Learning Markov Processes as Sum-of-Square Forms for Analytical Belief Propagation

Peter Amorese
University of Colorado Boulder

Morteza Lahijanian
University of Colorado Boulder

Abstract

Harnessing the predictive capability of Markov process models requires propagating probability density functions (beliefs) through the model. For many existing models however, belief propagation is analytically infeasible, requiring approximation or sampling to generate predictions. This paper proposes a functional modeling framework leveraging sparse Sum-of-Squares (SoS) forms for valid (conditional) density estimation. We study the theoretical restrictions of modeling conditional densities using the SoS form, and propose a novel functional form for addressing such limitations. The proposed architecture enables generalized simultaneous learning of basis functions and coefficients, while preserving analytical belief propagation. In addition, we propose a training method that allows for exact adherence to the normalization and non-negativity constraints. Our results show that the proposed method achieves accuracy comparable to state-of-the-art approaches while requiring significantly less memory in low-dimensional spaces, and it further scales to 12D systems when existing methods fail beyond 2D.

1 INTRODUCTION

Machine learning models offer a flexible and useful means of predicting real world processes. Due to either uncertainty, or stochastic environmental disturbances, such processes are often stochastic in nature. Models that account for this stochasticity, can significantly improve the quality and robustness of predictions, making them more reliable. However, propagating uncertainty in the state of the process through the model, i.e., *belief propagation*, is often analytically infeasible for continuous state-spaces, thus requiring sampling or approximations. Even with the most accurate models, the need for approximation can cause prediction quality and interpretability to suffer. This work studies a novel functional form for learning general stochastic processes, while permitting analytical belief propagation.

Belief propagation is the process of computing the predicted marginal distribution (belief) of the state of the system at a future time. For continuous state systems, a belief belongs to a *function-space* which is generally infinite-dimensional. Certain simple processes, in particular, linear systems with additive Gaussian noise, permit analytical belief propagation. However, such systems are often not adequate for describing many real world processes. Deviation from either the linear assumption or the additive Gaussian noise assumption typically makes belief propagation subject to intractable integrals (Jasour et al., 2021).

The standard means of mitigating this issue is through approximation. A prominent example is non-linear filtering, in which linearization-based methods (Schei, 1997) (such as the Extended Kalman Filter) and second-order approximations have been proposed (Julier and Uhlmann, 2004). Building upon such foundational approaches, Gaussian Mixture Model- (GMM) based approaches account for multi-modal approximation of the belief (Alspach and Sorenson, 2003; Figueiredo et al., 2024; Kulik and LeGrand, 2024). Not only are these methods approximate, they make restrictive assumptions about the underlying Markov process, such as additive Gaussian noise.

In contrast, sampling-based methods, such as particle filters (Arulampalam et al., 2002), can leverage generative models (Jonschkowski et al., 2018) and propagate particles through the model to obtain a particle approximation of the belief. Nonetheless, these methods often require a large number of samples for high-dimensional systems. Furthermore, reconstructing a probability density from a particle set involves post-processing through density estimation techniques, e.g., Kernel Density Estimation (Parzen, 1962). Other approaches, such as moment-based methods (Jasour et al., 2021) can exactly propagate the moments of a belief for a class of dynamical systems; however, moments are generally insufficient for reconstructing the full belief

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

(Akhiezer, 2020).

Amorese and Lahijanian (2026) recently proposed a method of solving the modeling problem for analytical belief propagation. The Markov process is modeled using Bernstein-polynomial Normalizing Flows (BNF), leveraging the flexible analytical characteristics of polynomials to perform belief propagation. The Bernstein polynomial basis is a *partition of unity* (i.e., a set of non-negative functions that sum to one everywhere), a unique and rare characteristic that allows Bernstein Normalizing Flow models to model valid Markov processes. Unfortunately, by definition, the Bernstein basis is *dense*, i.e., the number of required non-zero parameters is exponential in the dimension of the system. Consequently, the time cost of computing future beliefs is also exponential, preventing BNF from scaling beyond two-dimensional systems.

Sum-of-squares (SoS) forms are a powerful functional for modeling non-negativity, and have been used for tractable (conditional) density modeling (Loconte et al., 2025; Rudi and Ciliberto, 2021), and have been shown to be more expressive than mixture models (Marteau-Ferey et al., 2020; Loconte et al., 2023). Modeling Markov processes using the proposed conditional density model (e.g. in Rudi and Ciliberto (2021)) yields a rational function, which does not permit analytical belief propagation.

This work studies a novel functional form for modeling general Markov processes using SoS forms that, by construction, permits analytical belief propagation. We leverage Sum-of-Squares theory to form a valid *conditional* density estimator for which one can train not only the coefficients, but also the *basis functions* themselves. Similar to mixture models, this freedom allows the optimization to intelligently tune and allocate basis functions to achieve a sparse and accurate representation of the underlying process. We study the theoretical implications, limitations, and advantages of the SoS functional form, substantiated through experimental results.

The key contributions of this work are as follows:

- a theoretical analysis of the consequences of using SoS for conditional density estimation, proving an important result about its restrictions under mild assumptions,
- a novel functional form that alleviates the restrictions of SoS, without sacrificing any desireable theoretical attributes,
- a means of enforcing the valid-distribution constraints through direct parameterization, and
- experimental comparisons and demonstrations of the efficacy of the proposed approach for modeling high-dimensional systems.

Notation

We denote the open unit box by $\mathbb{U}^d = (0,1)^d$. Vectors are written in bold, e.g., $\mathbf{x} \in \mathbb{U}^d$, with the i-th element of $\mathbf{x}$ denoted by x_i. The set of all real-valued continuous multivariate functions on $\mathbb{U}^d$ is denoted by $\mathcal{C}(\mathbb{U}^d) \subset \mathcal{C}(\mathbb{R}^d)$. Probability density functions over random vectors $\mathbf{x}$ are written as $p(\mathbf{x})$. A positive semi-definite (resp. positive definite) matrix $\mathbf{A}$ is denoted by $\mathbf{A} \succeq 0$ (resp. $\mathbf{A} \succ 0$). The inner product of two functions f, g is defined as $\langle f, g \rangle = \int_{\mathbb{U}^d} f(\mathbf{x}) g(\mathbf{x}) d\mathbf{x}$. The Hadamard (element-wise) product of two matrices (or vectors) $\mathbf{A}$ and $\mathbf{B}$ is denoted by $\mathbf{A} \odot \mathbf{B}$.

2 PROBLEM FORMULATION

In this work, we consider a general discrete-time stochastic process that evolves in $\mathbb{R}^d$. Our goal is to study the propagation of its Probability Density Function (PDF), also referred to as the *belief*, solely by using data. For simplicity of presentation and without loss of generality, we assume that the state space is transformed to the open unit box $\mathbb{U}^d$.[1]

Let $\mathbf{x}_k \in \mathbb{U}^d$ denote the (transformed) state of the stochastic process at time step $k \in \mathbb{N}^0$, with associated PDF (belief) $p(\mathbf{x}_k)$. Furthermore, let $\vec{\mathbf{x}} = \mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_K$ with $K \in \mathbb{N}^0$ represent a sequence of states generated by the process. We assume that the process is Markov, i.e., the state evolution is conditionally dependent only on the previous state, so that transitions follow the time-invariant distribution $p(\mathbf{x}_k \mid \mathbf{x}_{k-1})$ (also denoted $p(\mathbf{x}' \mid \mathbf{x})$). Consequently, the process is uniquely defined by the initial belief $p(\mathbf{x}_0)$ and the state-transition distribution $p(\mathbf{x}_k \mid \mathbf{x}_{k-1})$ for $k = 1, \ldots, K$.

In this work, we focus on the scenarios where the Markov process is *unknown* and must be learned from data. To this end, we assume two sets of data are given: initial state data $\mathcal{D}_{\mathbf{x}_0} = \{(\hat{\mathbf{x}}_0)_i\}_{i=1}^{N_0}$ and state-transition data $\mathcal{D}_{\mathbf{x}'} = \{(\hat{\mathbf{x}}, \hat{\mathbf{x}}')_i\}_{i=1}^{N}$, where $\hat{\mathbf{x}} \in \mathbb{U}^d$ is a point and $\hat{\mathbf{x}}' \in \mathbb{U}^d$ is a realization of the one-step evolution of the process from $\hat{\mathbf{x}}$.

