	0.10	0.01

6.4 Results and Discussion

We compare LASSO with temporal consistency (competitor 1), the Prediction-Correction (PC) method (competitor 2), and the proposed Kalman-based online GTI (KalDO) under five regimes. Known-**F** uses the true transition matrix with a Kalman filter and serves as an oracle baseline. Relative-error statistics are reported in Table 6.1. Representative adjacency snapshots are shown in Figs. 6.1–6.4 and 6.5. Unless noted, $\mathbf{Q} = \sigma^2_{\text{proc.}}\mathbf{I}$ and $\mathbf{R} = \sigma^2_{\text{meas.}}\mathbf{I}$.

Before the results, note that for block-coupling scenarios we summarize the transition structure by the average off-diagonal magnitude F_{off} and its sparsity (fraction nonzero). These parameters indicate the strength and density of latent coupling and help interpret difficulty.

Random Walk (noise-driven dynamics: $\sigma^2_{\text{proc.}} = 0.10$, $\sigma^2_{\text{meas.}} = 0.01$). All methods are stable and visually follow the ground truth across time in Fig. 6.1. PC has the lowest error (0.25) followed by LASSO (0.28) while the oracle baseline is 0.19. With $\mathbf{F} \approx \mathbf{I}$ the state drifts slowly, so LASSO and PC already capture the evolution with current observations. KalDO with unknown sparsity is higher (0.41). Estimating the transition together with the adjacency in this regime introduces noisy off diagonals in the learned **F** and makes the estimates noisier. Because the transition offers little informative structure, KalDO's recursive updates do not improve over using the measurements directly and the resulting estimates remain noisier than LASSO and PC.

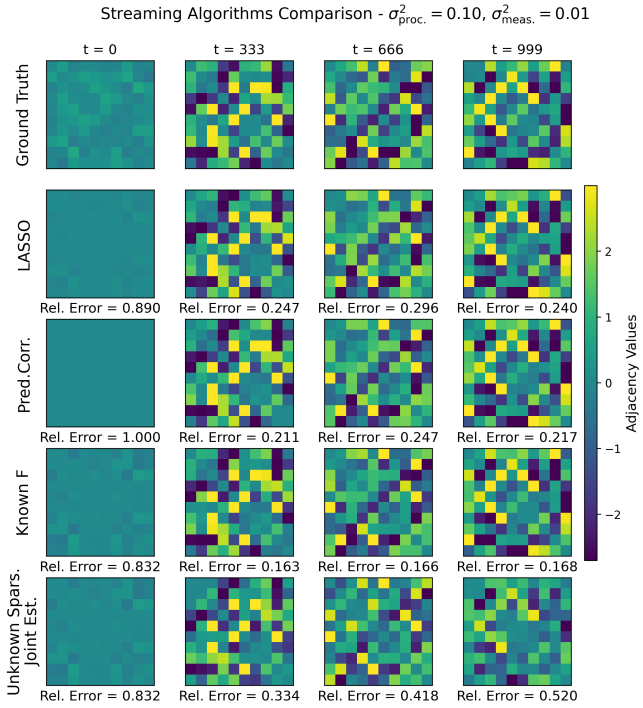


Figure 6.1: Random Walk baseline. $\sigma^2_{\text{proc.}} = 0.10$, $\sigma^2_{\text{meas.}} = 0.01$. All methods are stable and visually provide comparable results.

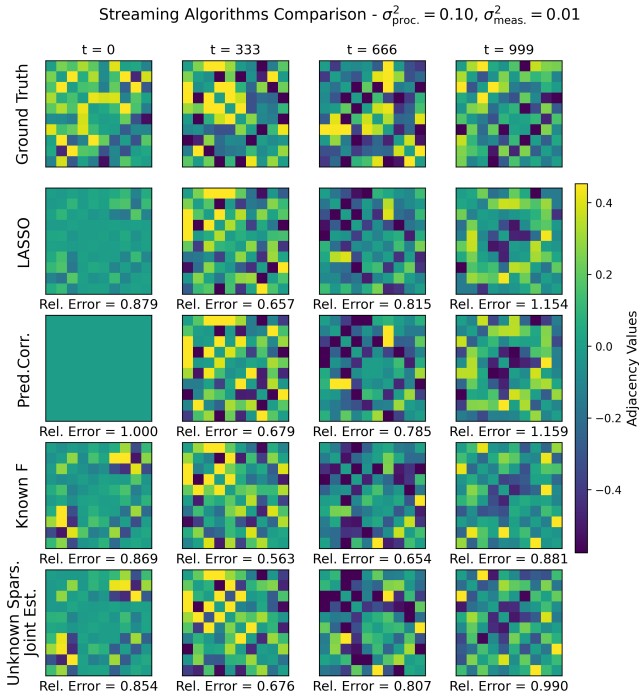


Figure 6.2: Block Coupling size-varying off-diagonals. $\sigma^2_{\text{proc.}} = 0.10$, $\sigma^2_{\text{meas.}} = 0.01$. As off-diagonal coupling increases, error rises across methods, consistent with reduced observability. Known-Spars. KalDO improves over Unknown-Spars. KalDO.

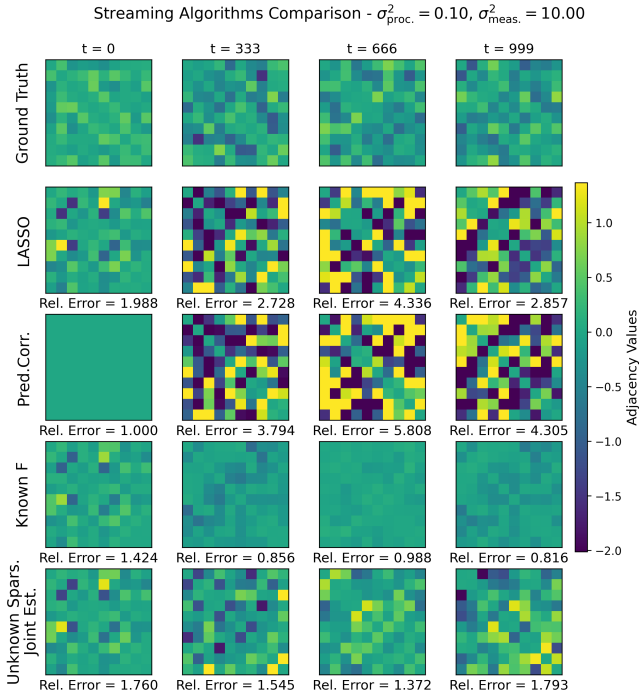


Figure 6.3: Block Coupling size-varying off-diagonals with extra measurement noise. $\sigma^2_{\text{proc.}} = 0.10$, $\sigma^2_{\text{meas.}} = 10.0$. Streaming LASSO and prediction–correction exhibit large errors, while KalDO remains bounded and substantially lower in error.