To learn the process, we consider a parametric density function $p_\theta(\mathbf{x}_0)$ and conditional density function $p_\theta(\mathbf{x}_k \mid \mathbf{x}_{k-1})$, where parameters θ must be learned from $\mathcal{D}_{\mathbf{x}_0}$ and $\mathcal{D}_{\mathbf{x}'}$. Equipped with $p_\theta(\mathbf{x}_0)$ and $p_\theta(\mathbf{x}_k \mid \mathbf{x}_{k-1})$, inference of future beliefs can be performed via recursive *belief propagation*:

$$p_\theta(\mathbf{x}_k) = \int_{\mathbf{x}_{k-1}} p_\theta(\mathbf{x}_k \mid \mathbf{x}_{k-1}) p_\theta(\mathbf{x}_{k-1}) d\mathbf{x}_{k-1} \quad (1)$$

starting from $p_\theta(\mathbf{x}_0)$. However, for many classes of conditional density estimators, the integral in Equation (1) is analytically intractable.

[1] Any random variable in $\mathbb{R}^d$ can be mapped to $\mathbb{U}^d$ via diffeomorphisms (such as $\mathrm{erf}(\cdot)$) without inhibiting integration capability. For details, see Amorese and Lahijanian (2026).

This paper tackles this challenge by choosing a functional form for learning $p_\theta(\mathbf{x}_0)$ and $p_\theta(\mathbf{x}' \mid \mathbf{x})$ subject to the following three criteria:

C1. **Representational capacity**: with enough parameters, the distributions can approximate the true distribution $p(\mathbf{x}_0)$ and conditional distribution $p(\mathbf{x}' \mid \mathbf{x})$ arbitrary well.

C2. **Analytical belief propagation**: Equation (1) can be computed exactly.

C3. **Sparse parameter representation**: the model can be stored with polynomial memory complexity with respect to the dimension d.

This Markov Process learning problem can be formally stated as follows.

Problem 1. Given datasets $\mathcal{D}_{\mathbf{x}_0}$ and $\mathcal{D}_{\mathbf{x}'}$ generated by a Markov process and a time horizon $K \in \mathbb{N}$, learn $p_\theta(\mathbf{x}_0)$ and $p_\theta(\mathbf{x}' \mid \mathbf{x})$ that adheres to Conditions C1- C3 and compute $p_\theta(\mathbf{x}_K)$.

Describing a class of models that adheres to all three conditions is a very challenging problem. For instance, Gaussian-mixture models with linear conditional dependence satisfy C2 and C3 at the cost of very limited representational capacity. Generative models satisfy C1 and C3, but require approximation or sampling to generate predictions. Lastly, dense Bernstein polynomial models satisfy C1 and C2, but struggle to scale due to exponential blow-up in parameters, violating C3.

Problem 1 is posed as a constrained functional optimization problem. Our approach is to restrict the class of functions to expressive Sum-of-Squares (SoS) forms that emit tractable normalization and non-negativity constraints as well as analytical belief propagation. In Section 3, we provide an overview of SoS functions and their properties and then detail our approach in Section 4.

For the remainder of the paper, we drop the subscript θ and assume densities $p(\cdot)$ refer to the parameterized model.

3 BACKGROUND: SUM-OF-SQUARES

To provide the necessary background for our methodology, we first review the key principles of Sum-of-Squares theory. Consider a vector of n basis functions $\mathbf{b}(\mathbf{x}) = [b_1(\mathbf{x}), ..., b_n(\mathbf{x})]^T \in \mathcal{C}(\mathbb{U}^d)^n$.

Definition 1 (Sum-of-Squares)**.** A function $f \in \mathcal{C}(\mathbb{U}^d)$ is *Sum-of-Squares* iff f can be written as $f(\mathbf{x}) = \mathbf{b}^T(\mathbf{x})\mathbf{Q}\mathbf{b}(\mathbf{x})$ such that $\mathbf{Q} \in \mathbb{R}^{d\times d}$ is positive semi-definite (PSD), i.e., $\mathbf{Q} \succeq 0$. The set of all SoS functions is denoted $\Sigma[\mathcal{C}] \subset \mathcal{C}(\mathbb{U}^d)$.

It is straightforward to show that if $f \in \Sigma[\mathcal{C}]$ then $f \geq 0$ for all $\mathbf{x} \in \mathbb{U}^d$. Specifically, since $\mathbf{Q} \succeq 0$, it can be factored as $\mathbf{Q} = \mathbf{L}^T\mathbf{L}$ where $\mathbf{L} \in \mathbb{R}^{d\times d}$. Then, $f(\mathbf{x}) = (\mathbf{L}\mathbf{b}(\mathbf{x}))^T(\mathbf{L}\mathbf{b}(\mathbf{x}))$, which is the inner product of a vector with itself.

Ensuring that a function is non-negative over a domain is a NP-hard problem in general (Parrilo, 2003), making functional optimization over non-negative functions intractable. SoS forms offer a *relaxation* of non-negative functions via a tractable PSD constraint. Additionally, SoS forms possess rich representational properties for some choices of basis functions (Putinar, 1993; Nie and Schweighofer, 2007). Given a fixed set of basis functions, optimization over non-negative functions is relaxed to optimization over PSD matrices, as is done in Semi-Definite Programming (SDP) (Vandenberghe and Boyd, 1996). We employ techniques from SoS theory to optimize over density functions, where ensuring non-negativity is essential.

4 SUM-OF-SQUARE FORM DENSITY ESTIMATION

To approach Problem 1, we propose models of the (conditional) density estimators as SoS forms. The benefits of this are three-fold. Firstly, SoS forms allow for rich non-negative functional representation (C1). Second, with basis functions belonging to a class of functions with attractive closed-form-integrable properties, belief propagation in Equation (1) can be performed analytically (C2). Lastly, general SoS forms do not put any restrictions (such as non-negativity) on the type or quantity of basis functions themselves, allowing for sparse representations (C3).

For the remainder of this section, we focus on conditional density estimators (CDE). Let $f : \mathbb{U}^d \times \mathbb{U}^d \to \mathbb{R}_{\geq 0}$ denote a (multivariate) function representation of the conditional distribution $f(\mathbf{x}, \mathbf{x}') = p(\mathbf{x}' \mid \mathbf{x})$. For f to be a valid conditional distribution, two fundamental properties must hold:

$$f(\mathbf{x}, \mathbf{x}') \geq 0 \qquad \forall \mathbf{x}, \mathbf{x}' \in \mathbb{U}^d, \tag{2}$$

$$\int_{\mathbb{U}^d} f(\mathbf{x}, \mathbf{x}')d\mathbf{x}' = 1 \qquad \forall \mathbf{x} \in \mathbb{U}^d. \tag{3}$$

Let f be a SoS form

$$f(\mathbf{x}, \mathbf{x}') = \mathbf{b}^T(\mathbf{x}, \mathbf{x}')\ \mathbf{Q}\ \mathbf{b}(\mathbf{x}, \mathbf{x}'). \tag{4}$$

By construction, if $\mathbf{Q} \succeq 0$, then Equation (2) is satisfied. The difficulty lies in ensuring that the condition in Equation (3) also holds. In fact, in Section 4.2, we study the inherent restrictions of (4), when satisfying the normalization condition in (3).

4.1 Conditional Normalization

The models proposed in this work revolve around the ability to perform exact analytical integrals of the density. Analyti-

cal integration is crucial to formulating feasible normalization constraints, as well as analytical belief propagation. To study the normalization criterion in Equation (3), we must first formally define an algebra of basis functions which possess closed-form anti-derivatives.

Definition 2 (Integrable Multiplicative-Algebra)**.** An *Integrable Multiplicative-Algebra* is a class of functions $\mathcal{A} \subset \mathcal{C}(\mathbb{U}^d)$ such that (i) for any $b_i, b_j \in \mathcal{A}$,

$$b_i(\mathbf{x})b_j(\mathbf{x}) \in \mathcal{A} \qquad \text{(multiplicative closure)}, \tag{5}$$

and (ii) for every $b(\mathbf{x}) \in \mathcal{A}$, its antiderivative $B(\mathbf{x})$ is known in closed form, i.e.,

$$\frac{\partial}{\partial \mathbf{x}_1} \cdots \frac{\partial}{\partial \mathbf{x}_d} B(\mathbf{x}) = b(\mathbf{x}). \tag{6}$$

There are many known integrable multiplicative-algebras. A non-exhaustive list is: polynomials with integer or real exponents, trigonometric functions with linear arguments, exponential functions with linear arguments, Gaussian kernels, Beta-PDFs, etc.

Therefore, restricting the family of basis functions to the integrable multiplicative-algebra facilitates exact analytical integration. Additionally, we make one more structural assumption on the basis functions to allow us to understand functional properties while remaining general to arbitrary choices of basis function families.

Assumption 1. The basis functions are separable in the dependent variable $\mathbf{x}'$ and conditioner variable $\mathbf{x}$. Namely, $\mathbf{b}(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x}) \odot \boldsymbol{\psi}(\mathbf{x}')$, for some arbitrary $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ such that $b_i(\mathbf{x}, \mathbf{x}') = \phi_i(\mathbf{x})\psi_i(\mathbf{x}')$.

At first glance, it may seem that Assumption 1 is restrictive; however, monomial basis functions (a common choice for SoS expressive functional optimization) are separable in all variables. Moreover, Assumption 1 does *not* require $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ to be separable in the components of the state, e.g., $\phi(x_1, x_2)$ does not need to be equal to $\phi_1(x_1)\phi_2(x_2)$.

To formulate the normalization criterion in Equation (3), we can rewrite Equation (4) in a double-sum representation

$$\begin{aligned}\int_{\mathbb{U}^d} f(\mathbf{x},&\mathbf{x}')d\mathbf{x}' \\ &= \int_{\mathbb{U}^d} \sum_{i=1}^{n}\sum_{j=1}^{n} q_{i,j}\phi_i(\mathbf{x})\psi_i(\mathbf{x}')\phi_j(\mathbf{x})\psi_j(\mathbf{x}')d\mathbf{x}' \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} q_{i,j}\phi_i(\mathbf{x})\phi_j(\mathbf{x})\langle\psi_i, \psi_j\rangle \end{aligned} \tag{7}$$

Let Γ be the Gram matrix of $\boldsymbol{\psi}(\cdot)$ where each element $\gamma_{i,j} = \langle\psi_i, \psi_j\rangle$. The following property holds.

Lemma 1. *If $f(\mathbf{x}, \mathbf{x}')$ is a SoS form then $\int_{\mathbb{U}^d} f(\mathbf{x}, \mathbf{x}')d\mathbf{x}'$ is also a SoS form.*

Proof. Γ is a Gram matrix which is guaranteed to be PSD. Substituting $\gamma_{i,j}$, (7) can be re-expressed as $\boldsymbol{\phi}^T(\mathbf{x})\big(\Gamma \odot \mathbf{Q}\big)\boldsymbol{\phi}(\mathbf{x})$. By the Schur product theorem, $\Gamma \odot \mathbf{Q}$ is PSD. □

4.2 Restrictions of SoS-form CDEs

Lemma 1 describes a key property of the proposed SoS form: integrating out the dependent variable induces another SoS form in just the $\mathbf{x}$-basis. This section highlights a counter-intuitive implication of this result. Despite having rich functional approximation properties, the SoS form in (4) requires a highly restrictive structure when modeling a valid conditional distribution.

Accounting for the symmetry in $\mathbf{Q}$, Equation (3) can be rewritten as

$$\sum_{i=1}^{n} q_{i,i}\gamma_{i,i}\phi_i^2(\mathbf{x}) + \sum_{i=1}^{n}\sum_{j=i+1}^{n} 2q_{i,j}\gamma_{i,j}\phi_i(\mathbf{x})\phi_j(\mathbf{x}) = 1. \tag{8}$$

Let $\mathbf{B}(\mathbf{x}) = \mathbf{b}(\mathbf{x})\mathbf{b}^T(\mathbf{x})$ and $\mathbf{b}_\triangle(\mathbf{x})$ denote the vector of all upper triangular elements of $\mathbf{B}(\mathbf{x})$.

Lemma 2. *For Equation* (8) *to hold, either (i) $q_{i,j} = 0$ for all non-constant basis function products $\phi_i(\mathbf{x})\phi_j(\mathbf{x})$, or (ii) there is linear dependence between at least two $\phi_i(\mathbf{x})\phi_j(\mathbf{x})$ and $\phi_k(\mathbf{x})\phi_l(\mathbf{x})$.*

Proof. Suppose all $\phi_i(\mathbf{x})\phi_j(\mathbf{x})$ are linearly independent for $1 \le i < j \le n$, i.e., for a vector $\mathbf{a} \in \mathbb{R}^{n(n-1)/2}$,

$$\mathbf{a}^T\mathbf{b}_\triangle(\mathbf{x}) = 0 \;\forall \mathbf{x} \in \mathbb{U}^d \iff \mathbf{a} = \mathbf{0}. \tag{9}$$

Following from (9), the partial derivatives satisfy

$$\frac{\partial}{\partial x_i}\mathbf{a}^T\mathbf{b}_\triangle(\mathbf{x}) = 0 \;\forall \mathbf{x} \in \mathbb{U}^d \;\forall i \in \{1, \ldots, n\} \tag{10}$$

holds only if $a_j = 0$ for all non-constant elements of $\mathbf{b}_\triangle(\mathbf{x})$. Taking the partial derivatives of the normalization criterion in (8) yields the same equations as in (10), and thus $q_{i,j} = 0$ for all non-constant $\phi_i(\mathbf{x})\phi_j(\mathbf{x})$. □

Lemma 2 clarifies the necessity of linear dependence between products of basis functions. Intuitively, all non-constant functions appearing in (8) must sum to zero, leaving only a constant term. More generally, the restrictions on $\boldsymbol{\phi}$ are as follows.

Theorem 1. *Let $\mathbf{b}(\mathbf{x}, \mathbf{x}')$ be a set of linearly independent, $\mathbf{x} - \mathbf{x}'$ separable basis functions. Then $f(\mathbf{x}, \mathbf{x}') = \mathbf{b}^T(\mathbf{x}, \mathbf{x}')\ \mathbf{Q}\ \mathbf{b}(\mathbf{x}, \mathbf{x}')$, with $\mathbf{Q} \succeq 0$, satisfies condition in Equation* (3) *if and only if*

$$\|\boldsymbol{\xi}(\mathbf{x})\| = 1 \tag{11}$$

where $\boldsymbol{\xi}(\mathbf{x}) = \big(\Gamma \odot \mathbf{Q}\big)^{1/2}\boldsymbol{\phi}(\mathbf{x})$ and $(\cdot)^{1/2}$ denotes a matrix square root.

Proof. Following from Lemma 1, $\Gamma \odot \mathbf{Q}$ is PSD, and therefore can always be decomposed using a matrix square root as $\Gamma \odot \mathbf{Q} = \mathbf{M}^T\mathbf{M}$. The LHS of (3) can then be written as

$$\begin{aligned}\boldsymbol{\phi}^T(\mathbf{x})\big(\Gamma \odot \mathbf{Q}\big)\boldsymbol{\phi}(\mathbf{x}) =& \Big(\boldsymbol{\phi}^T(\mathbf{x})\mathbf{M}^T\Big)\Big(\mathbf{M}\boldsymbol{\phi}(\mathbf{x})\Big) \\ =& \|\boldsymbol{\xi}(\mathbf{x})\|^2 \end{aligned} \tag{12}$$

Constraint (3) is therefore equivalent to (11). □

Theorem 1 illuminates a critically limiting property of SoS forms for conditional density estimation. Specifically, $\boldsymbol{\phi}(\mathbf{x})$ is restricted to functions that parameterize an ellipse manifold. Such a $\boldsymbol{\phi}$ can be constructed by normalizing the output, i.e.,

$$\boldsymbol{\xi}(\mathbf{x}) = \frac{\tilde{\boldsymbol{\xi}}(\mathbf{x})}{\|\tilde{\boldsymbol{\xi}}(\mathbf{x})\|} \tag{13}$$

where $\tilde{\boldsymbol{\xi}}$ is some arbitrary basis. However, constructing $\boldsymbol{\phi}$ via (13) generally necessitates non-constant functions in the denominator, and therefore forfeits analytical integrability, i.e. $\boldsymbol{\xi} \notin \mathcal{A}$ even if $\tilde{\boldsymbol{\xi}} \in \mathcal{A}$. There exist special choices of $\boldsymbol{\phi}$ that automatically satisfy (11) without normalization, namely, when $\boldsymbol{\xi}(\mathbf{x}) \odot \boldsymbol{\xi}(\mathbf{x})$ is a partition of unity (e.g., when $\boldsymbol{\xi}(\mathbf{x})$ is a Bernstein polynomial basis as in Amorese and Lahijanian (2026)). However, such special structures are often rigid and do not allow for sparse representations and/or optimization of the parameters of the basis functions.

To overcome this critical restriction, we present a rational SoS form for conditional density estimation that employs the flexible and expressive SoS form, while bypassing the aforementioned restrictions of the normalization constraint.