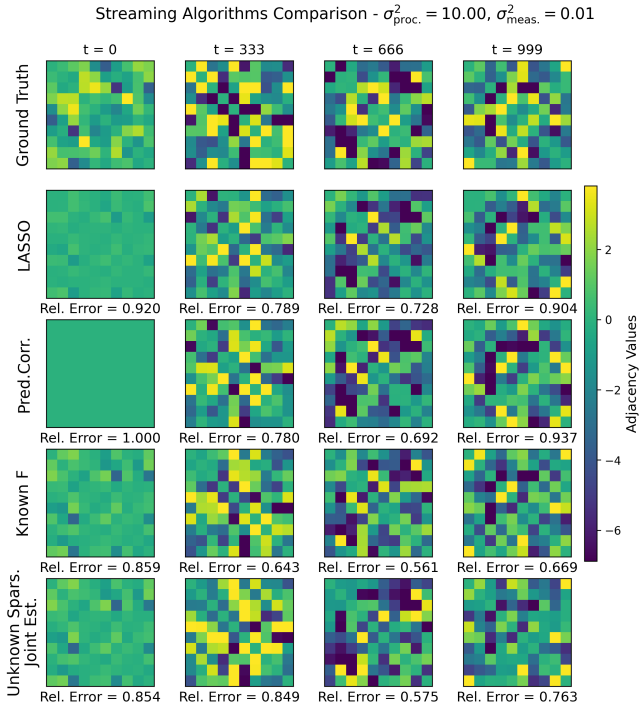


Figure 6.4: Extra process noise. $\sigma^2_{\text{proc.}} = 10.0$, $\sigma^2_{\text{meas.}} = 0.01$. All methods degrade, but KalDO retains a small advantage due to explicit modeling of process uncertainty.

Block Coupling with sparse off-diagonals ($\sigma^2_{\text{proc.}} = 0.10,\ \sigma^2_{\text{meas.}} = 0.01$). When $\mathbf{F}$ contains additional off-diagonal terms, all methods degrade (Fig. 6.2). The coupling mixes state components so each edge depends on others as well as itself. This reduces contrast in the predictions and worsens conditioning of the observability matrix, making some edge directions harder to disentangle from the outputs as discussed in chapter 4. Even with exact dynamics the oracle baseline rises to 0.67 because poor conditioning limits how well any filter can separate directions masked by noise. LASSO and PC are around 0.87 and their estimates are visibly noisier across time. KalDO improves slightly to 0.81 by propagating state uncertainty through its covariance, which tempers updates, but the core limitation remains reduced effective observability under coupling.

Block Coupling with extra measurement noise ($\sigma^2_{\text{proc.}} = 0.10,\ \sigma^2_{\text{meas.}} = 10.0$). Under heavy measurement noise ($\sigma^2_{\text{meas.}} = 10$), errors increase substantially and the effect is visible in Fig. 6.3. The oracle baseline is 0.97 because innovations are noise dominated and must be down-weighted. LASSO and PC become very noisy (3.48 and 4.63) since they do not incorporate $\mathbf{R}$ and the estimates track corrupted measurements. KalDO remains bounded at 1.53 by explicitly using $\mathbf{R}$ in the gain and relying more on predictions when measurements are unreliable. The images show that KalDO tracks the overall structure while LASSO and PC produce very noisy adjacency maps.

Block Coupling with extra process noise ($\sigma^2_{\text{proc.}} = 10.0,\ \sigma^2_{\text{meas.}} = 0.01$). With large process noise ($\sigma^2_{\text{proc.}} = 10$), the true adjacency changes rapidly across time as seen in Fig. 6.4. The oracle baseline is 0.61 indicating that even with exact $\mathbf{F}$ the estimator cannot fully keep up with the injected randomness. LASSO and PC produce higher errors (0.83 and 0.82) while KalDO is modestly better at 0.73 by modeling $\mathbf{Q}$ which increases responsiveness to state changes. The images show that all methods track the coarse pattern but fine contrasts fluctuate, consistent with noise-driven drift.

Block Coupling with aggressive coupling ($\sigma^2_{\text{proc.}} = 0.10,\ \sigma^2_{\text{meas.}} = 0.01$). In the aggregated coupling regime many edges co-evolve simultaneously and contrasts are flattened over time (Fig. 6.5). The oracle baseline is 0.45. LASSO and PC are 0.61 and KalDO is 0.69. Measurement information is not sufficient to separate strongly coupled directions so all methods converge to blurred estimates dominated by medium-valued entries. KalDO stabilizes updates but cannot restore sharpness when conditioning is poor.

Across all regimes, several patterns emerge:

- When dynamics are simple (Random Walk), PC and LASSO perform best. KalDO performs worse because estimating both $\mathbf{F}$ and the adjacency introduces noisy off-diagonal terms in the learned transition. With $\mathbf{F} \approx \mathbf{I}$ there is little dynamic structure to exploit, so recursive updates mainly propagate this noise.
- As latent coupling grows, errors rise across methods due to degraded conditioning of the observability matrix. Even with Known-$\mathbf{F}$ the error increases, showing this limitation is fundamental. KalDO provides a modest advantage by propagating uncertainty through $\mathbf{Q}$ and $\mathbf{R}$, but remains constrained by poor conditioning.

- Under heavy measurement noise, KalDO remains bounded while competitor methods diverge, since it explicitly incorporates $\mathbf{R}$ into the gain. LASSO and PC, lacking this, overfit to corrupted measurements and their estimates become noisy.
- With large process noise, KalDO achieves slightly lower error than competitors by modeling $\mathbf{Q}$, but even the Known-$\mathbf{F}$ baseline shows that no method can fully track rapid noise-driven drift.
- In highly coupled regimes, all methods degrade. Measurements do not contain enough independent information to disentangle coupled directions, so all algorithms converge to blurred estimates dominated by medium-valued entries. KalDO provides reasonable updates but cannot restore sharp contrasts in the adjacency matrix.