5 RATIONAL FACTOR FORM

We propose the following *rational-factor* (RF) form for modeling each piece of the Markov Process by introducing a factor function $g(\cdot)$:

$$p(\mathbf{x}'|\mathbf{x}) = \frac{g(\mathbf{x}')\tilde{f}(\mathbf{x},\mathbf{x}')}{g(\mathbf{x})} \tag{14}$$

$$p(\mathbf{x}_0) = g(\mathbf{x}_0)h_0(\mathbf{x}_0) \tag{15}$$

where $\tilde{f}$, g, and h_0 are all SoS form functionals. By ensuring $g(\cdot) > 0$, equations (14) and (15) are non-negative and well defined. Let $g(\cdot) = \boldsymbol{\phi}_g^T(\cdot)\mathbf{R}\boldsymbol{\phi}_g(\cdot)$. To guarantee that g is strictly positive, it is sufficient to ensure that $\mathbf{R} \succ 0$ and $\phi_g(\cdot) \neq 0$ on $\mathbb{U}^d$ for at least one $\phi_g(\cdot) \in \boldsymbol{\phi}_g(\cdot)$.

Crucially, the RF form maintains the analytical feasibility of belief propagation. This can be seen by using (1) to compute $p(\mathbf{x}_1)$ assuming the RF form:

$$\begin{aligned} p(\mathbf{x}_1) &= \int_{\mathbf{x}_0} p(\mathbf{x}_1|\mathbf{x}_0)p(\mathbf{x}_0)d\mathbf{x}_0 \\ &= \int_{\mathbf{x}_0} \frac{g(\mathbf{x}_1)\tilde{f}(\mathbf{x}_0,\mathbf{x}_1)}{g(\mathbf{x}_0)}\big(g(\mathbf{x}_0)h_0(\mathbf{x}_0)\big)d\mathbf{x}_0 \\ &= g(\mathbf{x}_1)\int_{\mathbf{x}_0} \tilde{f}(\mathbf{x}_0,\mathbf{x}_1)h_0(\mathbf{x}_0)d\mathbf{x}_0 \qquad (16) \\ &= g(\mathbf{x}_1)h_1(\mathbf{x}_1). \qquad (17) \end{aligned}$$

with $h_1(\mathbf{x}_1) := \int_{\mathbf{x}_0} \tilde{f}(\mathbf{x}_0,\mathbf{x}_1)h_0(\mathbf{x}_0)d\mathbf{x}_0$. Since $\tilde{f}(\mathbf{x}_0,\mathbf{x}_1) \in \mathcal{A}$ and $h_0(\mathbf{x}_0) \in \mathcal{A}$, $\tilde{f}(\mathbf{x}_0,\mathbf{x}_1)h_0(\mathbf{x}_0) \in \mathcal{A}$ and thus the integral in (16) is analytically feasible. While h_1 may not necessarily be of SoS form, it is guaranteed to be in $\mathcal{A}$. Thus, one can recursively repeat the above procedure to compute $p(\mathbf{x}_k)$ for any $k > 0$.

5.1 Formulating the Conditional Normalization Constraint

The RF form naturally permits non-negativity, and analytical belief propagation, leaving only the formulation of the normalization constraint. Intuitively, $g(\mathbf{x})$ is responsible for normalizing $\tilde{f}$ at each conditioner $\mathbf{x}_{k-1}$. As a consequence, the expressions of $\tilde{f}$ and h_0 must adjust their density expressions to compensate for the factor $g(\mathbf{x}')$. Assuming RF form, the constraint in Equation (3) can be rewritten as

$$\begin{aligned} &\int_{\mathbf{x}'} \frac{g(\mathbf{x}')\tilde{f}(\mathbf{x},\mathbf{x}')}{g(\mathbf{x})}d\mathbf{x}' = 1 \\ \implies &\int_{\mathbf{x}'} g(\mathbf{x}')\tilde{f}(\mathbf{x},\mathbf{x}')d\mathbf{x}' = g(\mathbf{x}). \end{aligned} \tag{18}$$

Using SoS forms,

$$\begin{aligned}\int_{\mathbf{x}'} \Big(\boldsymbol{\phi}_g^T(\mathbf{x}')\mathbf{R}\boldsymbol{\phi}_g(\mathbf{x}')\Big)\Big(\mathbf{b}_{\tilde{f}}(\mathbf{x},\mathbf{x}')^T\mathbf{Q}\mathbf{b}_{\tilde{f}}(\mathbf{x},\mathbf{x}')\Big)d\mathbf{x}' \\ = \boldsymbol{\phi}_g^T(\mathbf{x})\mathbf{R}\boldsymbol{\phi}_g(\mathbf{x}) \end{aligned} \tag{19}$$

where $\mathbf{b}_{\tilde{f}}(\mathbf{x},\mathbf{x}') = \boldsymbol{\phi}_{\tilde{f}}(\mathbf{x}) \odot \boldsymbol{\psi}_{\tilde{f}}(\mathbf{x}')$. Integrating over $\mathbf{x}'$ in the LHS of (19) yields a linear combination of products of $\boldsymbol{\phi}_{\tilde{f}}(\mathbf{x})$, whereas the RHS is a linear combination of products of $\boldsymbol{\phi}_g(\mathbf{x})$. Thus, equality (for non-trivial cases) requires g and $\tilde{f}$ to span the same function space. This can be enforced simply by letting g and $\tilde{f}$ share the same $\mathbf{x}$-basis functions, i.e., $\boldsymbol{\phi}_{\tilde{f}}(\mathbf{x}) = \boldsymbol{\phi}_g(\mathbf{x})$, which we simply denote as $\boldsymbol{\phi}(\mathbf{x})$. Normalization can then be formulated via equality between coefficients for each respective product function $\phi_i(\mathbf{x})\phi_j(\mathbf{x})$.

For ease of presentation, we denote $\mathbf{v}^{\mathbf{R}}, \mathbf{v}^{\mathbf{Q}} \in \mathbb{R}^{n^2}$ as the vectorization of the elements of $\mathbf{R}$ and $\mathbf{Q}$ respectively. Each element $v^{\mathbf{R}}_{(i,j)}$ of $\mathbf{v}^{\mathbf{R}}$ corresponds to a product of basis functions $\phi_i(\mathbf{x})\phi_j(\mathbf{x})$. Then, the normalization constraint can be expressed by a simple linear relationship.

Lemma 3. *The following linear relationship is sufficient for satisfying the normalization constraint* (19)

$$\mathbf{v}^{\mathbf{R}} = \mathit{diag}(\mathbf{v}^{\mathbf{Q}})\mathbf{E}\mathbf{v}^{\mathbf{R}}, \tag{20}$$

where $\mathbf{v}^{\mathbf{R}}$ *and* $\mathbf{v}^{\mathbf{Q}}$ *are vectors containing the elements in* $\mathbf{R}$ *and* $\mathbf{Q}$ *respectively and* $\mathbf{E} \in \mathbb{R}^{n^2 \times n^2}$ *is a matrix that depends only on* $\boldsymbol{\phi}(\cdot)$ *and* $\boldsymbol{\psi}_{\tilde{f}}(\cdot)$ *with elements*

$$e_{(i,j),(k,l)} = \left\langle \phi_i(\mathbf{x}')\phi_j(\mathbf{x}'), \psi_k(\mathbf{x}')\psi_l(\mathbf{x}') \right\rangle. \tag{21}$$

The proof is provided in the Appendix. Lemma 3 illuminates the circularity of using the same function $g(\cdot)$ in the denominator (with argument $\mathbf{x}$) and as a factor in the numerator (with argument $\mathbf{x}'$). With $\mathbf{Q} \succeq 0$, ensuring the elements of $\mathbf{R}$ satisfy the condition in Equation (20) and $\mathbf{R} \succ 0$ yields a valid CDE.

Once a valid RF form CDE is obtained, and thus $g(\cdot)$ is known, learning the initial belief is straightforward. Since $g(\cdot) > 0$, density estimation of $p(\mathbf{x}_0)$ can be equivalently recast as using $h_0(\mathbf{x}_0)$ to approximate $p(\mathbf{x}_0)(g(\mathbf{x}_0))^{-1}$. Since $g(\mathbf{x}_0)h_0(\mathbf{x}_0) \in \mathcal{A}$ and is analytically integrable, the normalization constant can be obtained exactly.

Remark 1. *The RF form is not restricted to SoS functionals. By substituting non-negative linear combinations of non-negative basis functions for* $\tilde{f}$*,* g*, and* h_0*, the estimators can inherit the same properties as SoS forms. However, doing so greatly restricts the choice of* $\mathcal{A}$*, as many universal approximators (e.g., monomial-basis polynomials) lose their universality when restricted to non-negative coefficients.*

5.2 Time and Memory Complexity

Let n_θ be the number of parameters of a basis function in $\boldsymbol{\phi}$ or $\boldsymbol{\psi}$. Then the model can be stored with $\mathcal{O}(n^2+n_\theta)$ parameters. Belief propagation has time complexity $\mathcal{O}(n_\theta n^4)$ dominated by the product of two SoS forms in Equations (16) and (21). While n_θ is often implicitly dependent on d, for many typical basis functions (e.g., monomials), this dependence is linear.

Remark 2. *The RF form can be extended to model Bayesian Networks (and time-inhomogenous Markov processes), e.g.,* $p(\mathbf{z}, \mathbf{y}, \mathbf{x})$*, with non-Markovian dependency, e.g.*

$$p(\mathbf{z}, \mathbf{y}, \mathbf{x}) = p(\mathbf{z}|\mathbf{y}, \mathbf{x})p(\mathbf{y}|\mathbf{x})p(\mathbf{x}). \tag{22}$$

In fact, since, for such models, the conditional densities are not the same and do not require the same recursion as a time-homogeneous Markov process. Thus, the factor functions g *in the numerator and denominator need not be the same, making the normalization constraint in Equation* (20) *a simple matrix equality (rather than an eigenvalue problem).*

6 CONSTRAINED TRAINING

In this section, we discuss the necessary pieces for formulating a tractable constrained optimization problem for training the Markov Process. Recall, for $p(\mathbf{x}'|\mathbf{x})$ to be a valid CDE, both $\mathbf{Q}$ and $\mathbf{R}$ must belong to the PSD and PD cones respectively, and are subject to the equality constraints in Equation (20). Fortunately, the constraints mirror those in a SDP (Vandenberghe and Boyd, 1996). Therefore, we can leverage SDP techniques to enforce the constraints during training.

The learning problem, however, cannot be formulated as an SDP and is non-convex for two reasons. Firstly, common density estimation objectives, such as Expected Log Likelihood (LLH), do not admit a linear objective function required by SDP. Secondly, the proposed functional form permits *optimization of the basis function parameters* themselves. This allows the optimizer to reduce redundancy between basis functions, and thereby achieving a sparse representation, at the cost of making the problem potentially highly non-convex depending on the choice of basis functions. Due to the non-convexity and the potentially large supply of data, we use stochastic gradient-descent (SGD) methods to train both $p(\mathbf{x}'|\mathbf{x})$ and $p(\mathbf{x}_0)$.

6.1 Enforcing the Conditional Normalization Constraint

Let $\mathbf{Q} = \mathbf{L}^T\mathbf{L}$ for some unconstrained parameter matrix $\mathbf{L}$, automatically making $\mathbf{Q} \succeq 0$. Since the parameters of the basis functions are subject to change during training, the equality constraint in Equation (20) may change as well, since $\mathbf{E}$ is dependent on the basis functions. Furthermore, equality constraints in SDP are notoriously challenging, requiring specialized techniques, e.g. augmented Lagrangians (Burer and Monteiro, 2003). In practice, if the technique has not properly converged, there may be non-negligible numerical errors in the normalization of the CDE. Consequently, when predicting beliefs in the future (using Equation (16)), the numerical errors can accumulate leading to substantial errors in the predictions.

To mitigate this problem, it is highly preferable to avoid iterative equality constraint techniques and rather use a direct parameterization of $\mathbf{R}$ such that (20) holds exactly (up to floating-point precision). Rearranging Equation (20) as

$$\big(\mathrm{diag}(\mathbf{v}^{\mathbf{Q}})\mathbf{E} - I\big)\mathbf{v}^{\mathbf{R}} = \mathbf{0} \tag{23}$$

shows that a feasible $\mathbf{v}^{\mathbf{R}}$ is the eigenvector of the matrix $\mathbf{M} = \mathrm{diag}(\mathbf{v}^{\mathbf{Q}})\mathbf{E}$ corresponding to eigenvalue $\lambda^{\mathbf{M}} = 1$.

Obtaining a feasible $\mathbf{v}^{\mathbf{R}}$ can be achieved in two steps: (i) scaling $\mathbf{v}^{\mathbf{Q}}$ such that $\mathbf{M}$ has at least one real eigenvalue $\lambda^{\mathbf{M}} = 1$, and (ii) computing the corresponding eigenvector. For (i), we can simply scale $\mathbf{Q}$ by $(\lambda^{\mathbf{M}}_{>0})^{-1}$ where $\lambda^{\mathbf{M}}_{>0}$ is any positive real eigenvalue of $\mathbf{M}$. If $\mathbf{M}$ does not have at least one positive real eigenvalue, a feasible $\mathbf{R}$ can not easily be obtained without manipulating the basis functions themselves, and thus a loss penalty can be applied. However, we found that this rarely happens in practice. By rescaling $\mathbf{Q}$ by

$(\lambda^{\mathbf{M}}_{>0})^{-1}$, $\mathbf{M}$ is also effectively scaled by the same quantity, thereby guaranteeing an eigenvalue of 1. The corresponding feasible eigen vector $\mathbf{v}^{\mathbf{R}}$ can then be found via an eigen decomposition.

To enforce that $\mathbf{R} \succ 0$, we can use a log-determinant barrier, as is common in interior-point SDP solvers. Specifically, $\log\det(\mathbf{R}) \to \infty$ as any eigenvalues approach 0, i.e., the semi-definite boundary. SGD is unpredictable and may exit the feasible region during training, thus a loss penalty on the non-positive eigenvalues can be applied to guide the parameters into the PSD cone, captured by the following loss term

$$\mathcal{L}_{\mathbf{R}} = \begin{cases} \log\det(\mathbf{R}) & \text{if all } \lambda^{\mathbf{R}} \geq 0 \\ \sum_{i=1}^{n} \Big(\max(-\mathrm{Re}(\lambda_i^{\mathbf{R}}) + \epsilon, 0) & \\ \quad + \mathrm{Im}(\lambda + \epsilon) \Big)^2 & \text{otherwise} \end{cases}$$

for some $\epsilon > 0$.

The cumulative loss function for training the RF form CDE is

$$\mathcal{L} = -\mathbb{E}_{\mathbf{x}',\mathbf{x}}\big[\log p(\mathbf{x}'|\mathbf{x})\big] - \mathcal{L}_{\mathbf{R}}, \tag{24}$$

where the first term is the empirical negative log-likelihood of the model. Additional regularization loss can be added depending on the chosen family of basis functions.

7 EXPERIMENTS

The aforementioned sections layout the theoretical formulations of the RF form Markov model. However, the utility of the proposed model is contingent on its ability to learn realistic high-dimensional systems. This section validates the theoretical prowess of the RF form Markov model, achieving exact (with respect to the learned model) belief propagation for systems of up to 12 dimensions. Details of the experiment setups and system dynamics along with more elaborate discussions on the results are provided in the Appendix.

7.1 Choice of basis functions

The experiments were performed using component-wise products of 1D Beta-pdf ϕ (and similarly ψ) basis functions of the form

$$\phi_i(\mathbf{x}) = \prod_{m=1}^{d} x_m^{1-\alpha_{i,m}} (1 - x_m)^{1-\beta_{i,m}} \tag{25}$$

where $\alpha_{i,m}$ and $\beta_{i,m}$ are trained parameters. This family of basis functions was chosen for direct comparison with Bernstein Normalzing Flow (BNF), since Beta-PDFs are closely related to the Bernstein basis polynomials.

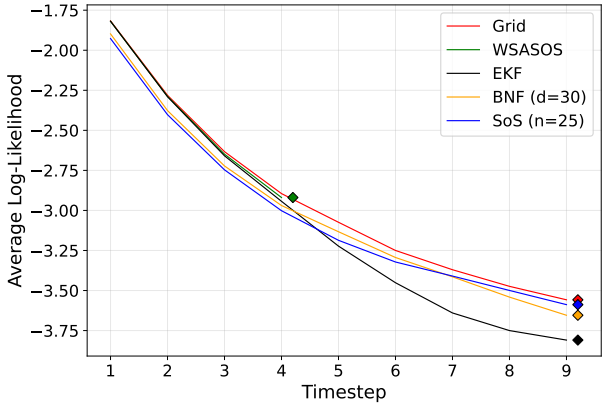

Figure 1: Accuracy Comparison on 2D system against state-of-the-art density propagation methods

7.2 2D Comparison

To ensure that the SoS form does not lose fine-grain accuracy in lower-dimensional problems, we performed an empirical comparison between SoS, BNF (Amorese and Lahijanian, 2026), and approximation methods. Specifically we compared against a Gaussian Process regression model using: traditional Extended Kalman Filter (EKF) (Schei, 1997) propagation, a grid-based Gaussian Mixture Model (GMM) method (Figueiredo et al., 2024), and a component-splitting GMM method (Kulik and LeGrand, 2024) (WSASOS). Each method was tested on data generated from a 2D Van der Pol system with additive Gaussian noise and evaluated according to the average log-likelihood over Monte Carlo samples generated from the true system. Each experiment was performed 10 times and all log-likelihood were found to have a variance across trials below 10^{-3}.