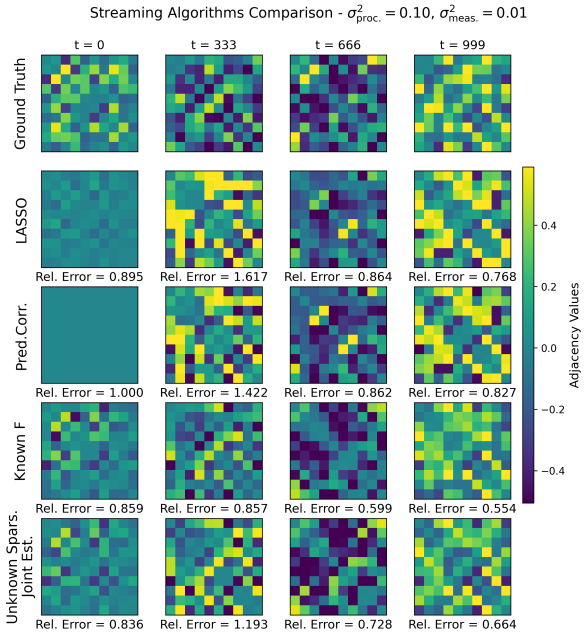


Figure 6.5: Block Coupling agressive scenario. $\sigma^2_{\text{proc.}} = 0.10$, $\sigma^2_{\text{meas.}} = 0.01$. All methods degrade producing a moderate blurring effect with KalDO retaining a slight advantage.

6.5 Conclusion

Three streaming approaches for GTI are compared under a unified experimetaion protocal. We consier LASSO with temporal consistency, PC, and the proposed Kalman-based online GTI (KalDO). Various state dynamic regimes were considered with noise driven dynamics, structured latent couping of increasing strength and severe noise conditions. We note that coupling degrades performance for all methods indicating that the similar observations in chapter 4 and chapter 5 are consistent across methods. Secondly, under heavy measurement noise, the competitor algorithms greatly struggle while KalDO remains providing reasonable estimates by explicitly incorporating $\mathbf{R}$ into the update

equations. This effect may also be partially attributed, however, to knowing the true $\mathbf{R}$.

KalDO is competitive and shows a clear advnatgage when measurement noise is large. It struggles in regimes with weak dynamics as the state transition matrix estimation likely picks up noisy off diagonal contributions where the other methods produce more of a smoothing effect using recent data to mitigate this effect.

Summary of Key Conclusions

- Across all regimes, KalDO is competitive with streaming LASSO and Prediction–Correction, and shows a clear advantage under high measurement noise due to explicit use of $\mathbf{R}$ in its gain.
- In weakly dynamic settings (Random Walk), competitor methods (LASSO, PC) perform better, as KalDO's joint estimation introduces errors in state dynamic estimation.
- As latent coupling increases, conditioning of the observability matrix worsens and errors rise across methods. KalDO benefits modestly from recursive covariance propagation now producing a relative error on par with comparator methods.
- Under heavy process noise, KalDO achieves slightly lower error than competitors by modeling $\mathbf{Q}$, but no method fully compensates for rapid adjacency drift.
- In highly coupled regimes, all methods begin to produce blurred estimates dominated by an increased amount entries tending towards the average value. KalDO performs comparably to competitors but cannot restore sharp contrasts.

7 Conclusion

Performing online and dynamic graph topology inference (GTI) using time-varying graph signals is a challenging endeavor. The solution developed here addressed this problem by modeling the adjacency matrix of a graph as a hidden state within a linear state–space framework, applying recursive Kalman filtering for online estimation, and extending this algorithm to a joint estimation algorithm where the adjacency was inferred from estimated state dynamics. The extension to the joint estimation algorithm is to accommodate more realistic scenarios where the graph state space model would not fully be known. The KalDO GTI algorithm enables the real-time tracking of evolving graph structures and provides performance results that are on par with or better than those of existing algorithms.

7.1 Main Contributions

The *main contributions* of this work are:

- Formalization of the online GTI problem using a graph State Space Model in both static and dynamic settings, including reduced-state formulations that exploit graph symmetry and zero-diagonal constraints to improve identifiability and computational efficiency.
- Development of a Kalman filter-based algorithm for static GTI, tested under Structural Equation, Diffusion, and Input–Output models. Results showed strong recovery for Diffusion and Input–Output, while SEM was demonstrated to be unidentifiable.
- Extension of the Kalman filter framework to dynamic GTI under known state dynamics. Experiments showed reliable online tracking under Random Walk, Block Coupling, and Decay Coupling scenarios, while also identifying the blurring effect caused by strong latent co-evolution.
- Introduction of a joint estimation framework for simultaneous recovery of the adjacency and the unknown state transition matrix. The algorithm integrates structural priors (identity, sparsity, mask), Adam-based adaptive learning, and retroactive smoothing. It was evaluated under known, unknown, and learned sparsity masks under non-identity state dynamics.
- Observability and identifiability analysis in the reduced state space, linking buffer size, conditioning, and mask information to practical recoverability of the state and transition dynamics.

- Benchmarking against streaming LASSO and prediction-correction algorithms in Chapter 6 showed that the KalDO framework is competitive across a variety of regimes. In particular, KalDO is robust under heavy measurement noise due to explicit use of $\mathbf{R}$, achieves modest gains under high process noise by modeling $\mathbf{Q}$, but struggles in weakly dynamic regimes where estimating $\mathbf{F}$ introduces noise into the updates.

Across the static, dynamic, and joint settings, the Kalman-based GTI framework consistently enabled accurate online topology inference under moderate dynamics and sparse interactions. In static graphs, recovery was strongest for the IO model, moderate for diffusion, and unattainable for SEM due to structural non-identifiability. In dynamic graphs with known state dynamics, tracking was accurate when edges evolved independently or weakly coupled, but performance degraded as latent coupling increased, consistent with the conditioning of the observability matrix. In the joint setting, ridge regression was insufficient as was assuming identity state dynamics, but the proposed KalDO GTI algorithm with structural priors and adaptive updates achieved stable estimation of both adjacency and transition dynamics. Known masks yielded performance close to the ground-truth baseline, unknown masks remained effective with longer horizons, and learned masks offered an intermediate solution that improved early-horizon accuracy. The benchmarking experiments of Chapter 6 further reinforced these observations that KalDO remained stable where competitors diverged under large measurement noise, matched performance under strong coupling, and underperformed only in regimes with weak dynamics.