The results are shown in Figure 1. The grid-based method performs the best, since the underlying system has additive Gaussian noise. This allows the Gaussian Process regression model to learn the true system nearly perfectly. Even with a near-perfect model, EKF looses significant prediction accuracy. The WSASOS method runs out of memory after $k = 4$ due to the exponential blow up in the number of components. SoS and BNF perform very comparably with SoS having a slight advantage over BNF in the final time steps.

However, the degree 30 BNF model has over 400k parameters, whereas the $n = 25$ SoS model has only 1450 parameters. In addition to BNF's inability to scale beyond 2D, the exponential blow-up in parameters evidences strong diminishing returns when using dense Bernstein polynomials. The SoS model, however, can achieve the same performance, with orders of magnitude fewer parameters. Belief propagation using SoS is thereby also orders of magnitude faster.

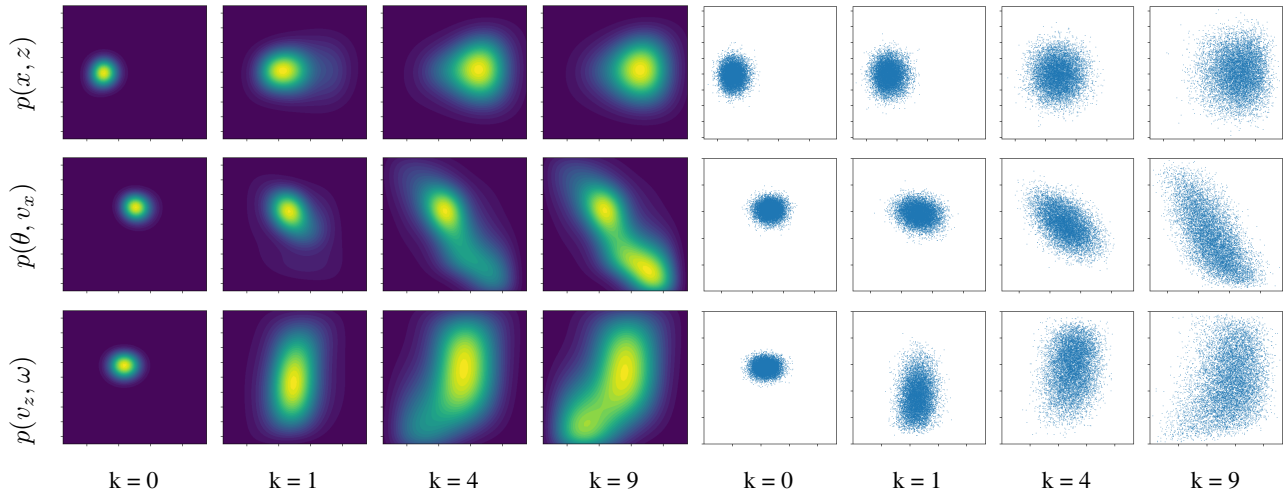

Figure 2: Visual Comparison for 6D Quadcopter system. (Left) Learned model. (Right) Monte Carlo simulation (ground truth).

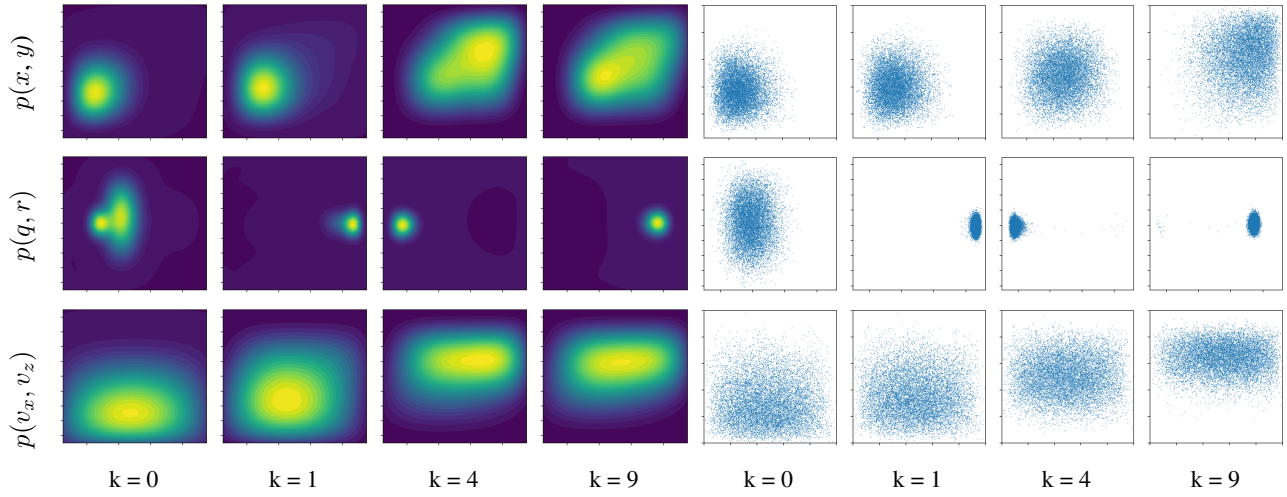

Figure 3: Visual Comparison for 12D Quadcopter system. (Left) Learned model. (Right) Monte Carlo simulation (ground truth).

7.3 6D Planar Quadcopter

We analyzed the performance of the model for learning a 6D quadcopter system subject to a state-feedback controller. The quadcopter has non-linear dynamics with additive Gaussian process noise. The state-space is defined as $\mathbf{x} = [x, y, v_x, v_y, \rho, \nu]$ where x, y are 2D planar position components, ρ, ν are the angular states and v_x, v_y are the planar velocity components. Specifically A state-space transformation was used to learn the Markov process and propagate beliefs in $\mathbb{U}^d$. To illustrate the power of the sparse model, we chose a relatively small $n = 17$ resulting in only 986 parameters for $p(\mathbf{x}'|\mathbf{x})$ and 493 parameters for $p(\mathbf{x}_0)$. The initial belief $p(\mathbf{x}_0)$ was trained using 4k data points, and $p(\mathbf{x}'|\mathbf{x})$ was trained using 40k data points. The full joint belief was propagated; however, for visualization, only pair-wise (analytical) marginal distributions are shown.

Fig. 3 shows the propagated marginal beliefs. As can be seen, the RF SoS model is able to capture the belief trends of the full 6D system. Similar to mixture models, the optimization is able to successfully allocate basis functions to achieve an expression of the density with very small memory footprint.

7.4 12D Full-state Quadcopter

To test the ability for the proposed model to scale to very high-dimensional systems, we performed a propagation experiment on a full 12D Quadcopter model. For this experiment $n = 20$ resulting in 1760 parameters for $p(\mathbf{x}'|\mathbf{x})$ and 880 parameters for $p(\mathbf{x}_0)$. The results can be seen in Fig. 3 where the position marginals $p(x, y)$, angular rate marginals

Table 1: Accuracy comparison to the deep Normalizing Flow model. Average (test) log-likelihood values are shown as mean (std).

Experiment	k=1	3	5	7	9	11	13
4D Cartpole (SoS)	-2.01 (0.0045)	-2.79 (0.0011)	-3.31 (0.0017)	-3.77 (0.0016)	-4.49 (0.0026)	-5.99 (0.017)	-8.60 (0.11)
4D Cartpole (NF)	-1.88 (0.0019)	-2.49 (0.011)	-3.12 (0.027)	-3.77 (0.056)	-4.32 (0.050)	-4.86 (0.085)	-5.41 (0.10)
6D Quadrotor (SoS)	-3.23 (0.22)	-4.56 (0.57)	-5.61 (0.84)	-6.42 (0.82)	-6.82 (0.92)	-7.06 (1.0)	-7.52 (1.2)
6D Quadrotor (NF)	-4.01 (0.0029)	-5.85 (0.013)	-7.04 (0.033)	-7.93 (0.017)	-8.69 (0.045)	-9.28 (0.031)	-9.88 (0.076)
6D Dubin's Car w/ Trailer (SoS)	-4.10 (1.1)	-5.60 (0.79)	-5.84 (0.84)	-6.33 (0.11)	-6.76 (0.15)	-6.97 (0.16)	-6.97 (0.19)
6D Dubin's Car w/ Trailer (NF)	-3.24 (0.023)	-2.96 (0.049)	-2.28 (0.054)	-3.64 (0.056)	-4.91 (0.058)	-5.64 (0.085)	-6.07 (0.086)
12D Quadcopter (SoS)	-9.54 (0.22)	-9.45 (0.024)	-9.36 (0.03)	-9.08 (0.03)	-9.20 (0.02)	-9.58 (0.01)	-10.3 (0.01)
12D Quadcopter (NF)	-8.15 (0.011)	-7.53 (0.0093)	-8.48 (0.013)	-9.36 (0.12)	-8.70 (0.0092)	-8.79 (0.0093)	-9.05 (0.010)

$p(q, r)$ and velocity marginals $p(v_x, v_y)$ are shown. The RF SoS model is able to capture the general behavior of the belief. In particular, due to a stabilization controller, the oscillation of the angular rates of the quadcopter (seen in the $p(q, r)$ marginals is captured by the belief propagation.