While a head-to-head benchmark is left for future work, the proposed method directly addresses key limitations of comparable approaches (Chapter 2). Relative to windowed batch estimators (e.g., TV-GLASSO), it maintains true streaming updates without sacrificing temporal resolution to long windows. Compared to online prediction-correction schemes, it avoids solving a convex subproblem at every step, yielding lower per-step complexity with short buffers. In contrast to batch LG-SSM methods such as GraphEM and DGLASSO, it does not require full-sequence smoothing, scaling more naturally to long horizons and real-time use. Finally, by operating in reduced coordinates and enforcing structural priors with retroactive smoothing, it preserves interpretability (explicit state/transition matrices) while improving conditioning and identifiability in practical buffer sizes.

7.2 Future Work

This thesis has demonstrated that Kalman filtering provides a principled, efficient, and flexible basis for online dynamic GTI. By unifying static, dynamic, and joint estimation within a state–space framework, it offers a step toward real-time recovery of evolving network structures. The methods and analyses presented here provide both practical algorithms and theoretical insights, laying the groundwork for further advances in dynamic network inference using a Kalman and graph state-space model approach.

Several directions remain open, motivated both by the limitations observed and by opportunities from systems theory and linear algebra:

- **Dense state dynamics.** A fundamental limitation is the decrease in performance as the state transition matrix becomes less sparse, even when $\mathbf{F}$ is known.
- **Diagnostics for excessive coupling.** Performance degrades smoothly with increasing latent co-evolution, but there may be thresholds beyond which tracking is unreliable. Future work could design a practical diagnostic test, based on condition numbers of Φ or $\mathcal{O}$ or on innovation statistics, to assess in advance whether a real dataset is suitable for this algorithm.
- **Temporally varying dynamics.** The current joint algorithm assumes fixed $\mathbf{F}$. Extending it to handle time-varying transition matrices $\mathbf{F}_t$ would allow modeling of regime shifts, smoothly changing dynamics, or state-dependent couplings.
- **Alternative structural priors.** Beyond identity and sparsity, new priors could be incorporated, including entropy-based constraints, spectral smoothness, or modularity.
- **Explicit state sparsity.** No explicit sparsity prior was imposed on the hidden state $\mathbf{u}_t$. Future extensions could incorporate ℓ_1 penalties, thresholding, or Bussgang-type nonlinearities in the Kalman update to enforce sparse adjacency trajectories.
- **Applications to real-world data.** All experiments were synthetic. Applying the framework to domains such as financial markets or brain connectivity would test robustness against missing data, nonlinearities, and non-Gaussian noise, while validating the proposed identifiability and observability diagnostics in practice.
- **Efficiency versus robustness trade-off.** Chapter 6 highlighted that KalDO's advantage is most pronounced under heavy noise, whereas simpler methods outperform it when dynamics are weak. Future work could develop hybrid schemes that switch between lightweight estimators and Kalman-based updates depending on detected noise levels or coupling strength, reducing unnecessary computational overhead.

Kalman Filter to Batch Least Squares Under Static Conditions

A

A.1 Derivation of the Batch Weighted Least Squares Estimator

The batch weighted least squares problem is presented with measurement noise covariance $\mathbf{R} \succ 0$. The goal is to solve the following equation,

$$\hat{\mathbf{a}}_{\text{LS}} = \arg\min_{\mathbf{a}} \sum_{t=1}^{T} \|\mathbf{y}_t - \mathbf{H}_t\mathbf{a}\|^2_{\mathbf{R}^{-1}} = \arg\min_{\mathbf{a}} \sum_{t=1}^{T} (\mathbf{y}_t - \mathbf{H}_t\mathbf{a})^\top \mathbf{R}^{-1} (\mathbf{y}_t - \mathbf{H}_t\mathbf{a}) \tag{A.1}$$

Stacking the observations,

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_T \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \\ \vdots \\ \mathbf{H}_T \end{bmatrix}, \quad \mathbf{R}_{\text{block}} = \text{blkdiag}(\mathbf{R}, \ldots, \mathbf{R}) \in \mathbb{R}^{TN \times TN} \tag{A.2}$$

The full cost equation is now,

$$\hat{\mathbf{a}} = \arg\min_{\mathbf{a}} (\mathbf{y} - \mathbf{H}\mathbf{a})^\top \mathbf{R}^{-1}_{\text{block}} (\mathbf{y} - \mathbf{H}\mathbf{a}) \tag{A.3}$$

Next, the derivative is taken,

$$\frac{\partial}{\partial \mathbf{a}} \left[(\mathbf{y} - \mathbf{H}\mathbf{a})^\top \mathbf{R}^{-1}_{\text{block}} (\mathbf{y} - \mathbf{H}\mathbf{a}) \right] = -2\mathbf{H}^\top \mathbf{R}^{-1}_{\text{block}} \mathbf{y} + 2\mathbf{H}^\top \mathbf{R}^{-1}_{\text{block}} \mathbf{H}\mathbf{a} \tag{A.4}$$

Which is then set to zero and solved,

$$\mathbf{H}^\top \mathbf{R}^{-1}_{\text{block}} \mathbf{H}\hat{\mathbf{a}} = \mathbf{H}^\top \mathbf{R}^{-1}_{\text{block}} \mathbf{y} \Rightarrow \hat{\mathbf{a}} = \left(\mathbf{H}^\top \mathbf{R}^{-1}_{\text{block}} \mathbf{H} \right)^{-1} \left(\mathbf{H}^\top \mathbf{R}^{-1}_{\text{block}} \mathbf{y} \right) \tag{A.5}$$

Equivalently, by summing over individual time steps,

$$\boxed{\hat{\mathbf{a}}_{\text{LS}} = \left(\sum_{t=1}^{T} \mathbf{H}_t^\top \mathbf{R}^{-1} \mathbf{H}_t \right)^{-1} \left(\sum_{t=1}^{T} \mathbf{H}_t^\top \mathbf{R}^{-1} \mathbf{y}_t \right)} \tag{A.6}$$

This is the weighted LS solution under independent, identically distributed Gaussian noise $\boldsymbol{\nu}_t \sim \mathcal{N}(0, \mathbf{R})$.

A.2 Kalman Filter: Standard Form

Prediction Step

$$\begin{aligned}
\hat{\mathbf{a}}_t^- &= \mathbf{F}_{t-1}\hat{\mathbf{a}}_{t-1}^+ && \text{(prior mean)}\\
\boldsymbol{\Sigma}_t^- &= \mathbf{F}_{t-1}\boldsymbol{\Sigma}_{t-1}^+\mathbf{F}_{t-1}^\top + \mathbf{Q}_{t-1} && \text{(prior covariance)}
\end{aligned}$$

Update Step

$$\begin{aligned}
\mathbf{r}_t &= \mathbf{y}_t - \mathbf{H}_t\hat{\mathbf{a}}_t^- && \text{(innovation)}\\
\mathbf{S}_t &= \mathbf{H}_t\boldsymbol{\Sigma}_t^-\mathbf{H}_t^\top + \mathbf{R}_t && \text{(innovation covariance)}\\
\mathbf{K}_t &= \boldsymbol{\Sigma}_t^-\mathbf{H}_t^\top\mathbf{S}_t^{-1} && \text{(Kalman gain)}\\
\hat{\mathbf{a}}_t^+ &= \hat{\mathbf{a}}_t^- + \mathbf{K}_t\mathbf{r}_t && \text{(posterior mean)}\\
\boldsymbol{\Sigma}_t^+ &= (\mathbf{I} - \mathbf{K}_t\mathbf{H}_t)\,\boldsymbol{\Sigma}_t^-\,(\mathbf{I} - \mathbf{K}_t\mathbf{H}_t)^\top + \mathbf{K}_t\mathbf{R}_t\mathbf{K}_t^\top && \text{(posterior covariance)}
\end{aligned}$$

The final update for the posterior covariance uses the Joseph form [35], which is known to improve numerical stability. It is commonly used in long-running filters or when working with nearly singular matrices. The Joseph update guarantees that the resulting covariance matrix remains symmetric and positive semi-definite, even in the presence of rounding errors. Although this version is slightly more computationally expensive due to additional matrix multiplications, it is preferred in this GTI application as it delivers increased robustness.

A.3 Kalman Filter in Recursive Information Form

We define information form first by the following information matrix and information vector.

$$\mathbf{S}_t := (\boldsymbol{\Sigma}_t^+)^{-1}, \quad \mathbf{h}_t := \mathbf{S}_t\hat{\mathbf{a}}_t^+ \tag{A.7}$$

A.3.1 Step 1: Recursive Update of the Information Matrix

In the standard Kalman update, we have the posterior covariance being updated as,

$$\boldsymbol{\Sigma}_t^+ = \left(\boldsymbol{\Sigma}_t^- - \mathbf{K}_t\mathbf{H}_t\boldsymbol{\Sigma}_t^-\right) \tag{A.8}$$

In the information form, this becomes,

$$\mathbf{S}_t = (\boldsymbol{\Sigma}_t^+)^{-1} = \left(\boldsymbol{\Sigma}_t^- - \boldsymbol{\Sigma}_t^-\mathbf{H}_t^\top\left(\mathbf{H}_t\boldsymbol{\Sigma}_t^-\mathbf{H}_t^\top + \mathbf{R}_t\right)^{-1}\mathbf{H}_t\boldsymbol{\Sigma}_t^-\right)^{-1} \tag{A.9}$$

Now, we must apply the matrix inversion lemma. The matrix inversion lemma states:

$$(\mathbf{A} + \mathbf{UCV})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}\left(\mathbf{C}^{-1} + \mathbf{VA}^{-1}\mathbf{U}\right)^{-1}\mathbf{VA}^{-1} \tag{A.10}$$

Applying this gives,

$$(\boldsymbol{\Sigma}_t^+)^{-1} = (\boldsymbol{\Sigma}_t^-)^{-1} + \mathbf{H}_t^\top\mathbf{R}_t^{-1}\mathbf{H}_t \tag{A.11}$$

Thus, the recursive update of the information matrix is,

$$\boxed{\mathbf{S}_t = \mathbf{S}_{t-1} + \mathbf{H}_t^\top \mathbf{R}_t^{-1} \mathbf{H}_t}$$

The recursive update for the information matrix has now been derived.

A.3.2 Step 2: Recursive Update of the Information Vector

Starting with the Kalman update of the hidden state,

$$\hat{\mathbf{a}}_t^+ = \hat{\mathbf{a}}_t^- + \mathbf{K}_t(\mathbf{y}_t - \mathbf{H}_t \hat{\mathbf{a}}_t^-)$$

Multiply both sides by $\mathbf{S}_t$:

$$\mathbf{h}_t = \mathbf{S}_t \hat{\mathbf{a}}_t^+ = \mathbf{S}_t \hat{\mathbf{a}}_t^- + \mathbf{S}_t \mathbf{K}_t (\mathbf{y}_t - \mathbf{H}_t \hat{\mathbf{a}}_t^-)$$

We now want to simplify the quantity, $\mathbf{S}_t \mathbf{K}_t$
Starting with the Kalman gain,

$$\mathbf{K}_t = \boldsymbol{\Sigma}_t^- \mathbf{H}_t^\top \left(\mathbf{H}_t \boldsymbol{\Sigma}_t^- \mathbf{H}_t^\top + \mathbf{R}_t \right)^{-1}$$

and using the identity $\mathbf{S}_t = (\boldsymbol{\Sigma}_t^+)^{-1} = (\boldsymbol{\Sigma}_t^-)^{-1} + \mathbf{H}_t^\top \mathbf{R}_t^{-1} \mathbf{H}_t$, we can define,

$$\mathbf{S}_t \mathbf{K}_t = \left[(\boldsymbol{\Sigma}_t^-)^{-1} + \mathbf{H}_t^\top \mathbf{R}_t^{-1} \mathbf{H}_t \right] \boldsymbol{\Sigma}_t^- \mathbf{H}_t^\top \left(\mathbf{H}_t \boldsymbol{\Sigma}_t^- \mathbf{H}_t^\top + \mathbf{R}_t \right)^{-1}$$

Next, we distribute,

$$\mathbf{S}_t \mathbf{K}_t = \left[\mathbf{I} + \mathbf{H}_t^\top \mathbf{R}_t^{-1} \mathbf{H}_t \boldsymbol{\Sigma}_t^- \right] \mathbf{H}_t^\top \left(\mathbf{H}_t \boldsymbol{\Sigma}_t^- \mathbf{H}_t^\top + \mathbf{R}_t \right)^{-1}$$

Now, we can use the identity,

$$\left(\mathbf{I} + \mathbf{R}_t^{-1} \mathbf{H}_t \boldsymbol{\Sigma}_t^- \mathbf{H}_t^\top \right) \left(\mathbf{H}_t \boldsymbol{\Sigma}_t^- \mathbf{H}_t^\top + \mathbf{R}_t \right)^{-1} = \mathbf{R}_t^{-1}$$