7.5 Comparison to Conditional Deep Generative models

This section analyzes the efficacy of the method against state-of-the-art deep neural network models capable of learning general conditional state-transition distributions. Such deep network models struggle to propagate belief densities, therefore a particle set is used as the belief approximation. The initial state data is propagated via sampling, then to compare accuracy, a GMM is fitted to each particle belief. In particular, we compare to a conditional masked-affine autoregressive normalizing flow Papamakarios et al. (2017) with 8 layers each of 128 neurons (1760 total parameters). All SoS models used $n = 17$ except for the 12D system which used $n = 20$. Each experiment was performed 12 times on a fixed train/test data set with randomized training seeds; the mean average log-likelihood (higher is better) with variance in brackets is shown. The GP-based methods generally ran out of memory, and hence, are omitted from these experiments.

As can be seen in Table 1, for moderate dimensions, the SoS method performs comparably to the NF method, even showing benefits in the 6D Quadrotor experiment. However, for the 6D Dubin's Car experiment, NF outperforms SoS since the transition distribution is very sharp (due to multiplicative noise), and thus is not as well captured by the Beta PDF basis functions used in SoS. This warrants future work on the efficacy of certain basis functions for modeling sharper transition distributions. Additionally, SoS performs slightly worse on 12D, showing that deep methods may currently possess better scalability or representation capacity.

8 CONCLUSION

We present a novel approach to modeling Markov processes that (i) can learn general Markov processes, (ii) permit analytical belief propagation, and (iii) achieve a sparse parameter representation, allowing such models to scale to high-dimensional systems. Through an in-depth theoretical analysis, we find that Sum-of-Squares representations alone are highly restrictive for modeling valid conditional distributions while maintaining analytical tractability. The proposed Rational Factor form bypasses such restrictions, modeling valid Markov process models for any choice of basis functions. Our empirical experiments confirm the proposed model's ability to leverage sparsity to scale to systems of up to 12 dimensions.

Acknowledgments

This work was supported in part by the National Science Foundation (NSF) Center for Autonomous Air Mobility & Sensing (CAAMS) under award 2137269.

References

Akhiezer, N. I. (2020). *The classical moment problem and some related questions in analysis*. SIAM.

Alspach, D. and Sorenson, H. (2003). Nonlinear bayesian estimation using gaussian sum approximations. *IEEE transactions on automatic control*, 17(4):439–448.

Amorese, P. (2026). Bernsteinflow. https://github.com/peteramorese/BernsteinFlow/tree/dev-anony-factor-sos.

Amorese, P. and Lahijanian, M. (2026). Universal learning of stochastic dynamics for exact belief propagation using bernstein normalizing flows. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 36610–36617.

Arulampalam, M. S., Maskell, S., Gordon, N., and Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on signal processing*, 50(2):174–188.

Burer, S. and Monteiro, R. D. (2003). A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical programming*, 95(2):329–357.

Figueiredo, E., Patane, A., Lahijanian, M., and Laurenti, L. (2024). Uncertainty propagation in stochastic systems via mixture models with error quantification. In *2024 IEEE 63rd Conference on Decision and Control (CDC)*, pages 328–335. IEEE.

Jasour, A., Wang, A., and Williams, B. C. (2021). Moment-based exact uncertainty propagation through nonlinear stochastic autonomous systems. *arXiv preprint arXiv:2101.12490*.

Jonschkowski, R., Rastogi, D., and Brock, O. (2018). Differentiable particle filters: End-to-end learning with algorithmic priors. *arXiv preprint arXiv:1805.11122*.

Julier, S. J. and Uhlmann, J. K. (2004). Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422.

Kulik, J. and LeGrand, K. A. (2024). Nonlinearity and uncertainty informed moment-matching gaussian mixture splitting. *arXiv preprint arXiv:2412.00343*.

Loconte, L., Mengel, S., and Vergari, A. (2025). Sum of squares circuits. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 19077–19085.

Loconte, L., Sladek, A. M., Mengel, S., Trapp, M., Solin, A., Gillis, N., and Vergari, A. (2023). Subtractive mixture models via squaring: Representation and learning. *arXiv preprint arXiv:2310.00724*.

Marteau-Ferey, U., Bach, F., and Rudi, A. (2020). Non-parametric models for non-negative functions. *Advances in neural information processing systems*, 33:12816–12826.

Nie, J. and Schweighofer, M. (2007). On the complexity of putinar's positivstellensatz. *J. of Complexity*, 23(1):135–150.

Papamakarios, G., Pavlakou, T., and Murray, I. (2017). Masked autoregressive flow for density estimation. *Advances in neural information processing systems*, 30.

Parrilo, P. A. (2003). Semidefinite programming relaxations for semialgebraic problems. *Mathematical programming*, 96(2):293–320.

Parzen, E. (1962). On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076.

Putinar, M. (1993). Positive polynomials on compact semi-algebraic sets. *Indiana University Mathematics Journal*, 42(3):969–984.

Rudi, A. and Ciliberto, C. (2021). Psd representations for effective probability models. *Advances in Neural Information Processing Systems*, 34:19411–19422.

Schei, T. S. (1997). A finite-difference method for linearization in nonlinear estimation algorithms. *Automatica*, 33(11):2053–2058.

Vandenberghe, L. and Boyd, S. (1996). Semidefinite programming. *SIAM review*, 38(1):49–95.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Not Applicable]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]

(b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]

(c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Learning Markov Processes as Sum-of-Square Forms for Analytical Belief Propagation: Supplementary Materials

A PROOFS

A.1 Proof of Lemma 3

Proof. Assuming $\boldsymbol{\phi}_g = \boldsymbol{\phi}$, the constraint (19) can be expanded into double-sum form

$$\begin{aligned}&\int_{\mathbf{x}'} \Big(\sum_{k=1}^{n}\sum_{l=1}^{n} r_{k,l}\phi_k(\mathbf{x}')\phi_l(\mathbf{x}')\Big)\Big(\sum_{i=1}^{n}\sum_{j=1}^{n} q_{i,j}\phi_i(\mathbf{x})\psi_i(\mathbf{x}')\phi_j(\mathbf{x})\psi_j(\mathbf{x}')\Big)d\mathbf{x}' \\ =&\sum_{i=1}^{n}\sum_{j=1}^{n} r_{i,j}\phi_i(\mathbf{x})\phi_j(\mathbf{x}). \end{aligned} \tag{26}$$

Rearranging the terms in the integrand and factoring out functions of $\mathbf{x}$, the RHS of (26) becomes

$$\sum_{i=1}^{n}\sum_{j=1}^{n} q_{i,j}\phi_i(\mathbf{x})\phi_j(\mathbf{x}) \sum_{k=1}^{n}\sum_{l=1}^{n} r_{k,l}\ e_{k,l,i,j} \tag{27}$$

where

$$e_{k,l,i,j} = \int_{\mathbf{x}'} \phi_k(\mathbf{x}')\phi_l(\mathbf{x}')\psi_i(\mathbf{x}')\psi_j(\mathbf{x}')d\mathbf{x}'. \tag{28}$$

The RHS and LHS of (26) therefore result in linear combinations of bilinear products of the elements in $\boldsymbol{\phi}(\mathbf{x})$. Thus, for equality, it is sufficient to match like coefficients of corresponding basis function pairs, i.e. $\phi_i(\mathbf{x})\phi_j(\mathbf{x})$:

$$r_{i,j} = q_{i,j}\sum_{k=1}^{n}\sum_{l=1}^{n} r_{k,l}e_{k,l,i,j}. \tag{29}$$

Using the matrix vectorization notation of $\mathbf{R}$ and $\mathbf{Q}$, we can see that (29) exactly matches (20) when the $((i,j),(k,l))$ element of $\mathbf{E}$ is defined as (28). In other words, (28) corresponds to a four-dimensional tensor when flattened into a matrix with a (i,j) multi-row-index and (k,l) multi-column-index. □

B ADDITIONAL EXPERIMENTAL RESULTS

B.1 Full 6D Experiment

Figure 4 shows the full evolution of the 6D quadcopter system for three marginals.