Therefore, we simplify to,

$$\boxed{\mathbf{S}_t \mathbf{K}_t = \mathbf{H}_t^\top \mathbf{R}_t^{-1}}$$

Recall also that $\mathbf{h}_{t-1} = \mathbf{S}_{t-1} \hat{\mathbf{a}}_{t-1}^+ = \mathbf{S}_{t-1} \hat{\mathbf{a}}_t^-$.
Now the information vector can now be written as,

$$\mathbf{h}_t = \mathbf{h}_{t-1} + \mathbf{H}_t^\top \mathbf{R}_t^{-1} \mathbf{y}_t$$

Hence, the recursive update of the information vector is:

$$\boxed{\mathbf{h}_t = \mathbf{h}_{t-1} + \mathbf{H}_t^\top \mathbf{R}_t^{-1} \mathbf{y}_t}$$

The posterior mean estimate, $\mathbf{a}_t^+$ is recovered as,

$$\boxed{\hat{\mathbf{a}}_t^+ = \mathbf{S}_t^{-1} \mathbf{h}_t}$$

Thus, the Kalman Filter in Recursive information form under static conditions and without process noise ($\mathbf{Q}_t = 0$), is,

$$\boxed{\begin{aligned} \mathbf{S}_t &= \mathbf{S}_{t-1} + \mathbf{H}_t^\top \mathbf{R}_t^{-1} \mathbf{H}_t \\ \mathbf{h}_t &= \mathbf{h}_{t-1} + \mathbf{H}_t^\top \mathbf{R}_t^{-1} \mathbf{y}_t \\ \hat{\mathbf{a}}_t^+ &= \mathbf{S}_t^{-1} \mathbf{h}_t \end{aligned}}$$

A.3.3 Equivalence to Batch LS under Static Conditions

When the following assumptions hold (1) The graph state is static $\mathbf{a}_t = \mathbf{a}$, (2) The process noise is zero: $\mathbf{Q} = \mathbf{0}$, and (3) The prior is uninformative: $\boldsymbol{\Sigma}_0^{-1} \to \mathbf{0}$ certain characteristics emerge.

The recursive information form accumulates over time as,

$$\mathbf{S}_T = \sum_{t=1}^{T} \mathbf{H}_t^\top \mathbf{R}_t^{-1} \mathbf{H}_t$$
$$\mathbf{h}_T = \sum_{t=1}^{T} \mathbf{H}_t^\top \mathbf{R}_t^{-1} \mathbf{y}_t$$

and the posterior mean estimate becomes,

$$\hat{\mathbf{a}}_T = \mathbf{S}_T^{-1} \mathbf{h}_T = \left(\sum_{t=1}^{T} \mathbf{H}_t^\top \mathbf{R}_t^{-1} \mathbf{H}_t \right)^{-1} \left(\sum_{t=1}^{T} \mathbf{H}_t^\top \mathbf{R}_t^{-1} \mathbf{y}_t \right)$$

We can compare to the batched least squares solution,

$$\hat{\mathbf{a}}_{\text{LS}} = \left(\sum_{t=1}^{T} \mathbf{H}_t^\top \mathbf{R}^{-1} \mathbf{H}_t \right)^{-1} \left(\sum_{t=1}^{T} \mathbf{H}_t^\top \mathbf{R}^{-1} \mathbf{y}_t \right)$$

and notice that they exactly match. The Kalman filter reduces to the batch LS estimate under static dynamics, uninformative prior, and zero process noise.

$$\boxed{\hat{\mathbf{a}}_T^{\text{KF}} = \hat{\mathbf{a}}_{\text{LS}}}$$

Selection and Expansion Matrices for Reduced State Representation

B

This appendix details the explicit construction of the selection and expansion matrices first referred to in section 3.7 but used consistently throughout chapter 4 and chapter 5. These matrices allow the adjacency matrix $\mathbf{A}_t$ to be represented in reduced coordinates while enforcing structural constraints such as zero diagonals and symmetry.

B.0.1 Zero-Diagonal Reduction

When only the zero-diagonal constraint is imposed, the reduced hidden state $\mathbf{u}_t \in \mathbb{R}^{N^2-N}$ is obtained by removing all diagonal entries of $\mathbf{A}_t$. The downsizing matrix $\mathbf{D} \in \mathbb{R}^{(N^2-N)\times N^2}$ selects all off-diagonal entries as, $\mathbf{u}_t = \mathbf{D}\,\mathrm{vec}(\mathbf{A}_t)$ built with the algorithm in Algorithm 8.

Algorithm 8 Build $\mathbf{D}$ Removing Zero Diagonal, given N

Input: N (number of nodes), $row \leftarrow 1$
Initialize $\mathbf{D} \leftarrow \mathbf{0}_{(N^2-N)\times N^2}$
for $j = 1$ to N **do**
    for $i = 1$ to N **do**
        if $i \neq j$ **then**
            $col \leftarrow (j-1)N + i$
            $\mathbf{D}_{row,col} \leftarrow 1$
            $row \leftarrow row + 1$
        end if
    end for
end for
Output: D

We can consider for example, $N = 3$

$$\underbrace{\begin{bmatrix} A_{21} \\ A_{31} \\ A_{12} \\ A_{32} \\ A_{13} \\ A_{23} \end{bmatrix}}_{\mathbf{u}} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}}_{\mathbf{D}} \underbrace{\begin{bmatrix} A_{11} \\ A_{21} \\ A_{31} \\ A_{12} \\ A_{22} \\ A_{32} \\ A_{13} \\ A_{23} \\ A_{33} \end{bmatrix}}_{\mathbf{a}}. \tag{B.1}$$

B.0.2 Zero-Diagonal and Symmetry Reduction

When both zero-diagonal and symmetry constraints are enforced, the reduced state $\mathbf{u}_t \in \mathbb{R}^{N(N-1)/2}$ collects the strictly upper-triangular entries of $\mathbf{A}_t$. The downsizing matrix $\mathbf{D} \in \mathbb{R}^{\frac{N(N-1)}{2} \times N^2}$ extracts these entries, while the expansion matrix $\mathbf{E} \in \mathbb{R}^{N^2 \times \frac{N(N-1)}{2}}$ reconstructs $\mathbf{a}_t = \mathrm{vec}(\mathbf{A}_t)$ by duplicating entries across symmetric positions and inserting zeros along the diagonal as described in Algorithm 9.

Algorithm 9 Build $\mathbf{D}$ (strictly upper triangle), given N

Input: N (number of nodes), $row \leftarrow 1$
Initialize $\mathbf{D} \leftarrow \mathbf{0}_{\frac{N(N-1)}{2} \times N^2}$
for $j = 2$ to N **do**
    for $i = 1$ to $j - 1$ **do**
        $col \leftarrow (j-1)N + i$
        $\mathbf{D}_{row,col} \leftarrow 1$
        $row \leftarrow row + 1$
    end for
end for
Output: D

Practically, for $N = 3$, applying Algorithm 9 yields,

$$\underbrace{\begin{bmatrix} A_{12} \\ A_{13} \\ A_{23} \end{bmatrix}}_{\mathbf{u}} = \underbrace{\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}}_{\mathbf{D}} \underbrace{\begin{bmatrix} A_{11} \\ A_{21} \\ A_{31} \\ A_{12} \\ A_{22} \\ A_{32} \\ A_{13} \\ A_{23} \\ A_{33} \end{bmatrix}}_{\mathbf{a}}. \tag{B.2}$$

An example for a 3x3 would be,

$$\underbrace{\begin{bmatrix} 0 \\ A_{21} \\ A_{31} \\ A_{12} \\ 0 \\ A_{32} \\ A_{13} \\ A_{23} \\ 0 \end{bmatrix}}_{\mathbf{a}} = \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}}_{\mathbf{E}} \underbrace{\begin{bmatrix} A_{12} \\ A_{13} \\ A_{23} \end{bmatrix}}_{\mathbf{u}}. \tag{B.3}$$

Algorithm 10 Build $\mathbf{E}$ (symmetric expansion), given N

Input: N (number of nodes), $row \leftarrow 1$
Initialize $\mathbf{E} \leftarrow \mathbf{0}_{N^2 \times \frac{N(N-1)}{2}}$
for $j = 2$ to N **do**
    for $i = 1$ to $j - 1$ **do**
        $col_1 \leftarrow (j-1)N + i$
        $col_2 \leftarrow (i-1)N + j$
        $\mathbf{E}_{col_1,row} \leftarrow 1, \;\; \mathbf{E}_{col_2,row} \leftarrow 1$
        $row \leftarrow row + 1$
    end for
end for
Output: $\mathbf{E}$

The downsizing matrix $\mathbf{D}$ removes redundant or constrained entries from the adjacency, while the expansion matrix $\mathbf{E}$ restores the full vectorized form by enforcing the required structure. The reduced state $\mathbf{u}_t$ is therefore the minimal representation of $\mathbf{A}_t$, with dimension d_{red} depending on which constraints are assumed. All algorithms in Chapters 4 and 5 are carried out in these reduced coordinates.

C Prior Matrix Explained

This appendix explains why we add the prior matrix $\mathbf{\Lambda}$ and how it improves identifiability. The idea comes from estimation theory in that the Fisher Information Matrix (FIM) measures how well parameters can be inferred from data [36]. If the FIM has very small eigenvalues, some parameter directions are poorly constrained, making estimation unstable and noise-sensitive. Priors act as an additional source of information. By adding them, we increase the curvature of the loss function and enlarge the smallest eigenvalues of the information matrix. This improves conditioning and stabilizes estimation [37, 34, 38, 39].

C.1 Background Theory

In the vectorized measurement model, the outputs stacked over L time steps can be written as,

$$\mathbf{Y} \approx \mathbf{\Phi}\,\mathrm{vec}(\mathbf{F}), \tag{C.1}$$

where $\mathbf{\Phi}$ collects the compressed states and observation matrices. It is known that the process and measurement noise in our graph state–space model are assumed Gaussian. Under Gaussian noise, the negative log-likelihood of the data is quadratic in the parameters, and the Fisher Information Matrix reduces to $\mathbf{\Phi}^\top\mathbf{\Phi}$ [36]. For Gaussian noise, the Fisher Information Matrix is proportional to,

$$\mathbf{\Phi}^\top\mathbf{\Phi}. \tag{C.2}$$

This matrix determines the curvature of the likelihood in the space of $\mathrm{vec}(\mathbf{F})$. If it has near-zero eigenvalues, some directions in $\mathbf{F}$ cannot be distinguished from the data, and the estimation problem becomes ill-posed.

The modified information matrix incorporates prior information as,

$$\mathcal{I}_{\text{window}} = \mathbf{\Phi}^\top\mathbf{\Phi} + \mathbf{\Lambda}. \tag{C.3}$$

Here $\mathbf{\Lambda}$ is diagonal with nonnegative entries. This form arises because the priors are separable across entries of $\mathbf{F}$, so their Hessian contribution acts independently on each parameter. This is the same principle as ridge regression, where a quadratic penalty on coefficients adds λI, improving numerical stability by lifting small eigenvalues [37, 34].

C.2 Setup and Priors

Let $\mathbf{F} \in \mathbb{R}^{d_{\text{red}}\times d_{\text{red}}}$ and $\mathrm{vec}(\mathbf{F}) \in \mathbb{R}^{d_{\text{red}}^2}$. The prior part of the loss is

$$\lambda\sum_{i=1}^{d_{\text{red}}}(F_{ii}-1)^2 + \gamma\sum_{i\neq j}(1-M_{ij})F_{ij}^2 + \eta\|\mathbf{F}\|_F^2. \tag{C.4}$$

Only the quadratic terms contribute to conditioning because they affect the Hessian (curvature, quantity of interest for the FIM).

C.3 Example

For

$$\mathbf{F} = \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix}, \qquad \mathrm{vec}(\mathbf{F}) = \begin{bmatrix} F_{11} \\ F_{21} \\ F_{12} \\ F_{22} \end{bmatrix}, \tag{C.5}$$

the quadratic penalties are

$$\lambda(F_{11}^2 + F_{22}^2) + \gamma\big((1 - M_{21})F_{21}^2 + (1 - M_{12})F_{12}^2\big) + \eta(F_{11}^2 + F_{21}^2 + F_{12}^2 + F_{22}^2). \tag{C.6}$$

Therefore

$$\mathbf{\Lambda} = \mathrm{diag}\Big(\lambda + \eta,\ \gamma(1 - M_{21}) + \eta,\ \gamma(1 - M_{12}) + \eta,\ \lambda + \eta\Big). \tag{C.7}$$

So diagonal entries of $\mathbf{F}$ get weight $\lambda + \eta$, masked off-diagonals get $\gamma + \eta$, and allowed off-diagonals ($M_{ij} = 1$) get only η. This shows explicitly how priors map to diagonal entries of $\mathbf{\Lambda}$.