B.2 Full 12D Experiment

Figure 5 shows the full evolution of the 12D stabilizing quadcopter six different marginals.

B.3 Experimental Details

Each state-space was converted to $\mathbb{U}^d$ via mapping each state-space dimension independently through a Gaussian cumulative distribution function. For more technical details, as well as how each transformation can be chosen, refer to Amorese and

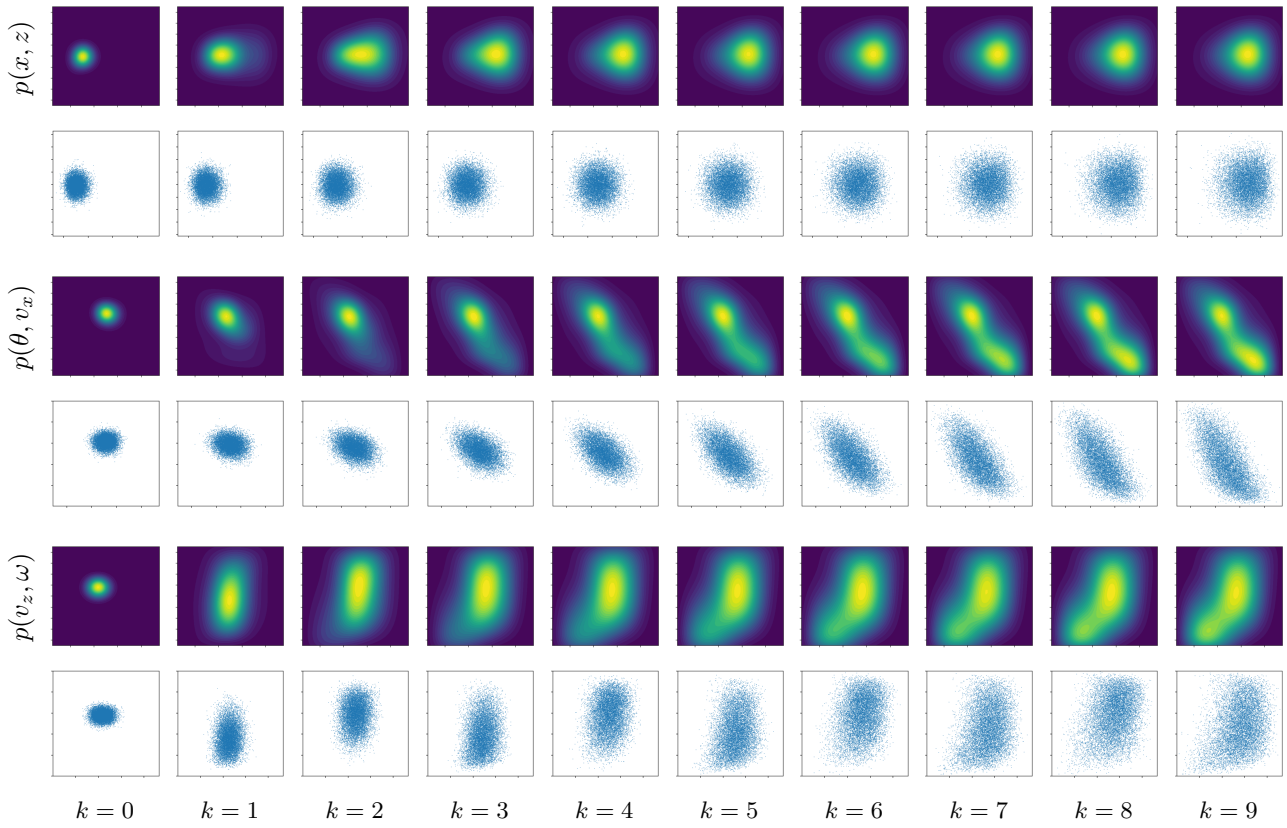

Figure 4: Visual comparison for the 6D quadcopter system across timesteps $k = 0 \dots 9$. Beliefs are shown with corresponding Monte Carlo ground truth below.

Lahijanian (2026). The full state-space dynamics equations can be found in the supplementary code in Amorese (2026), with the used parameters.

Regularization loss was applied in the form

$$\mathcal{L}_{reg} = c_{reg} \sum_i \sum_m \left(\alpha_{i,m}^2 + \beta_{i,m}^2\right). \tag{30}$$

A regularization weight of $c_{reg} = 10^{-4}$ with $\alpha, \beta \in [0.4, 400.0]$

All models were trained on GPU hardware, with training split between a NVIDIA RTX A5000 and NVIDIA GeForce RTX 2070. All models were trained in under two hours, and belief propagation was performed on the CPU in under three seconds.

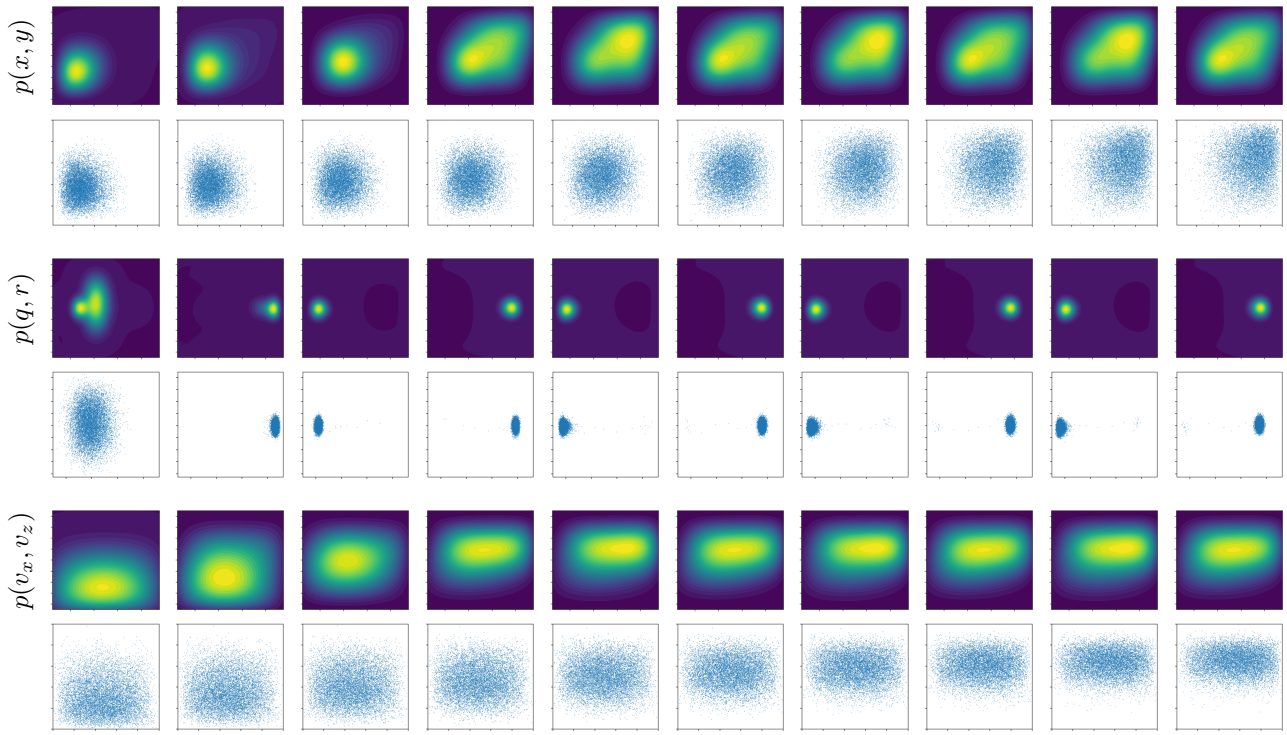

Figure 5: Visual comparison for the 12D stabilizing quadcopter system across timesteps $k = 0 \dots 9$. Beliefs are shown with corresponding Monte Carlo ground truth below. The full state definition is $\mathbf{x} = [x, y, z, \phi, \theta, \psi, v_x, v_y, v_z, \omega_p, \omega_q, \omega_r]$ with position (x, y, z) and Euler angles (ϕ, θ, ψ). Position marginals $p(x, y)$, angular velocity marginals $p(r, q)$ and translational velocity marginals $p(v_x, v_z)$ are shown.