C.4 Interpretation

The matrix $\mathbf{\Lambda}$ is simply the matrix form of the quadratic prior penalties. Adding it increases the smallest eigenvalue of the information matrix, which improves conditioning and reduces sensitivity to noise. In practice this means that parameter directions which were previously ambiguous become better identified. This ensures more stable and interpretable estimates of the transition matrix $\mathbf{F}$ [36, 37, 34, 38, 39].

Bibliography

[1] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[2] A. Ortega, P. Frossard, J. Kovačević, J. M. F. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 808–828, 2018.

[3] M. E. J. Newman, *Networks: An Introduction*. Oxford University Press, 2018.

[4] X. Dong, D. Thanou, M. Rabbat, and P. Frossard, "Learning graphs from data: A signal representation perspective," *IEEE Signal Processing Magazine*, vol. 36, no. 3, pp. 44–63, 2019.

[5] G. Mateos, S. Segarra, A. G. Marques, and A. Ribeiro, "Connecting the dots: Identifying network structure via graph signal processing," *IEEE Signal Processing Magazine*, vol. 36, no. 3, pp. 16–43, 2019.

[6] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, "Learning laplacian matrix in smooth graph signal representations," *IEEE Transactions on Signal Processing*, vol. 64, no. 23, pp. 6160–6173, 2016.

[7] V. Kalofolias, "How to learn a graph from smooth signals," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR, 2016, pp. 920–929.

[8] J. Friedman, T. Hastie, and R. Tibshirani, "Sparse inverse covariance estimation with the graphical lasso," *Biostatistics*, vol. 9, no. 3, pp. 432–441, 12 2007. [Online]. Available: https://doi.org/10.1093/biostatistics/kxm045

[9] D. Hallac, Y. Park, S. Boyd, and J. Leskovec, "Network inference via the time-varying graphical lasso," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 205–213. [Online]. Available: https://doi.org/10.1145/3097983.3098037

[10] G. B. Giannakis, Y. Shen, and G. V. Karanikolas, "Topology identification and learning over graphs: Accounting for nonlinearities and dynamics," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 787–807, 2018.

[11] K. Yamada, Y. Tanaka, and A. Ortega, "Time-varying graph learning based on sparseness of temporal variation," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5411–5415.

[12] A. Natali, M. Coutino, E. Isufi, and G. Leus, "Online time-varying topology identification via prediction-correction algorithms," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 5400–5404.

[13] D. Zambon and C. Alippi, "Graph kalman filters," in *Temporal Graph Learning Workshop @ NeurIPS 2023*, 2023. [Online]. Available: https://openreview.net/forum?id=KGqtCfYJon

[14] L. Dabush, N. Shlezinger, and T. Routtenberg, "Sparsity-aware extended kalman filter for tracking dynamic graphs," 2025. [Online]. Available: https://arxiv.org/abs/2507.09999

[15] N. Biggs, Lloyd, and R. J. Wilson, *Graph theory, 1736-1936*. Clarendon Press, Feb. 1986.

[16] V. Kalofolias, A. Loukas, D. Thanou, and P. Frossard, "Learning time varying graphs," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 2826–2830.

[17] A. Natali, E. Isufi, M. Coutino, and G. Leus, "Learning time-varying graphs from online data," *IEEE Open Journal of Signal Processing*, vol. 3, pp. 212–228, 2022.

[18] T. Dabush and T. Routtenberg, "A kalman filter for tracking network dynamics," *arXiv preprint arXiv:2401.08451*, 2024.

[19] V. Elvira and E. Chouzenoux, "Graphical inference in linear-gaussian state-space models," *IEEE Transactions on Signal Processing*, vol. 70, p. 4757–4771, 2022. [Online]. Available: http://dx.doi.org/10.1109/TSP.2022.3209016

[20] E. Chouzenoux and V. Elvira, "Sparse graphical linear dynamical systems," 2024. [Online]. Available: https://arxiv.org/abs/2307.03210

[21] R. E. Kalman and Others, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[22] B. D. O. Anderson and J. B. Moore, *Optimal filtering*. Prentice Hall, 1979.

[23] P. S. Maybeck, *Stochastic models, estimation, and control*. Academic Press, 1979, vol. 1.

[24] R. Shafipour, S. Segarra, A. G. Marques, and G. Mateos, "Directed network topology inference via graph filter identification," in *2018 IEEE Data Science Workshop (DSW)*, 2018, pp. 210–214.

[25] Y. Han, "Graph topology identification using structural equation models," Master's thesis, Delft University of Technology, 2024, mSc Thesis.

[26] L. Ljung, *System identification (2nd ed.): theory for the user*. USA: Prentice Hall PTR, 1999.

[27] Q. Zhang and L. Zhang, "Stability Analysis of the Kalman Predictor," *International Journal of Control*, pp. 1–16, Jul. 2019. [Online]. Available: https://inria.hal.science/hal-02373906

[28] K. P. Murphy, *Machine learning : a probabilistic perspective*, ser. Adaptive computation and machine learning series. Cambridge, MA: MIT, 2012. [Online]. Available: https://www.worldcat.org/title/machine-learning-a-probabilistic-perspective/oclc/781277861?referer=br&ht=edition

[29] P. Contributors, "Adam." [Online]. Available: https://docs.pytorch.org/docs/stable/generated/torch.optim.Adam.html

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017. [Online]. Available: https://arxiv.org/abs/1412.6980

[31] A. Natali, "Signal processing and optimization on graphs: Learning time-varying structures and generalizing convolution principles," Dissertation (TU Delft), Delft University of Technology, 2024.

[32] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Sparse identification of nonlinear dynamics with control (sindyc)," 2016. [Online]. Available: https://arxiv.org/abs/1605.06682

[33] SKLearn. [Online]. Available: https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html

[34] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996. [Online]. Available: http://www.jstor.org/stable/2346178

[35] R. Zanetti and K. J. DeMars, "Joseph formulation of unscented and quadrature filters with application to consider states," *Journal of Guidance, Control, and Dynamics*, vol. 36, no. 6, pp. 1860–1864, 2013.

[36] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*. Upper Saddle River, NJ: Prentice Hall, 1993.

[37] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[38] K. D. Ross, *Statistical Notes: Lecture Handouts*, 2020, available online. [Online]. Available: https://bookdown.org/kevin_davisross/stat415-handouts/normal-normal.html

[39] "Adding a small constant to the diagonals of a matrix to stabilize," CrossValidated (StackExchange), 2019. [Online]. Available: https://stats.stackexchange.com/questions/390532/adding-a-small-constant-to-the-diagonals-of-a-matrix-to-stabilize